UNIVERSITY OF THE PHILIPPINES
OPEN UNIVERSITY

Master of Information Systems

WINDE MARIS U. SORIA

MAINSPRING BOOKS SMART FINANCIAL MANAGEMENT SYSTEM (MBSFMS)

Thesis/Dissertation Adviser:
Asst. Prof. Mari Anjeli L. Crisanto
Faculty of Information and Communication Studies

Date of Submission
07 January 2020

Permission is given for the Following people to have access to this thesis/dissertation:

Available to the general public	Yes/No
Available only after consultation with author/thesis/dissertation adviser	Yes/No
Available only to those bound by confidentiality agreement	Yes/No

Student's Signature:
Signature of Thesis/Dissertation/Adviser:

This Special Project titled

_____

is hereby accepted by the Faculty of Information and Communication Studies in partial fulfillment of the requirements for the Master of Information Systems.

ASST. PROF. MARI ANJELI L. CRISANTO
Adviser
27 June 2022
Date

PROF. CONCEPCION L. KHAN
Program Chair
_____
Date

PROF. ALEXANDER G. FLOR
Dean
Faculty of Information and Communication Studies
_____
Date

ABSTRACT

MainSpring Books is a Book publishing company located in Salinas Drive, Lahug, Cebu City that offer services that give life to every author's manuscript. The company provides publishing services, online marketing and editing services to authors. As the company grows, it is difficult for them to track their financial status due to manual accounting. Manual accounting involves the recording of their business' financial information and transactions, in which orders, invoices, expenses, and bank reconciliation are maintained using Microsoft Excel Spreadsheets to store, classify and analyze financial transactions of the company [1]. This project introduces a system to the company that converts their manual accounting process to a centralized financial management system. The system is developed to improve their accounting process to a faster, efficient and accurate account reporting.

The Agile Method is the method used by the researchers in order to properly manage the activities and system functionalities.

ACKNOWLEDGMENTS

This project would not have been possible without the support and help from my friends and family. I would like to extend my sincere thanks to all of them.

My thanks and appreciations to my friend Adelio Bato-on for giving me the opportunity to do the project for your startup company and giving all the support and guidance by providing all the necessary information for developing a good system.

I would like to express my special gratitude and thanks to my husband Ken for his continued and unfailing love, support, and understanding which made the completion of this project possible. You were always around to lift me up at times when I felt it impossible to continue, you helped me keep things in perspective. I greatly appreciate your belief in me. To my babies Elena and Eli, I thank you for making me stronger, better, and more fulfilled than I could have ever imagined. I love you all so much.

TABLE OF CONTENTS

Dedicated to:
Ken Edwale T. Soria
Elena Grace U. Soria
Eli Gavin U. Soria

Chapter I

INTRODUCTION

1.1 Statement of the problem

Financial information has always been an integral part of the business decision making process. The need-to-know instant cashflow, accounts receivable, accounts payable, and profits is important for business success [2]. MainSpring Books as a growing publishing company uses manual accounting in completing financial tasks. Errors can be quite frequent in manual accounting processes. Common errors include entering incorrect order information, transposing figures or recording information backwards. A lack of security is another common disadvantage to their manual accounting. The company may be unable to prevent employees from reviewing sensitive data in spreadsheet and printed copies. Files copied and stored on a computer may also be less secure. This may allow employees to abuse financial information through fraud or embezzlement. Disgruntled employees may also irreparably damage the information and destroy important financial records [4].

1.2 Background

Accounting is the art of recording, classifying and summarizing in a significant manner and in terms of money, transactions and events which are, in part at least, of financial character, and interpreting the results thereof. Accounting can also be referred to as an information system that measures, processes and communicates financial information about an economic entity. Advancements in information technology have dramatically improved accounting systems and transformed economic life. Computers and other digital technologies have increased office productivity facilitating the rapid exchange of documents, research, and the collection and analysis of data. Information technology gave all sorts of individual economic actors the new valuable tools for identifying and pursuing economic and business opportunities [5].

In a business environment, most small businesses used spreadsheet applications to record their financial information, such as Microsoft Excel. Spreadsheets allow the company to create customized financial reports to track specific information. Such programs can perform a variety of functions such as adding and subtracting numbers simply by formatting cells on the current project file.

A variety of accounting programs are also available to any growing companies. Accounting application suites provide specific programs and reports to make an accountant’s job easier. Financial reports are maintained simply by entering information and creating reports with a set of keystrokes, many of which can be configured by the user. Many accounting programs also have industry-specific applications for different types of small businesses.

MainSpring Books is Book publishing company which offers publishing, marketing, and other services that values the creativity of Book authors all around the world. The company is located at Salinas Drive, Cebu City, Philippines with head office in Los Angeles, California USA. It started to operate in early 2018.

MainSpring Books seek authors in Amazon Books to have their publishing services. There are about thousands of authors who published their books in amazon.

The process of recording information from the ordering point of the service up to calculating the sales and expenses of the company are done in traditional way. In this case the company is storing their data through spreadsheets and other similar tools. The process might take their time tracking their sales and expenses due to growing amount of data and might cause inaccurate financial reporting. The researchers propose an accounting system that allows the users to easily manage their transactions and can generate their desired financial reports accurately and efficiently. In this way the company won't have a hard time tracking their financial matters.

1.3 Objectives of the Project

The main objective of the study is to develop a Smart Financial Management System to MainSpring Books company.

Specifically, researchers aim:

1. to determine solutions to the existing problems and difficulties regarding the tracking of sales and expense of the company.
2. to identify ways on how to improve the traditional process in recording company's transactions; and
3. to identify the different tasks involved in an accounting system

1.3 Significance of the project

MBSFMS is used to manage the income, expenses, and other financial activities of MainSpring Books company. It allows the company to keep track of all types of financial transactions, including purchases (expenses) and sales (invoices and income), etc. and is capable of generating comprehensive reporting that provide management with a clear set of data to aid in the decision-making process.

This project will be utilized by MainSpring Books company for its financial tracking system.

1.4 Scope of the project

The main goal of this project is to develop a web-based system that provides an efficient process of managing the sales and inventory of the company.

The system is capable of managing the records of book authors, services offered and orders from book authors. It also allows the user to gather information of book authors from various book selling site in the internet in a hassle free and none time consuming data research. The system also includes the process of tracking down the sales and inventory of the company, from the author's ordering point up to comprehensive reporting of company's income. The predictive analytics will also be included in the scope where in the system will estimate or predict the company's income in the future dates base on the company's provided quota.

1.4 Documentation of existence and seriousness of the problem

In business environment, strategic preparation is important and requires valid information to make key decisions. Choosing the right tools to input, track, analyze and store data will help business owners and managers make the best decisions for their company. One of the components of the software productivity suites is the spreadsheet. Spreadsheets are common among accountants and those who want to collect and record data, but there are some drawbacks that may not make them the right choice for any office use.

The advantage of spreadsheet is that it is the simplest form of tool used for collecting and organizing business data. Information can easily be placed in neat columns and rows and then sorted by information type. The great appeal of spreadsheets is that the program does all the math for the user. Once a formula is written and the program has a set command, complex calculations can easily be computed for the related data that has been inputted. In today's collaborative work environment, multiple users within an office often need access to the same documents. If using Microsoft Excel, the spreadsheets can be shared, but only one user can change data at a time.

However, the downside of spreadsheet in organizing data is that only the information that the user chooses for analysis is included in these presentations, and therefore, other pertinent information that may influence decision making might be excluded, unintentionally. In streamline calculations, the difficult part for many users, is that the calculations must be entered into the spreadsheet as formulas. This requires learning the correct syntax for each type of calculation you wish to make. Although many classes are available to learn the skills necessary to use these formulas, many users still find them difficult. If the syntax is incorrect, the program will not return the correct information when the calculations are run. Additionally, if users input the wrong data, even in only one cell of the spreadsheet, all related calculations and cells will be affected and have incorrect data. Another spreadsheet disadvantage is the lack of security for the company’s files. Typically, spreadsheets are not that secure and therefore are at greater risk for data corruption or mismanagement of information. Files that contain

sensitive financial information may not be safe from hackers, even if password protected.[6]

An estimated 1.2 billion people use Microsoft Excel. Couple that with more than 3 million businesses using Google G Suite, which includes Google Sheets. With 467 functions across 12 categories, Excel is used for a wide variety of business uses, ranging from data manipulation to accounting. A survey by FSN found that 71% of organizations still rely on spreadsheets across the majority of their business units. [7]

The past decade has been rough for businesses who rely on spreadsheets. The London Olympics accidentally sold 20,000 tickets to a swimming event that could only accommodate 10,000. According to The Telegraph, “a member of staff made a single keystroke mistake and entered ‘20,000’ into a spreadsheet rather than the correct figure of 10,000 remaining tickets.”

Chapter II

REVIEW OF EXISTING ALTERNATIVES

MainSpring Books as a starting company uses manual recording of their transactions. Manual recording is a recording method that record transactions by hand. It is the most time-consuming option for recording transactions. However, it is the cheapest solution for small business owners. When a company record transactions by hand, manually account for each transaction and calculate totals.

MainSpring Books uses Microsoft Excel for financially-related activities as an alternative resource to their accounting system. This spreadsheet software helps the company to keep track of sales leads, order status reports, and invoice reports.

The company make use of MS Excel spreadsheet software for carrying out their accounting activities. They create a basic accounting program that allows them to keep track of the organization’s financial transactions. By entering data in this manner, the company create charts and graphs over time to compare business income and expenditures.

The company also uses this spreadsheet to keep track of their service sales. By doing these, the company owner can design plans to their sales in the market. By maintaining the track of their service, company owner gets an idea about the progress has made over the specified time period and in this alternative, they can identify the high and low sales trends.

The spreadsheet is also being used by the company to store contact information of their clients and customers. This information acts as a customer database and can make use of this to contact their customers.

As the company is growing, the records are populating their spreadsheets. It is inconvenient for them to track, modify, and compare their records since they need to open multiple files and shift to multiple sheets. This process may result to multiple errors, miscalculations and inaccurate data analytics.

The researchers proposed a system to the company to make their accounting process more efficient and prevents inaccurate result of data. The system allows the company to lessen their income tracking time since it automatically generates reports based on the user's preferences. It has also additional features that can add up their service sales and improve marketing strategies. The company will no longer worry of opening multiple files since all the company transactions and records are already stored and processed in the system.

MBSFMS as an accounting system of MainSpring Books aims to give accurate and efficient income tracking system to the company. Just like the company's alternative it has the basic accounting features but the process of the system functionality is customized based on what MainSpring Books' preferences and company environment. It has data mining in which the functionality fetches data of authors and their contact information from certain websites. This feature will give ease to the company in discovering new authors as their client, this will also increase their market sales. It also has predictive analytics. Predictive analytics uses multiple techniques such as data mining, statistics, and artificial intelligence to predict the possibility of something happening based on historical data [8]. This feature allows the company to predict their future sales based on their historical and current data.

This will guide the company on their decision making in order to come up to their company goals.

Chapter III

PROJECT DETAILS

This chapter covers the process model, data model, the functionalities of the system and the methods used.

A. Overview

A web-based accounting system has been created that allows the users to easily manage their transactions and generate their desired financial reports accurately and efficiently. In this way the company won't have a hard time tracking their financial matters.

The system is capable of managing the records of book authors, services offered and orders from clients. It also allows the user to gather information of book authors from various book selling site on the internet in a hassle free and none time consuming data research. The system also includes the process of tracking down the sales of the company, from the author's ordering point up to comprehensive reporting of company's income. The predictive analytics is also included in the scope wherein the system will estimate or predict the company's income in the future dates based on the company's existing revenue.

a. Use Case

This section shows the use case diagram of the system. As shown in Figure 2, the diagram has 10 use cases, namely: Manage Users, Manage Account, Manage Services, Data Mining, Manage Clients, Manage Orders, Manage Order Costing, Verify Orders, Generate Reports, Generate Predictive Analytics. The production staff, finance staff and the administrator of the company are the actors of this system.

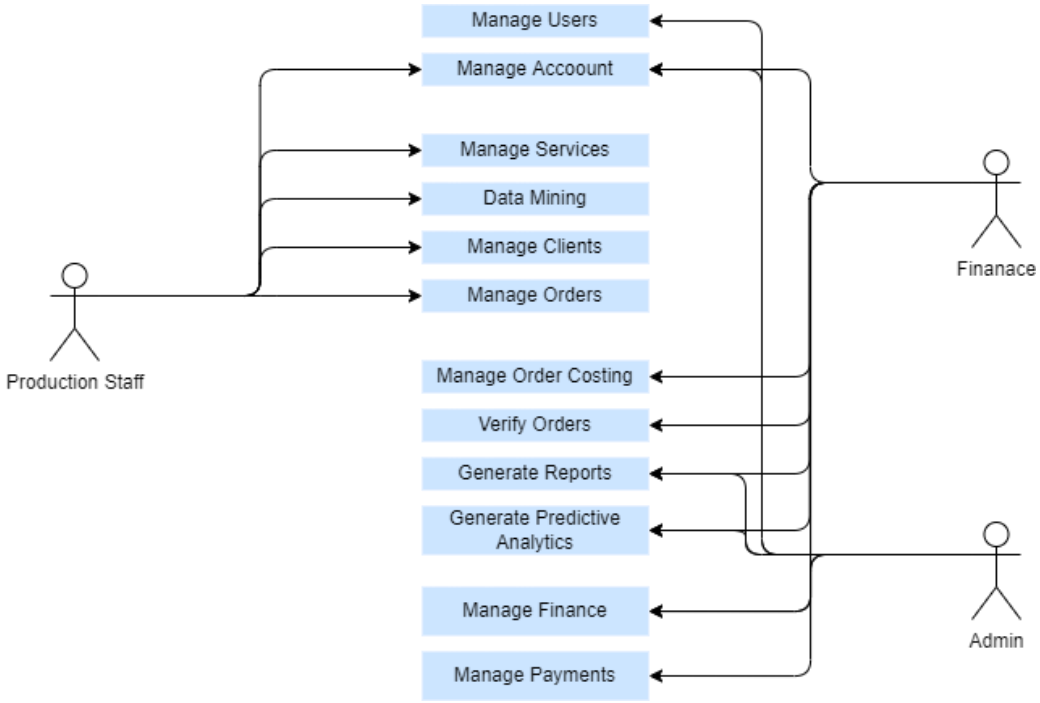


Figure 2. Use Case Diagram of MBSFMS (Overview of the system)

The admin handles 4 use cases. This includes Manage Users, Manage Account, Generate Reports, Generate Predictive Analytics. The admin manages all users and monitor each transaction in the system. The Production Staff handles 5 use cases, this includes Manage Account, manage services, Data Mining, Manage Clients, Manage Orders. The Finance staff covers 8 uses cases, this includes, Manage Account, Manage Order Costing, Verify Orders, Generate Reports, Generate Predictive Analytics, Manage Finance, and Manage Payments.

Data Mining and Generate Predictive Analytics are the key features of the system. These key features improves marketing strategy, effective planning, precise decisions and increases sales of the company.

In Manage User use case, the Admin can add, edit, delete a Production Staff, Finance Staff, and Admin into the system. All users can be added if there newly hired or designated to take the position.

In Manage Account use case, all users can modify their information and credentials into the system.

The Manage Services uses case is the primary input of service data into the system. The system allows the users to add, delete or modify packages. Packages consist of multiple service inclusions. Inclusions can also be modified by the users. Inclusions are categorized in to different service categories. Each inclusion is initialized with default price but can be modify during costing stage.

Data Mining uses case is a key feature of the system. Data mining is a process used by companies to turn raw data into useful information. By using software to look for patterns in large batches of data, businesses can learn more about their customers to develop more effective marketing strategies, increase sales and decrease costs. Data mining depends on effective data collection, warehousing, and computer processing [9]. It allows the production staff to easily navigate and discover book authors on a certain website. This function generates basic information about the author of a book, including contact information, background, and even published books.

In Manage Clients uses case, the system allows the user to register, delete, or modify client. Client cannot be deleted if there are transactions made in the system.

In Manage Order use case, the system allows the production staff to input orders from the client. This includes package selection and service inclusions. The default service inclusion of a package can be removed or if the client prefers adding of inclusions that is not part of the package. Custom package is also possible if the client prefers different service inclusions and services to be included in his/her package that is not stated to the company's services or packages.

After order form is complete by the production staff it will be sent to finance staff for order costing. The Manage Order Costing allows the finance staff to modify the system prices of inclusions. The finance staff evaluates the prices of each inclusions and finalize the amount payable of the client. In this phase the finance staff estimates the operating expenses that will be included in the order. Commission expenses will also be included in the calculation of the order.

The next step of the order will the verification. The Verify Order use case allows the finance staff to verify the orders. This will be identified if the client paid the order cost. The order details will no longer be available for modification once it is verified. The transaction will be recorded as verified in the system.

In Generate report use case, the system generates reports of detailed income status of the company. This includes pending orders, for verification order, verified orders, commission reports, production cost reports, operating

expense reports, one-time expense reports, and honorarium reports. Reports can be generated by date range specified by the user.

Generate Predictive Analytics is also one of the key features of the system. This allows the user to generate estimated expenses and revenue of specified date. The user needs to input basic data of transaction in order for the system to generate estimated data. This feature is useful for the company to have their business or income goal or to properly budget their expenses to the specified date range.

The process of revenue prediction starts with collecting the verified sales of the company since the first transaction on the system. It calculates the average monthly revenue and orders from the first transaction up to the current date. The sales of the company depends on the order packages it offers. During the prediction process of the system, it also generates revenue statistics of different order packages from the existing sales. For the system to predict the future revenue, it generates percentage of average cost per package to the total amount revenue of all packages. This percentage will be used as calculated future cost per package, the future cost will be multiplied to the predicted number of orders. The result will be the total predicted cost per package. The accumulated predicted cost per package will be the predicted revenue on specified date.

The Manage Finance functionality allows the users to manage One time expenses of the company, Honoraria of the staffs, and Operating Expenses.

In Manage payments, it allows the users to enter payment data of the clients. This can be used of the client chooses partial payments.

b. Methods

i. Design Studies

In the development of the system, the researcher used the agile method. “The Agile Method is a particular approach to project management that is utilized in software development. This method assists teams in responding to the unpredictability of constructing software. It uses incremental, iterative work sequences that are commonly known as sprints.”.

The researcher used this approach because it anticipates change and allows for much more flexibility than traditional methods. The process improves the quality of the project, by breaking the project into manageable units, and the researchers can focus on high-quality development, testing, and collaboration. Also, by producing frequent builds and conducting testing and reviews during each iteration, quality is improved by finding and fixing defects quickly and identifying expectation mismatches early.

The advantage of Agile over Waterfall model is, it is client focused process. So, it makes sure that the client is continuously involved during every stage. This can easily be implemented the fact that the respondents are just within the vicinity of the researcher’s location. Agile software development method assures that quality of the development is maintained. The process is completely based on the incremental progress. Therefore, the client and team know exactly what is complete

or not. This reduces risk in the development process. In the waterfall model, if the requirement is not clear at the beginning, it is less effective method. Very difficult to move back to make changes in the previous phases, and the testing process starts once development is over. Hence, it is high chances of bugs to be found later in the development where they are expensive to fix.

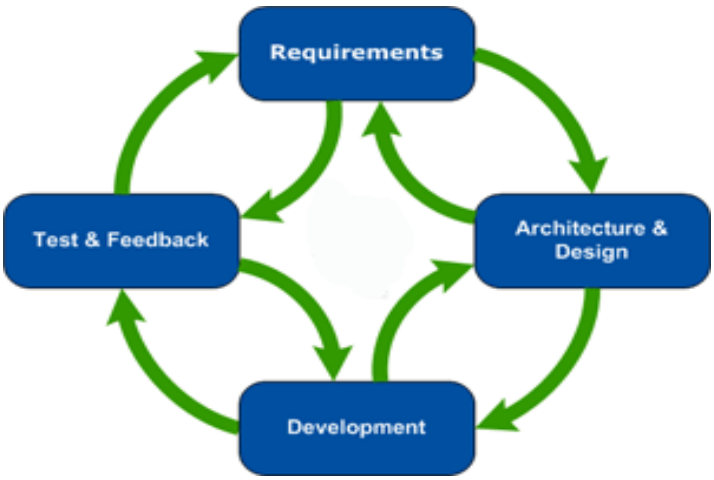


FIGURE 4: AGILE METHODOLOGY

As shown in Figure 1, the whole process is a cycle that starts at the requirement phase, in which all the data necessary for the development of the system is gathered. After gathering of data, the process proceeds to the next phase, which is the architecture and design, in this phase the researchers develop the data model of the system. After the architecture and design, it proceeds to the development phase. In this phase, the development and testing of each modules are performed. When the system works without failures, then the system is ready for the test and feedback phase. In this phase, the researches allows the users to test and give feedback about the system. Based on the diagram, the process can return to any phase if they fail or miss some data needed and

functionalities in the development process. The whole process repeats until the system works without any flaws and errors and can meet the needs of the users.

The first phase is the requirement gathering and analysis. In this phase, the company business process was analyzed. Gathering of information was done in this phase through series of interview and read selected studies. From the information gathered, the use case diagram was created.

During the system Architecture and Design phase, the researcher created the data model which represents the different data necessary for the proposed accounting system based on the information gathered during the requirement gathering and analysis phase. The MBSFMS was designed using relational database for the relationship of the tables in the database, file structure for the structure of the tables, and program hierarchy that conveys the relationship of the various functions in the system. This phase also includes choosing of technologies that are used in the system, such as the development framework and development tools. This phase specifies requirements and define the overall system architecture. The process returns to the previous phase which is the Requirement gathering and analysis if the researcher missed some data that are needed in the design phase.

The next phase is the development. In this phase, the programmer developed the modules of the system using the input of the system design. Each module undergoes unit testing before it proceeds to the next phase. Unit testing is a software development process in which the

smallest part of the system, called units, are individually and independently examined for proper operation. The process returns to the previous phase which is the Architecture and Design if some modules does not work well, because it lacks some data or some data needs to be changed in the data model in order for the module to work well during the unit testing.

In the test and feedback phase, the researcher conducted the integration testing that achieved by the developer and the process of acceptance testing and feedback conducted by the production staff, finance staff, and admin. The process returns to the Development phase if some modules does not function well according to its functionality. The whole process repeats until the system meets the users' needs without any flaws and errors.

c. Review of Existing Systems

Accounting is a fundamental part of running a successful business. Accounting software is one that cultivates under the same roof all systems and applications dedicated to managing and processing financial data. If consolidated well, it can be an essential part of the company's system and substantially improve all things finances inside the company from bookkeeping to tax accounting [8].

Accounting software helps to increase efficiency as it has the ability to manage account receivables and accounts payables, among others. The accounting software takes care of tracking all the transactions with seamless scaling and implementation of new products and services. The

benefits resulting from the speed and efficiency of accounting software often go hand-in-hand with a reduction of overall costs.

This chapter covers the examination of existing alternatives related to the study. The chapter deals with the existing software including the definition and origin.

i. Accounting Seed

Accounting Seed has core project management features such as a general ledger and a client engagement tracker built right into it. This free project accounting software is based on the Cloud. Hence, a company can access financial data easily and quickly. It enables the company to enter project activities and costs into a shareable general ledger, manage revenue through each step of the project lifecycle, handle orders and perform detailed project accounting as well [10].

ii. Sage Intacct

This is an accounting software that is capable of handling the monetary aspect of a project from its inception to completion. Sage Intacct presents all critical project information from one convenient dashboard. As a result, you can handle elements such as project scheduling, track expenses and perform project-based accounting from one interface. The project accounting software is capable of automating accounting tasks such as billing, resource management, cost tracking and recognizing revenues. By doing so, it streamlines the project construction and makes financial management easier [11].

d. Literature Review

A computerized accounting system, as mentioned by Marivic (2009), is a method or scheme in which financial data related to business transactions are being recorded, organized, summarized, analyzed, interpreted and communicated to stakeholders with the use of computers and computer-based systems. He believes that one of the vital parts of any organization is the keeping of accurate accounting records.

According to Indira (2008), there is a recognizable improvement in the business performance with the use of computerized accounting systems because it encompasses performance enhancing features. He pronounces that it enhances communication through easy and fast access of information which leads to a much quicker decision making.[12]

Littleton (2013) argues that the central purpose of accounting is to make possible the periodic marching of cost efforts and revenues accomplishments. This involves fixed point of accounting theory, and a bench mark that afford a fixed point of reference for accounting sessions. Accounting is the art of recording, classifying and summarizing in a significant manner and in terms of money, transaction and events, which are in part at least of a financial character, and interpreting the result thereof (ALCPA, 1961). In the same way, Benjamin (2008) reiterates that the primary function of accounting is to accumulate communication information essential to understanding the activities of an enterprise, whether large or small, corporate or non-corporate, private or public. Such information System is aimed at smoothing the running of an organization and eventually maximizing profits. (Anderson & Caldwale, 2011) suggest that accounting is an information system for measuring,

processing and communicating information that is useful in making economic decisions. Needles (2004) points out that opinion that accounting information System is essential to decision system because it provides qualitative information for three functions: planning, control and evaluation. Simon (2009) in his study used the first part of the statement as a measure of control for management and the second part for evaluating the effectiveness of the accounting information via continuous monitoring. Accounting information is said to be effective when it serves widely the requirements of the system users. Consequently, the effectiveness of accounting information has long been a subject of many rese (Chenhall, Kim, and Mia 2008). [13]

e. Assessment of Existing Alternatives

The researcher conducted a SWOT analysis of the owners of MainSpring Books, which included a question about how they dealt with the accounting problem and what alternative methods they used to temporarily solve the problem. The company uses spreadsheet as an alternative in order to track their financial status.

The researcher asked sample forms and reports that is in paper form or file forms for the assessment of the alternative. The samples were analyzed and come up with the designs from basic inputs of data up to accounting reports. In order to construct a more effective, user-friendly and precise framework compared to the current alternative, the recording in spreadsheet formats was reviewed by the researcher according to the transaction routine of the company.

f. Testing

The researcher has conducted a system testing that involved the actors of the system and these users also took part of the pilot implementation, to ensure that the system meet the requirements of the users and functionalities are working properly without flaws and errors. Each module has been tested according to their functionalities. The findings served as the basis for fixing bugs and enhancements to the system.

The system has been tested from functionalities up to key features. The basic inputs of the system were filled up. This includes the clients, packages, inclusions, and staffs. After the user entered the basic data, the orders were tested, and then each form was costed and the order was verified. Once the system had order transactions, this generated income reports based on what users inputted. Researcher then manually calculated the numerical output to see if the data is accurate.

The researcher asked one production staff to test the functionalities or use case of production user. To test if basic data form and order form were working according to its designed functionality. One finance staff tested order costing functionality and verifications of orders for the finance user end. The owner tested the admin end of the system, to check if user management and accounting reports are working and generate outputs according to designs.

The assessment was analyzed after all users tested their corresponding system ends. The researcher required the users to give feedbacks based on their experiences on the system. The researcher also observed the user's performance and inputs on the system. All feedbacks, observations, and

recommendations of the users was gathered, and served as the basis for revisions.

B. Theoretical Framework

MBSFMS is an accounting system of MainSpring Books company, a web-based system that will help improve the income tracking process of the company. The production staff, finance staff, and the admin are the users of the system. The production staff manage the service order of book authors. The finance staff manage the financial matter of the company, the billing and reporting are produced by this user. The admin user manages all the users and transactions of the company. The process of the system starts with, recording of client's order of the company's service. The company will perform the costing of the service in the system, adjust the default prices generated by the system if ever there are changes in the initialized prices. After the costing is completed, it will proceed to verification. This order will be identified as verified. All transactions will be recorded in the system and the company can now generate detailed reports base on specified date and data.

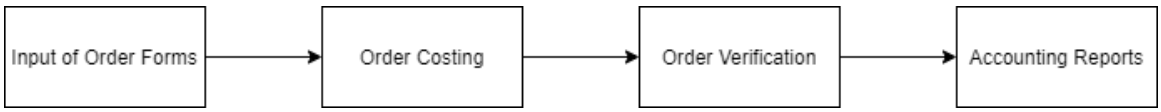


Figure 1. General System Process

a. Rationale for Framework

MainSpring Books uses spreadsheets as an alternative resource to their manual accounting. As the company uses this alternative, they manually set formulas to their calculations and the formulas are complex and prone to error. It is also susceptible to fraud because it is easy to change information and

hard to keep track of who is making the changes. Errors are easier to make in this alternative (copy/paste errors, moving content out of cells that other formulas base their calculations on, etc.), and are much harder to track down.

As a proposed solution to the company's problem, MBSFMS is an accounting system that converts the traditional accounting into a more efficient and accurate accounting process. The system handles all the calculations of numerical data. It also maintains audit trail for the company's awareness of accountabilities in case of data modifications. The ability to call reports at any given time for a multitude of different reports to see the business from many different angles. It is built to ensure the company records their accounting data effectively with minimal mistakes and good auditing.

C. Technologies Used

The MBSFMS was developed using a computer with Core i5-10210U with 12GB RAM, 1TB of Hard Disk Drive (HDD), 256 of Solid-State Drive (SSD) and windows 10 64-bit OS. PHP (Hyper text preprocessor) version 7.1 was used as the programming language with Laravel Framework 7.13. Laravel is an open-source software rapid development web framework for building dynamic websites with PHP. XAMPP 7 is a free and open source cross-platform web server solution stack package, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP. React JS was used as Front-end framework for the application. React is a front-end library developed by Facebook. It is used for handling the view layer for web and mobile apps. ReactJS allows us to create reusable UI components. It is currently one of the most popular JavaScript libraries and has a strong foundation and large community behind it.

D. System Design

This section shows the system design. This includes the Entity Relationship Diagram, File Structure, and Program Hierarchy.

a. Entity Relationship Diagram

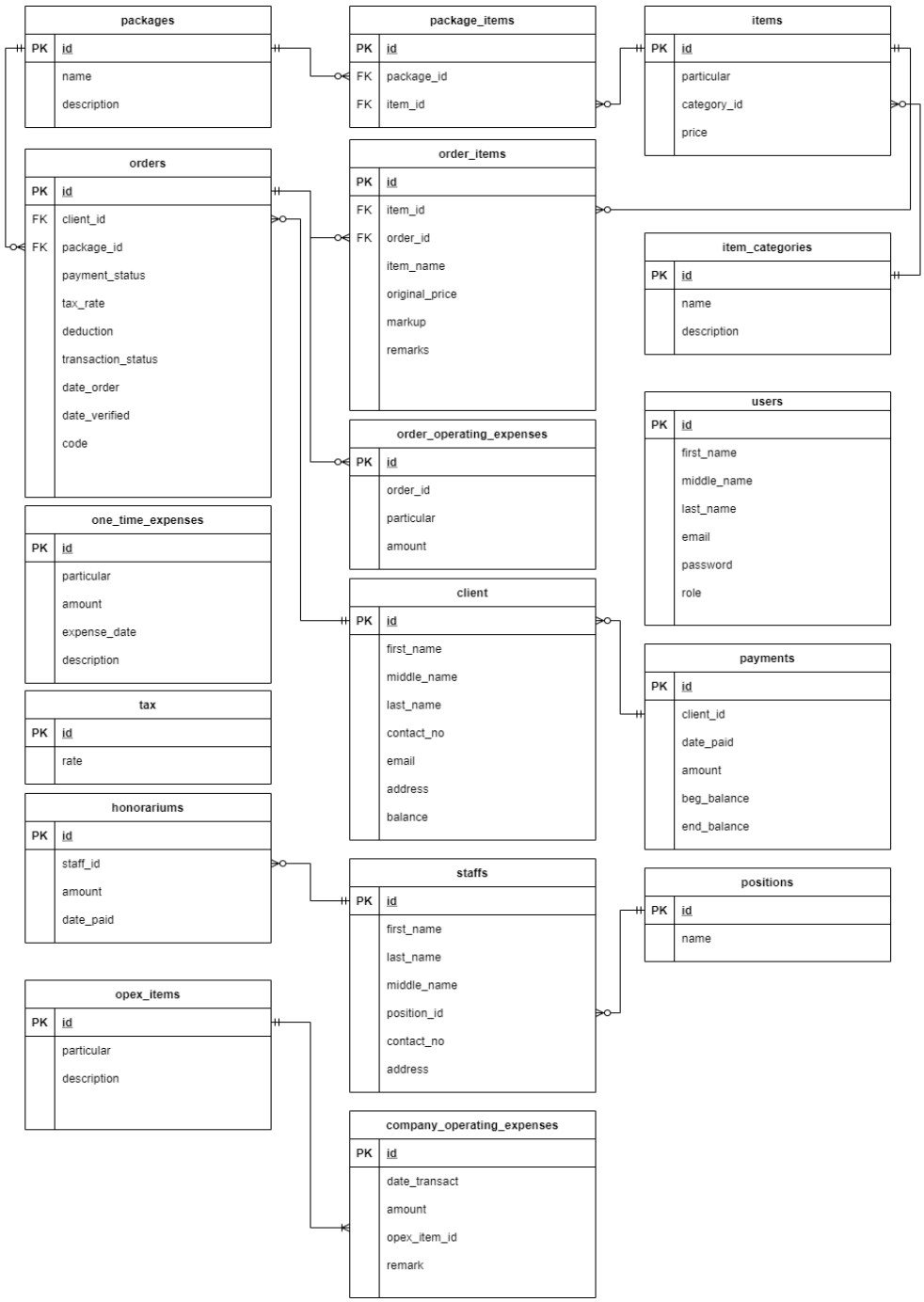


Figure 3. Entity Relationship Diagram of MBSFMS

b. File Structure

Table 1. User

A table that contains the user information.

Fieldname	Type	Description
id	int(6)	An incremental ID of user. This uniquely identifies the user.
first_name	varchar(20)	The First name of the user
middle_name	varchar(20)	The Middle name of the user
last_name	varchar(20)	The Last name of the user
email	varchar(50)	The email of the user. Used for the user to log in.
password	varchar(100)	An encrypted text of the user's password.
role	varchar(20)	Specifies the type of user.

Table 2. Tax

A table that contains the tax rate.

Fieldname	Type	Description
id	int(6)	Id of the tax rate
rate	float	The tax rate serves as the initial tax charge of every order of client. This will be calculated as percentage.

Table 3. One_time_expenses

A table that contains the expenses made by the company.

Fieldname	Type	Description
id	int(6)	An incremental ID of expenses. This uniquely identifies a certain expense.
particular	varchar(50)	Title of the expense made.
description	text	Description of the expense made
amount	float	Amount of the expense
expense_date	date	Date expense made

Table 4. Position

A table that contains the positions of the staffs.

Fieldname	Type	Description
id	int(6)	An incremental ID of position. This uniquely identifies a certain position.
name	varchar(20)	Name of the position

Table 5. Staff

A table that contains staff information.

Fieldname	Type	Description
id	int(6)	An incremental ID of the staff. This uniquely identifies a certain staff.
first_name	varchar(20)	The First name of the staff

middle_name	varchar(20)	The Middle name of the staff
last_name	varchar(20)	The Last name of the staff
position_id	int(6)	ID that links to a position
contact_no	varchar(20)	Contact no of the staff
address	varchar(50)	Address of the staff

Table 6. Honorarium

A table that contains the details of honorarium given to the staffs

Fieldname	Type	Description
id	int(6)	An incremental ID of the transaction. This uniquely identifies a certain transaction.
staff_id	int(6)	ID that links to staff table that identifies the staff given by the honorarium
amount	int(6)	Amount of honorarium given to the staff
date_paid	date	Date of the honorarium given to the staff

Table 7. Packages

A table that contains service package titles

Fieldname	Type	Description
id	int(6)	An incremental ID of the package name. This uniquely identifies a certain package.
name	varchar(20)	Package name
description	text	Description of the package

Table 8. items

A table that contains services offered by the company

Fieldname	Type	Description
id	int(6)	An incremental ID of the item. This uniquely identifies a certain item.
particular	varchar(20)	Item name
category	varchar(20)	Category of an item
price	float	Default price of the service

Table 9. package_items

A table that contains services included in a package

Fieldname	Type	Description
id	int(6)	An incremental ID of the package item. This uniquely identifies a certain package item.
package_id	int(6)	ID link to the package name. This identifies what package does the item belongs.
item_id	int(6)	ID link to the items. This identifies what item is part of a package.

Table 10. Client

A table that contains author information.

Fieldname	Type	Description
id	int(6)	An incremental ID of the client. This uniquely identifies a certain client.
first_name	varchar(20)	The First name of the client
middle_name	varchar(20)	The Middle name of the client
last_name	varchar(20)	The Last name of the client
contact_no	varchar(20)	Contact no of the client
address	varchar(50)	Address of the client
email	varchar(20)	Email of the client
balance	float	Balance payable of the client

Table 11. Order

A table that contains order details.

Fieldname	Type	Description
id	int(6)	An incremental ID of the order. This uniquely identifies a certain order.
client_id	int(6)	ID link to the client that specifies who orders the service.
package_id	int(6)	ID link to package items that specifies the chosen package
payment_status	varchar(20)	Can be stored with full or initial payment.
tax_rate	varchar(20)	Records the tax rate of the order.
deduction	varchar(50)	Stores the amount of deductions or discounts applied to the order.
transaction_status	varchar(20)	Can be stored with Pending, For verification, and Verified
date_order	Float	The date client ordered the service
date_verified	varchar(20)	The date order is verified
code	varchar(20)	A unique system generated code to be displayed in the Invoice

Table 12. Order_items

A table that contains preferred services by the client that will be included in the order.

Fieldname	Type	Description
id	int(6)	An incremental ID of the order item. This uniquely identifies a certain order item.
item_id	int(6)	ID link to an item. This could be null if the client doesn't

		choose any services offered but preferred custom service.
order_id	int(6)	ID link to its order details
item_name	varchar(20)	A custom service that wants the client to be included in the order.
original_price	float	Amount of the company spent for the service
markup	float	Marup of the service
remarks	text	Description of the service. It could be null. This explains how the client wants the service to be done.

Table 13. Order_operating_expenses

A table that contains operating expenses of orders or the production costs

Fieldname	Type	Description
id	int(6)	An incremental ID of the particular. This uniquely identifies a certain particular.
order_id	int(6)	ID link to its order details
particular	varchar(20)	Title of the expense
amount	float	Amount of the expense

Table 14. Payments

A table that contains client payments

Fieldname	Type	Description
id	int(6)	An incremental ID of the payment. This uniquely identifies a certain payment.
cleint_id	int(6)	ID link to the client. This identifies the payment of the client
date_paid	date	Date of the payment
amount	float	Amount of the payment
beg_balance	float	Balance before the payment is deducted to client's balance
end_balance	flaot	Balance after the payment is deducted to client's balance

Table 15. Item_categories

A table that contains service item category titles

Fieldname	Type	Description
id	int(6)	An incremental ID of the category name. This uniquely identifies a certain category.
name	varchar(20)	Category name
description	text	Description of the category

Table 16. opex_items

A table that contains operating expenses names

Fieldname	Type	Description
id	int(6)	An incremental ID of the opex name. This uniquely identifies a certain opex.
particular	varchar(20)	Particular name
description	text	Description of the expense name

Table 17. company_operating_expenses

A table that contains operating expenses of the company

Fieldname	Type	Description
id	int(6)	An incremental ID of the opex name. This uniquely identifies a certain opex.
date_transact	varchar(20)	Particular name
amount	text	Description of the expense name
opex_item_id	Int(11)	ID link to the opex_item. This identifies the particular of and expense to the company
remark	text	Description of the expense. This could be null. This describes the expense.

Chapter IV

PROJECT ASSESSMENT

A. User Testing

A survey was created using the web-based application SoGoSurvey to allow end-users to share their feedback while using the MainSpring Books Smart Financial Management System. The survey link was shared to the stakeholders. There were 6 survey responses from different age groups where 67% of the respondents are 35 – 44 years old and 33% for age range of 25 – 34 years old. The survey is intendent to measure and quantify the readiness and usability of the system.

Below were the statements used on the survey where questions 1-3 is related to the participant's particulars and questions 4-13 is the standard system usability (SU) questions. Each standard SU question is presented using five-point scale from 1 to 5 where 1 means "Strongly Disagree" and 5 means "Strongly Agree".

i. Name
ii. Occupation or Role (e.g. Finance, Operations, Admin)
iii. Age
iv. I think I would like to use this tool frequently.
v. I found the system unnecessarily complex.
vi. I thought the system was easy to use.
vii. I think that I would need the support of a technical person to be able to use this system.
viii. I found the various functions in this system were well integrated.
ix. I thought there was too much inconsistency in this system.

x. I would imagine that most people would learn to use this system very quickly.

xi. I found the system very cumbersome to use.

xii. I felt very confident using the system.

xiii. I do not need to learn a lot of things before I could get going with this system.

a. Participant Profiles

ID	Name	Age Range	Occupation or Role
1	Mary Lou Demabildo	25 - 34	Operations
2	Rod Bryan	35 – 44	Operations
3	Jacky Avenido	25 – 34	Admin
4	Iris Uy	35 – 44	Finance
5	Adelio Bato-on	35 – 44	Admin
6	Harry	35 – 44	Finance

Table 1 Participant Profiles Table

b. Results

The results were interpreted using the system usability scale where we subtracted 1 from user responses to odd statements and subtracted corresponding value from 5 in the even-numbered statements. At this point the converted response scale will be ranging from 0 to 5 with five being the most positive. By adding responses from all participants and multiplying the total by 2.5 converts the range from 0-50 to 0-100.

The computed scores were then converted into letter grades for better understanding.

Score	Letter Grade	Adjective Rating
Above 80.3	A	Excellent
Between 68 and 80.3	B	Good
68	C	OK
Between 51 and 67	D	Poor
Below 51	F	Awful

Table 2 Letter Grade Table

Following are the actual survey responses to the standard system usability questions (Q4-13):

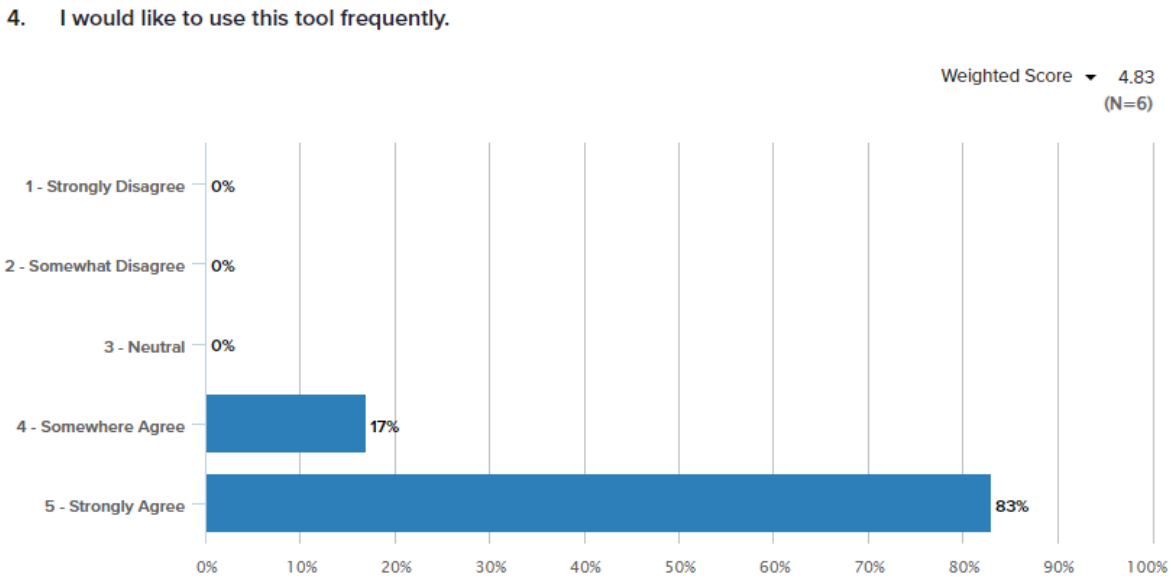


Figure 1 Responses to question 4

5. I found the system unnecessarily complex.

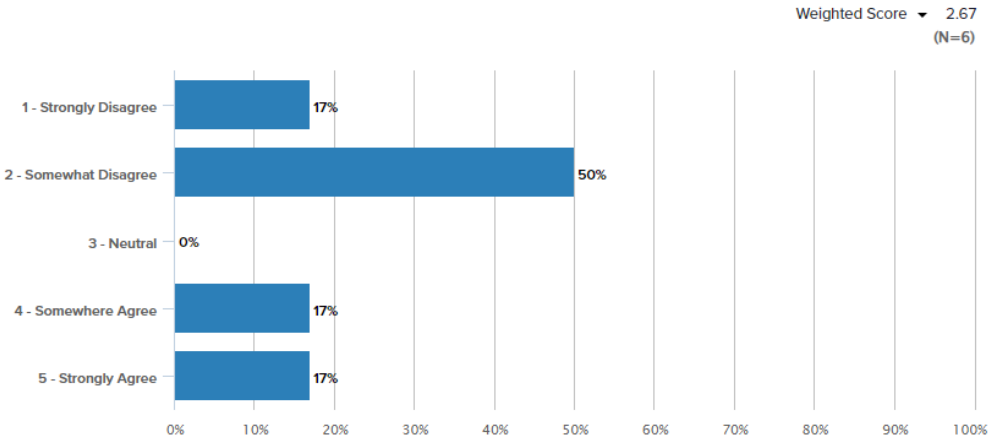


Figure 2 Responses to question 5

6. I thought the system was easy to use.

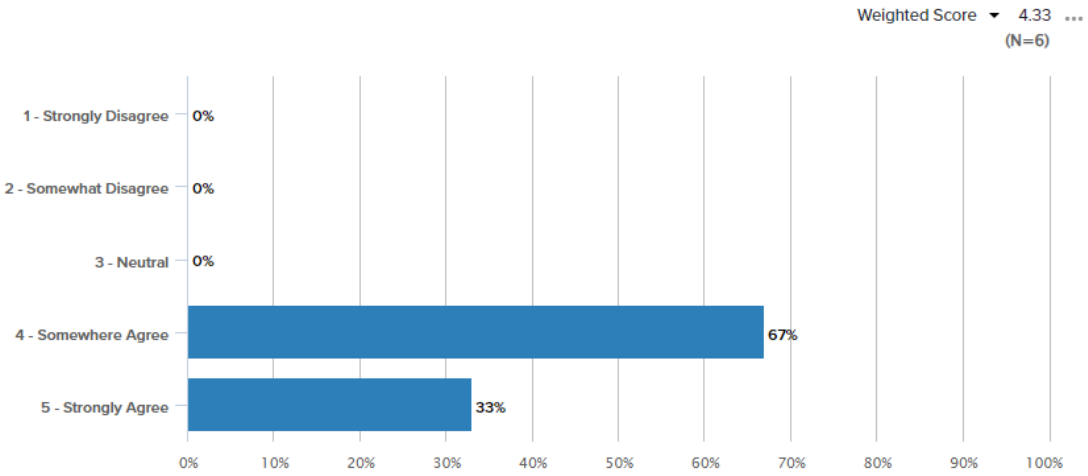


Figure 3 Responses to question 6

7. I think that I would need the support of a technical person to be able to use this system.

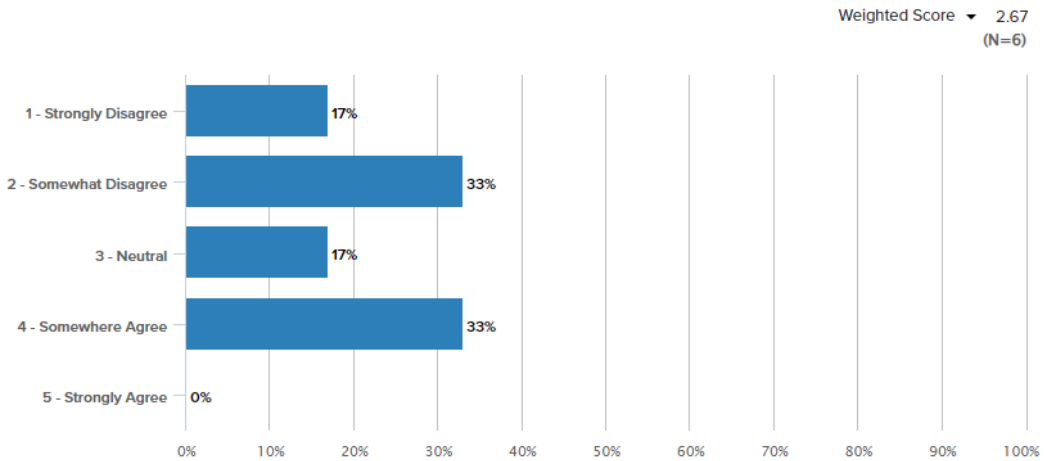


Figure 4 Responses to question 7

8. I found the various functions in this system were well integrated.

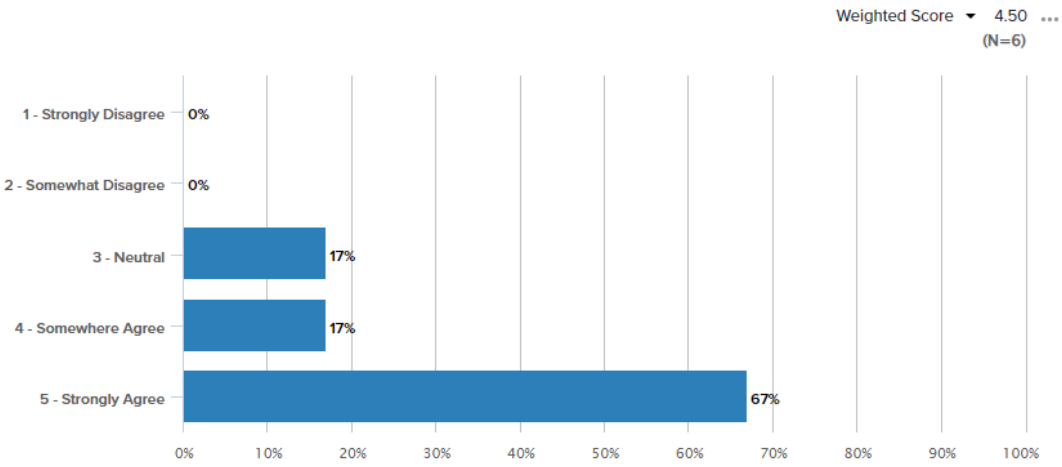


Figure 5 Responses to question 8

9. I thought there was too much inconsistency in this system.

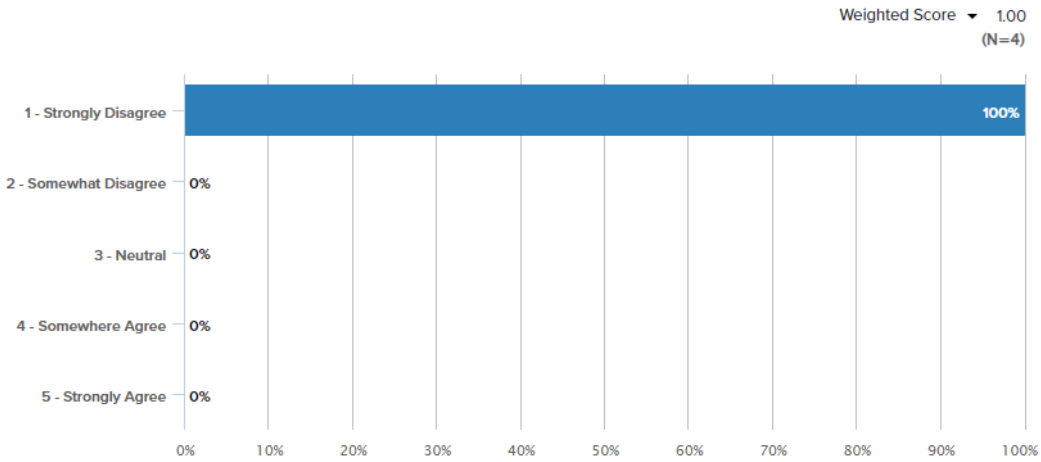


Figure 6 Responses to question 9

10. I would imagine that most people would learn to use this system very quickly.

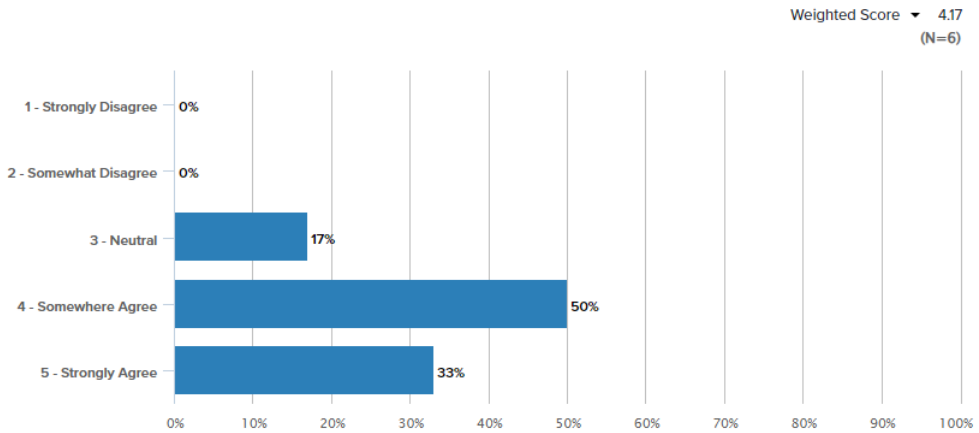


Figure 7 Responses to question 10

11. I found the system very cumbersome to use.

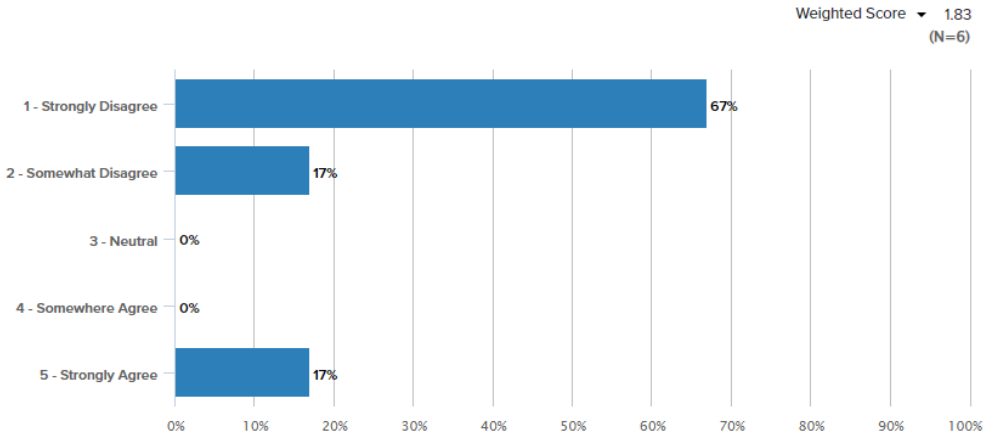


Figure 8 Responses to question 11

12. I felt very confident using the system.

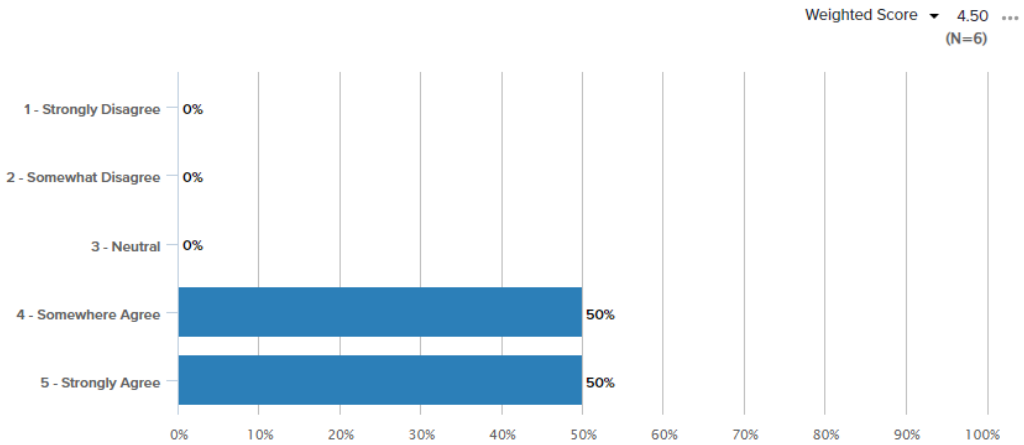


Figure 9 Responses to question 12

13. I do not need to learn a lot of things before I could get going with this system.

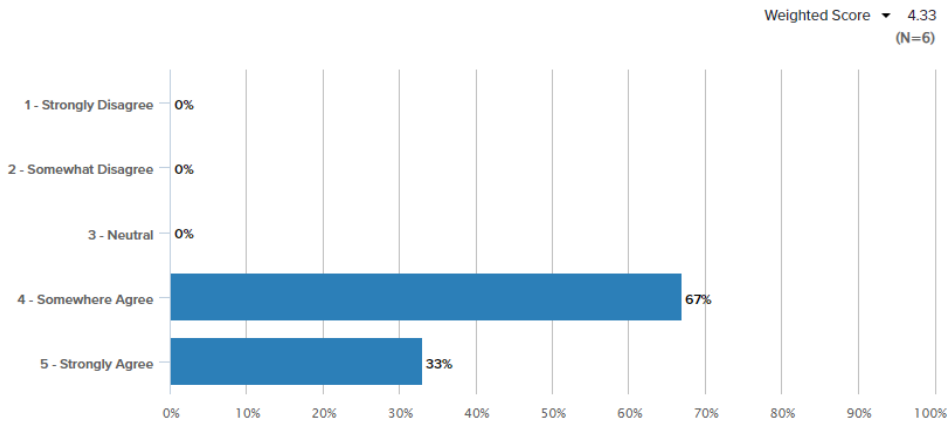


Figure 10 Responses to question 13

Based on the survey responses, the SUS score is 81.25 which means that the users found the system usable and is ready for production deployment. Although the system received a high grade, improvement is still required for future releases.

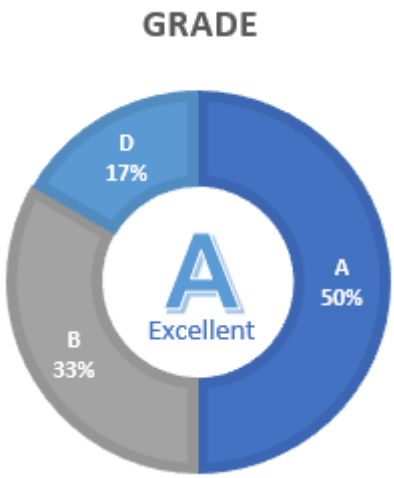


Figure 11 SUS Score

The system has been tested and evaluated by the respective stakeholders and their response indicates that the system is easy to use. Moreover, the system usability score which has resulted to 81.25 or a letter grade of A means that the system has captured the requirements needed by the client to replace their current manual system.

B. Security Testing

The system was tested using OWASP ZAP, an open-source web application security scanner. The following vulnerabilities were identified: 1 Medium Risk, 4 Low Risks, 1 Informative.

The medium risk found was X-frame-options Header not set which is a result of the temporary free domain where this system is currently being hosted for testing

purposes until the client decides to buy the official domain. The low risks found were as follows:

- Cookie No HttpOnly Flag
- Server Leaks Information via "X-Powered-by" HTTP Response Header Fields
- Timestamp Disclosure – Unix
- X-content-type-Options Header Missing

While there was an informational risk identified about Information Disclosure – Suspicious comments.

There are no critical risks identified after running the security scanner and for this reason, no action has been taken. Another security scan will be executed when the system will be hosted on their official domain.

Chapter V

DISCUSSIONS

Despite the fact that almost all transactions and routine tasks have already been developed systems, the project began with making plans on what system to develop that could benefit or improve a company or organization on their daily basis transactions. The researcher formulated a plan to create a financial system while also assisting the organization in forecasting future revenue based on their financial data. After finalizing the plan, the researcher selected a candidate company to serve as the project's host or primary foundation. Analyzing the system needs and wants of a company is not an easy task, it would take time and patients to plan an efficient way to reduce their burdens using the traditional/manual process. The duration of the web-based system development is one year. This covers the development environment setup, design of Database and UI, completion of masterdata modules as it is prerequisite in creating transactional functionalities, and it also includes the feature modules. Constant communication with the client is maintained during the development period to ensure that the system meets the requirements as planned. Since the researcher uses Agile methodology, it is simple to jump to a specific phase of development if some details need to be changed before moving on to the next phase.

Chapter VI

CONCLUSION

After undertaking the process that included company research, system analysis and design, development and user testing, the MainSpring Books Smart Financial Management System have commendable results in solving the company's problem in their manual accounting system. Having services and client management, the company can easily handle and monitor their orders and client's credit balances. The system also includes company expenses management to easily manage the company's expenses in both operation and assets. Tracker report can help track the expenses and the net income of the company. The predictive analytics tool predicts future income for the organization, which may assist them plan their strategic income target and corporate budget. The Data Mining capability of the company gives it an advantage in marketing since it gathers author data from a website and analyzes it for user readability, which may aid in marketing strategy.

Chapter VII

FUTURE WORK

To further improve the study, it would be a great development to have client-side system to allow authors to manage their own orders and inquire the company's services in an efficient way. Establishing live chat feature to have constant communication to the clients regarding their concerns on the orders without using their personal accounts or contacts.

To further establish the significance of the project, creating a mobile platform of the system would improve the user experience. Having live notifications of the order status monitoring and expenses tracking would be an additional advantage of the company's daily operation.

REFERENCES

Holly, J. (2019). *How to Set Up a Basic Bookkeeping System*
[1] https://scalefactor.com/scaleblog/set-basic-bookkeeping-system/

Newman, E. J. (1992). *Computers in small business accounting software problems*
[2] https://scholarworks.umt.edu/cgi/viewcontent.cgi?article=2899&context=etd

Vitez, O. (2017). *What Are the Characteristics of a Computerized System Accounting Environment?*
[3] https://bizfluent.com/list-6874340-characteristics-computerized-system-accounting-environment-.html

Thomason, K. (2019). *The Disadvantages of Manual Accounting*
[4] https://bizfluent.com/info-8132024-disadvantages-manual-accounting.html

Assessing the Impact of Accounting Software in the Processing of Accounting Information.
[5] https://nairaproject.com/projects/1956.html

Lewis, R. *Software Required for Databases in a Small Business.*
[6] https://smallbusiness.chron.com/software-required-databases-small-business-16832.html

Top 3 Disadvantages of Using Spreadsheets for Contract Management.
[7] https://www.contractlogix.com/contract-management/top-3-disadvantages-of-using-spreadsheets-for-contract-management/

How Predictive Forecasting Can Benefit Your Business
[8] https://saphanajourney.com/sap-analytics-cloud/learning-article/how-predictive-forecasting-can-benefit-your-business/

Twin, A. (2020). *Data Mining.*
[9] https://www.investopedia.com/terms/d/datamining.asp#:~:text=Data%20mining%20is%20a%20process,increase%20sales%20and%20decrease%20costs.

Williams, E. (2020). *Top 5 Project Accounting Software.*
[10] https://pdf.wondershare.com/accounting/project-accounting-software.html

Literature Review Of A Computerized Accounting System.
[11] https://www.ipl.org/essay/Literature-Review-Of-A-Computerized-Accounting-System-FKD75S74SCFR

Masanja, N. M. (2020). *Chapter Two Review of Related Literature 2.1*

[12] https://www.researchgate.net/publication/339021355_CHAPTER_TWO_REVIEW_OF_RELATED_LITERATURE_21

APPENDICES

I. Complete Program Listing

1. General Components

1.1 General Container

```
import React from 'react';
import { connect } from 'react-redux';
import CssBaseline from '@material-ui/core/CssBaseline';
import Typography from '@material-ui/core/Typography';
import Container from '@material-ui/core/Container';
import 'fontsource-roboto';
import { Link } from 'react-router-dom';
import { emphasize, withStyles } from '@material-ui/core/styles';
import Breadcrumbs from '@material-ui/core/Breadcrumbs';
import Chip from '@material-ui/core/Chip';
import Card from '@material-ui/core/Card';

const StyledBreadcrumb = withStyles((theme) => ({
  root: {
    backgroundColor: theme.palette.grey[100],
    height: theme.spacing(3),
    color: theme.palette.grey[800],
    fontWeight: theme.typography.fontWeightRegular,
    '&:hover, &:focus': {
      backgroundColor: theme.palette.grey[300],
    },
    '&:active': {
      boxShadow: theme.shadows[1],
      backgroundColor:      emphasize(theme.palette.grey[300],
0.12),
    },
  },
}))(Chip);

function GenCont(props) {
  const bcrp = props.breadcrumps;

  return (
    <React.Fragment>
      <CssBaseline />
      <Container maxWidth="lg">
        <Breadcrumbs aria-label="breadcrumb">
          {bcrp.map((c) => (

            <Link to={{
              pathname: c.route,
              state: c.states
            }} >
              <StyledBreadcrumb
                component="a"
                href="#"
                label={c.label}
                icon={c.icon}
              // onClick={handleClick}
              />
            </Link>

          ))}
        </Breadcrumbs>
        <br />

        {
          props.disable_card == "true" ?
            <>
              <Typography variant="h6">
                {props.title}
              </Typography>
              {props.content}
            </>

            :
            <>
              <Card style={{ padding: "10px", zIndex: 0 }}>
                <Typography variant="h6">
                  {props.title}
                </Typography>
                <div style={{zIndex: '5 !important'}}>

                  {props.content}
                </div>
              </Card>
            </>
        }
      </Container>
    </React.Fragment>
  );
}

const mapStateToProps = (state) => ({
  isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(GenCont);
```

1.2 Form Container

```
import React from 'react';
import { connect } from 'react-redux';
import 'fontsource-roboto';
import { Link } from 'react-router-dom';
import { makeStyles } from '@material-ui/core/styles';
import Paper from '@material-ui/core/Paper';
import { ReportProblemSharp } from '@material-ui/icons';


function FormCont(props) {
  const fAttr = props.formAttr[0];

  const useStyles = makeStyles((theme) => ({
    root: {
      display: 'flex',
      flexWrap: 'wrap',
      '& > *': {

        width: theme.spacing(fAttr.width),
        height: fAttr.height? theme.spacing(fAttr.height) : "auto",
      },
    },
  }));

  const classes = useStyles();

  return (
    <>
      <div className={classes.root}>
        <Paper elevation={3} >
          <div style={{margin: "3%"}}>
          {props.form}
          </div>
        </Paper>
      </div>
    </>
  );
}

const mapStateToProps = (state) => ({
  isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(FormCont);
```

1.3 Date Range

```
import React, { useState, useEffect } from 'react';
import { makeStyles } from '@material-ui/core/styles';
import TextField from '@material-ui/core/TextField';
import IconButton from '@material-ui/core/IconButton';
import Clear from '@material-ui/icons/Clear';

const useStyles = makeStyles((theme) => ({
  container: {
    display: 'flex',
    flexWrap: 'wrap',
  },
  textField: {
    marginLeft: theme.spacing(1),
    marginRight: theme.spacing(1),
    width: 200,
  },
}));
```

```
export default function DatePickers(props) {
  const classes = useStyles();
  const [state, setState] = useState({ datefrom: null, dateto: null , dateKey: 0});

  var date = new Date();
  var dateToday = date.toISOString().split('T')[0];

  function onChange(event) {//Sets the value of states
    const { name, value } = event.target;
    setState(prevState => ({ ...prevState, [name]: value }));
  }

  useEffect(() => {

    if (state.datefrom && state.dateto) {
      var dates = {
        datefrom: state.datefrom,
        dateto: state.dateto,
      }

      if(props.dates){
        props.dates(dates)
      }

    }

  }, [state]);

  function cleardate(){
    if(props.dates){
      props.dates("clear")
      setState(prevState => ({ ...prevState, datefrom: null,dateto: null, dateKey: state.dateKey +1 }))
    }
  }

  return (
    <>
      <form  className={classes.container}  noValidate key={state.dateKey}>
        <TextField
          id="datefrom"
          label="Date from"
          name="datefrom"
          type="date"
          onChange={onChange}
          inputProps={{ max: state.dateto ? state.dateto : dateToday, name: 'datefrom' }}
          defaultValue={state.datefrom}
          className={classes.textField}
          InputLabelProps={{
            shrink: true,
          }}
        />
        <TextField
          id="dateto"
          label="Date To"
          type="date"
          name="dateto"
          onChange={onChange}
          defaultValue={state.dateto}
          className={classes.textField}
          inputProps={{ max: dateToday, min: state.datefrom, name: 'dateto' }}
          InputLabelProps={{
            shrink: true,
          }}
        />

      {
      state.datefrom && state.dateto ?
        <IconButton  onClick={cleardate}  color="primary"  aria-label="upload picture" component="span">
          <Clear />
        </IconButton>
        : <></>
        }
      </form>
      <br />
    </>
  );
}
```

2. Sample Masterdata Code

2.1 View

```
import React, { useState, useEffect } from 'react';
import { connect } from 'react-redux';
import GenCont from '../../../components/gen_container';
import Fab from '@material-ui/core/Fab';
import AddIcon from '@material-ui/icons/Add';
import { Link } from 'react-router-dom';
import Tooltip from '@material-ui/core/Tooltip';
import Http from '../../../Http';
import { useSnackbar } from 'notistack';
import DeleteIcon from '@material-ui/icons/Delete';
import IconButton from '@material-ui/core/IconButton';
import EditIcon from '@material-ui/icons/Edit';
import ListAltIcon from '@material-ui/icons/ListAlt';
import ReactTable from '../components/reactTable';

var _isMounted = false;

function content() {
    _isMounted = true;
    const api = '/api/v1/items';
    const { enqueueSnackbar } = useSnackbar();

    var localTerminal = JSON.parse(localStorage.getItem("items") || "[]");
    const [state, setState] = useState({ data: localTerminal });

    useEffect(() => {
        Http.get(api)
            .then((response) => {
                localStorage.setItem('items', JSON.stringify(response.data.items))
                localTerminal = JSON.parse(localStorage.getItem("items") || "[]");
                if (_isMounted) {
                    setState({ data: localTerminal })
                }
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch items', { variant })
            });
    }, []);

    function deleteItem(key) {
        var variant;
        const itm = state.data;
        if (confirm('Are you sure you want to proceed?')) {

            Http.delete(`${api}/${key}`)
                .then((response) => {
                    if (response.status === 204) {
                        const index = itm.findIndex(
                            (item) => parseInt(item.id, 10) === parseInt(key, 10),
                        );
                        const update = [...itm.slice(0, index), ...itm.slice(index + 1)];
                        if (_isMounted) {
                            localStorage.setItem('items', JSON.stringify(update))
                            localTerminal = JSON.parse(localStorage.getItem("items") || "[]");
                            if (_isMounted) {
                                setState({ data: localTerminal })
                            }
                        }
                    }
                    variant = 'success'
                enqueueSnackbar('User successfully deleted', { variant })
                })
                .catch((error) => {
                    console.log(error);
                    variant = 'error'
                    enqueueSnackbar('Error deleting item', { variant })
                });

        }

    }
```

```
    function actionFormat(cell, row) {
        var id = row.id;
        return (
            <>
                <IconButton aria-label="delete" style={{ padding: "0" }}
onClick={() => deleteItem(id)} >
                    <DeleteIcon />
                </IconButton>
                  
                <Link to={{
                    pathname: `/form/items`,
                    state: {
                        particular: row.particular,
                        price: row.price,
                        category_id: row.category_id,
                        type: "Update",
                        id: id
                    }
                }}>
                  <IconButton aria-label="delete" style={{ padding: "0" }} >
                      <EditIcon />
                  </IconButton>
                </Link>
            </>
        );
    }

    const columns = [
        {
            dataField: 'id',
            text: ' ID',
            hidden: true
        }, {
            dataField: 'category_id',
            text: 'Particular',
            hidden: true
        }, {
            dataField: 'particular',
            text: 'Particular',
            sort: true
        }, {
            dataField: 'category',
            text: 'Category',
            sort: true
        },{
            dataField: 'price',
            text: 'Price',
            sort: true
        }, {
            text: 'Actions',
            formatter: actionFormat,
            headerStyle: (colum, colIndex) => {
                return { width: '100px', textAlign: 'center' };
            }
        }];

    return (
        <>
            <Link style={{ float: "right" }} to="/form/items">
                <Tooltip title="Create new service items" aria-
label="add">
                    <Fab size="medium" color="primary" aria-label="add">
                        <AddIcon />
                    </Fab>
                </Tooltip>
            </Link>
            <br />

            <ReactTable columns={columns}  data={state.data}/>
        </>
    )
}

function AllItems() {
    const breadcrumps = [
        {
            label: "Service Items",
            icon: <ListAltIcon fontSize='small' />,
            route: "/all/items"
        }
    ]

    return (
        <>
            <GenCont title="All Service Items"
breadcrumps={breadcrumps} content={content()} />
            <br /><br />
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllItems);
```

2.2 Controller

```
<?php

namespace App\Http\Controllers;

use App\Category;
use App\Item;
use Illuminate\Http\Request;

class ItemController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $items =      Item::join('categories',  'items.category_id',  '=',
'categories.id')->select('items.*',  'categories.name  as  category')-
>get();
        } catch (Exception $e) {
        }

        return response()->json([
            'status' => 200,
            'items' => $items,
        ], 200);
    }
}
```

3. Assignment of services to package

3.1 Controller

```
<?php

namespace App\Http\Controllers;

use App\Item;
use App\Package_item;
use Illuminate\Http\Request;

class PackageItemController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $items =  $this->getallitems(request('package_id'));
        } catch (Exception $e) {
        }
```

```
        return response()->json([
            'status' => 200,
            'items' => $items,
        ], 200);
    }

    function getallitems($package_id)
    {
        $items = Package_item::join('packages', 'package_items.package_id', '=', 'packages.id')
            ->join('items', 'package_items.item_id', '=', 'items.id')
            ->join('categories', 'items.category_id', '=', 'categories.id')->select('items.*', 'categories.name as category', 'package_items.id as packserv_id')
            ->where('package_items.package_id', '=', $package_id)
            ->get();

        return $items;
    }


    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }
        // Warning: Data isn't being fully sanitized yet.
        try {

            $check = Package_item::where('package_id', request('package_id'))->where('item_id', request('item_id'))->first();
            if ($check) {
                return response()->json([
                    'stat' => 150,
                ], 201);
            } else {
                $packs = Package_item::create([
                    'package_id' => request('package_id'),
                    'item_id' => request('item_id')
                ]);
                $items =  $this->getallitems(request('package_id'));
                return response()->json([
                    'status' => 201,
                    'id' => $packs->id,
                    'items' => $items,
                ], 201);
            }
        } catch (Exception $e) {
            return $this->responseServerError('Error creating resource.');
        }
    }

    public function addAllItems(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }
        // Warning: Data isn't being fully sanitized yet.
        try {
            Package_item::where('package_id', request('package_id'))->delete();

            $items =  Item::select('items.*')
            ->get();

            foreach($items as $i){

                Package_item::create([
                    'package_id' => request('package_id'),
                    'item_id' => $i['id']
                ]);

            }

        } catch (Exception $e) {

            return $this->responseServerError('Error creating resource.');
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  \App\Package_item  $package_item
     * @return \Illuminate\Http\Response
     */
    public function destroy(Request $request, $id)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $item = Package_item::where('id', $id)->firstOrFail();
            $item->delete();

            return response()->json([
                'status' => 204,
                'message' => "Deleted successfully"
            ], 204);
        } catch (Exception $e) {
            return $this->responseServerError('Error deleting resource.');
        }
    }

    public function removeItm(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $item = Package_item::where('id', request('id'))->firstOrFail();
            $item->delete();

            $items =  $this->getallitems(request('package_id'));

            return response()->json([

                'status' => 200,
                'message' => "Deleted successfully",
                'items' => $items,
            ], 200);
        } catch (Exception $e) {
            return $this->responseServerError('Error deleting resource.');
        }
    }
}
```

3.2 View

```
import React, { useState, useEffect } from 'react';
import { useLocation } from 'react-router-dom'
import { connect } from 'react-redux';
import GenCont from '../../../components/gen_container';
import PackItemForm from '../../../pages/forms/package_item_form';
import Fab from '@material-ui/core/Fab';
import AddIcon from '@material-ui/icons/Add';
import Tooltip from '@material-ui/core/Tooltip';
import Http from '../../../Http';

import BootstrapTable from 'react-bootstrap-table-next';
import 'react-bootstrap-table-next/dist/react-bootstrap-table2.min.css';
import paginationFactory from 'react-bootstrap-table2-paginator';
import 'react-bootstrap-table2-paginator/dist/react-bootstrap-table2-paginator.min.css';
import ToolkitProvider, { Search } from 'react-bootstrap-table2-toolkit';
import 'react-bootstrap-table2-toolkit/dist/react-bootstrap-table2-toolkit.min.css';

import { useSnackbar } from 'notistack';
```

```
import DeleteIcon from '@material-ui/icons/Delete';
import IconButton from '@material-ui/core/IconButton';
import ListAltIcon from '@material-ui/icons/ListAlt';
import Dialog from '@material-ui/core/Dialog';
import DialogTitle from '@material-ui/core/DialogTitle';

const { SearchBar, ClearSearchButton } = Search;
var _isMounted = false;

function content() {
   const location = useLocation();
   const { id } = location.state;
   _isMounted = true;
   const api = '/api/v1/packitms/index';
   const { enqueueSnackbar } = useSnackbar();

   const [dataitems, setData] = useState([]);

   useEffect(() => {
      Http.post(api, { package_id: id })
         .then((response) => {

            if (_isMounted) {
               setData(response.data.items);
            }
         })
         .catch((error) => {
            var variant = 'error';
            console.log(error);
            enqueueSnackbar('Failed to fetch items', {
variant })
         });
   }, []);

   function deleteItem(key) {
      const { id } = location.state; // package id
      var variant;
      _isMounted = true;

      if (confirm('Are you sure you want to proceed?')) {

         Http.post(`/api/v1/packitms/remove`,
{package_id: id, id: key})
            .then((response) => {
                  if (_isMounted) {
                     // setState(prevState => ({
...prevState, data: itm.filter(item => item.packserv_id !==
key) }));
                        setData(response.data.items);
                  }
               variant = 'success'
               enqueueSnackbar('Sevice successfully
removed', { variant })
            })
            .catch((error) => {
               console.log(error);
               variant = 'error'
               enqueueSnackbar('Error deleting item', {
variant })
            });

      }

   }

   function actionFormat(cell, row) {
      var piid = row.packserv_id;
      return (
         <>
            <IconButton aria-label="delete" style={{
padding: "0" }} onClick={() => deleteItem(piid)} >
               <DeleteIcon />
            </IconButton>

         </>
      );
   }

   function addNew(data) {
      const { id } = location.state;


      Http.post(api, { package_id: id })
      .then((response) => {

         if (_isMounted) {
            setData(response.data.items);
         }
      })
      .catch((error) => {
         var variant = 'error';
         console.log(error);
         enqueueSnackbar('Failed to fetch items', {
variant })
      });


   }

   function AddService(props) {
      const { id } = location.state;
      return (
         <>
            <DialogTitle id="simple-dialog-title">Add
service to this package</DialogTitle>
            <div style={{ padding: "10px" }}>
               <PackItemForm newservice={addNew}
pack_id={id} />
            </div>

         </>
      );
   }


   const [open, setOpen] = React.useState(false);
   const handleClickOpen = () => {
      setOpen(true);
   };

   const handleClose = () => {
      setOpen(false);

   };

const columns = [
   {
      dataField: 'packserv_id',
      text: 'pi',
      hidden: true
   },{
      dataField: 'id',
      text: ' ID',
      hidden: true
   }, {
      dataField: 'category_id',
      text: 'Particular',
      hidden: true
   }, {
      dataField: 'particular',
      text: 'Particular'
   }, {
      dataField: 'category',
      text: 'Category'
   }, {
      dataField: 'price',
      text: 'Price'
   }, {

      formatter: actionFormat,
      headerStyle: (colum, colIndex) => {
         return { width: '50px', textAlign: 'center' };
      }
   }];


   return (
      <>

         <div style={{ float: "right" }}>
            <Tooltip title="Create new package items"
aria-label="add">
               <Fab size="medium" color="primary" aria-
label="add" onClick={handleClickOpen}>
                  <AddIcon />
               </Fab>
            </Tooltip>
         </div>
         <Dialog aria-labelledby="simple-dialog-title"
open={open} onClose={handleClose}>
            <AddService />
         </Dialog>
         <br />
         <ToolkitProvider
```

```
                keyField="id"
                data={dataitems}
                columns={columns}
                search
            >
                {
                    tab => (
                        <>
                            <div style={{ float: "left" }}>

                                <div className="inline_block">
                                    <SearchBar {...tab.searchProps}
/>
                                </div>
                                <div className="inline_block">
                                    <ClearSearchButton
{...tab.searchProps} />
                                </div>

                            </div>
                            <BootstrapTable
                                {...tab.baseProps}
                                pagination={paginationFactory()}
                            />
                        </>
                    )
                }
            </ToolkitProvider>

        </>
    )
}

function AllItems() {
    const location = useLocation();
    const { name } = location.state;
    const breadcrumps = [
        {
            label: "Packages",
            icon: <ListAltIcon fontSize='small' />,
            route: "/all/packages"
        },
        {
            label: "Package Services",
            icon: <ListAltIcon fontSize='small' />
        }
    ]

    return (
        <>
            <GenCont            title={"All            "            +
name.charAt(0).toUpperCase()   +   name.slice(1)   +   "
Services"}                      breadcrumps={breadcrumps}
content={content()} />
            <br /><br />
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllItems);
```

4. Tax percentage update

4.1 View

```
import React, { useState, useEffect } from 'react';
import { makeStyles } from '@material-ui/core/styles';
import { connect } from 'react-redux';
import GenCont from '../../../components/gen_container';
import Http from '../../../Http';

import { useSnackbar } from 'notistack';
import CardActions from '@material-ui/core/CardActions';
import Button from '@material-ui/core/Button';
import Typography from '@material-ui/core/Typography';

import InputAdornment from '@material-ui/core/InputAdornment';

import SaveIcon from '@material-ui/icons/Save';

var _isMounted = false;

const useStyles = makeStyles({
    root: {
        maxWidth: 345,
    },
    media: {
        height: 150,
    },
})

function content() {
    const classes = useStyles();

    _isMounted = true;
    const api = '/api/v1/tax';
    const { enqueueSnackbar } = useSnackbar();

    var  localTerminal  =  JSON.parse(localStorage.getItem("tax")  ||
"[]");
    const [state, setState] = useState({ tax: localTerminal, id: null });
    const [rate, setRate] = useState(null);

    useEffect(() => {
        Http.get(api)
            .then((response) => {
                localStorage.setItem('tax', response.data.tax)
                localTerminal  =  JSON.parse(localStorage.getItem("tax")
|| "[]");
                if (_isMounted) {
                    setState({ tax: localTerminal, id: response.data.id })
                }
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch items', { variant })
            });
    }, []);


    const [openUp, setOpenUp] = React.useState(false);

    const handleClickUp = () => {
        setOpenUp(!openUp);
    };

    function onChange(event) {//Sets the value of states
        const { name, value } = event.target;
        if (_isMounted) {
            setRate(value);
        }
    }

    function submit(e) {
        e.preventDefault();

        if (confirm("Are you sure you want to update tax?")) {
            var variant;
            Http.patch(`${api}/${state.id}`, { rate: rate })
                .then(({ data }) => {
                    if (_isMounted) {
                        setState(prevState => ({ ...prevState, tax: rate }));
                        localStorage.setItem('tax', rate)
                    }

                    variant = 'success';
                    enqueueSnackbar('Tax successfully updated', { variant
})
                })
                .catch(() => {
                    variant = 'error';
                    enqueueSnackbar('Error updating tax', { variant })
                });
        }
    }

    return (
        <>
            <Card className={classes.root}>
                <CardActionArea>
                    <CardMedia
                        className={classes.media}
                        image="/images/tax.jpg"
                        title="Contemplative Reptile"
                    />
```

```
                <CardContent>
                  <Typography gutterBottom variant="h5"
component="h2">
                    {state.tax}% tax rate
                  </Typography>
                  <Typography variant="body2"
color="textSecondary" component="p">
                    This rate will be included in all order calculations.
                  </Typography>
                </CardContent>
              </CardActionArea>
              <CardActions>
                <Button onClick={handleClickUp} size="small"
color="primary">
                  Update
                </Button>

              </CardActions>
            </Card>

            <br />

            {
              openUp ?
                <>
                  <form onSubmit={submit} autoComplete="off">

                    <Card className={classes.root} style={{
padding: "10px" }}>
                      <FormControl variant="outlined" style={{ width:
"100%" }}>
                        <OutlinedInput
                          id="outlined-adornment-weight"

                          onChange={onChange}
                          endAdornment={<InputAdornment
position="end">%</InputAdornment>}
                          aria-describedby="outlined-weight-helper-
text"
                          inputProps={{
                            'aria-label': 'tax',
                          }}
                          labelWidth={0}
                          type="number"
                        />
                        <FormHelperText id="outlined-weight-
helper-text">Tax</FormHelperText>
                      </FormControl>

                      <Button
                        style={{ float: "right" }}
                        variant="contained"
                        color="primary"
                        size="small"
                        startIcon={<SaveIcon />}
                        type="submit"
                      >
                        Save
                      </Button>
                    </Card>
                  </form>
                </>
                :
                <></>
            }
          </>
    )
}

function tax() {
  const breadcrumps = [
    {
      label: "Tax",
      icon: <AccountBalanceWalletIcon fontSize='small' />,
      route: "/tax"
    }
  ]

  return (
    <>
      <GenCont disable_card="true" title=" "
breadcrumps={breadcrumps} content={content()} />
      <br /><br />
    </>
  );
}
```

```
const mapStateToProps = (state) => ({
  isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(tax);
```

4.2 Controller

```
<?php

namespace App\Http\Controllers;

use App\Tax;
use Illuminate\Http\Request;

class TaxController extends Controller
{
  /**
   * Display a listing of the resource.
   *
   * @return \Illuminate\Http\Response
   */
  public function index(Request $request)
  {
    // Get user from $request token.
    if (!$user = auth()->setRequest($request)->user()) {
      return $this->responseUnauthorized();
    }

    try {
      $tax =  Tax::select('taxes.*')->first();
    } catch (Exception $e) {
    }

    return response()->json([
      'status' => 200,
      'tax' => $tax->rate,
      'id' => $tax->id,
    ], 200);
  }
  /**
   * Update the specified resource in storage.
   *
   * @param  \Illuminate\Http\Request  $request
   * @param  \App\Tax  $tax
   * @return \Illuminate\Http\Response
   */
  public function update(Request $request, $id)
  {
    // Get user from $request token.
    if (!$user = auth()->setRequest($request)->user()) {
      return $this->responseUnauthorized();
    }

    try {
      $tax = Tax::where('id', $id)->firstOrFail();

      if (request('rate')) {
        $tax->rate = request('rate');
      }
      $tax->save();

      return response()->json([
        'status' => 200,
      ], 200);
    } catch (Exception $e) {
      return $this->responseServerError('Error updating
resource.');
    }
  }


}
```

5. Onetime Expenses

5.1 View

```
import React, { useState, useEffect } from 'react';
import { connect } from 'react-redux';
import GenCont from '../../../components/gen_container';
import Fab from '@material-ui/core/Fab';
import AddIcon from '@material-ui/icons/Add';
import { Link } from 'react-router-dom';
import Tooltip from '@material-ui/core/Tooltip';
import Http from '../../../Http';
import Button from '@material-ui/core/Button';

import BootstrapTable from 'react-bootstrap-table-next';
import 'react-bootstrap-table-next/dist/react-bootstrap-table2.min.css';
import { Search, CSVExport } from 'react-bootstrap-table2-toolkit';
import 'react-bootstrap-table2-toolkit/dist/react-bootstrap-table2-toolkit.min.css';

import { useSnackbar } from 'notistack';
import DeleteIcon from '@material-ui/icons/Delete';
import IconButton from '@material-ui/core/IconButton';
import EditIcon from '@material-ui/icons/Edit';
import ListAltIcon from '@material-ui/icons/ListAlt';
import ReactTable from '../components/reactTable';

import DateRangeGen from '../../../components/daterangeGen';
import { useReactToPrint } from 'react-to-print';
import { ComponentToPrint } from '../components/printTemplate';

var _isMounted = false;

function content() {

    const componentRef = React.useRef();
    const handlePrint = useReactToPrint({
        content: () => componentRef.current,
    });

    _isMounted = true;
    const api = '/api/v1/ote';
    const { enqueueSnackbar } = useSnackbar();

    var localTerminal = JSON.parse(localStorage.getItem("ote") || "[]");
    const [state, setState] = useState({ data: localTerminal });

    useEffect(() => {
        getData()
    }, []);

    function getData(post = {}){

        Http.post('/api/v1/ote/tracker', post)
        .then((response) => {
            localStorage.setItem('ote', JSON.stringify(response.data.ote))
            localTerminal = JSON.parse(localStorage.getItem("ote") || "[]");
            if (_isMounted) {

                setState({ data: localTerminal })
            }
        })
        .catch((error) => {
            var variant = 'error';
            console.log(error);
            enqueueSnackbar('Failed to fetch items', { variant })
        });
    }

    function deleteItem(key) {
        var variant;
        const itm = state.data;
        if (confirm('Are you sure you want to proceed?')) {

            Http.delete(`${api}/${key}`)
                .then((response) => {
                    if (response.status === 204) {
                        const index = itm.findIndex(
                        (item) => parseInt(item.id, 10) === parseInt(key, 10),
                        );
                        const update = [...itm.slice(0, index), ...itm.slice(index + 1)];
                        if (_isMounted) {
                            localStorage.setItem('ote', JSON.stringify(update))
                            localTerminal = JSON.parse(localStorage.getItem("ote") || "[]");
                            if (_isMounted) {
                                setState({ data: localTerminal })
                            }
                        }
                    }
                    variant = 'success'
                    enqueueSnackbar('OTE successfully deleted', { variant })
                })
                .catch((error) => {
                    console.log(error);
                    variant = 'error'
                    enqueueSnackbar('Error deleting item', { variant })
                });

        }

    }

    function getdates(data) {

        if (data == "clear") {
            getData()
        } else {
            console.log("get dates")
            console.log(data)

            getData({ from_date: data.datefrom, to_date: data.dateto })
        }

    }

    function actionFormat(cell, row) {
        var id = row.id;
        return (
            <>
                <IconButton aria-label="delete" style={{ padding: "0" }} onClick={() => deleteItem(id)} >
                    <DeleteIcon />
                </IconButton>
                  
                <Link to={{
                    pathname: `/form/ote`,
                    state: {
                        particular: row.particular,
                        amount: row.amount,
                        desc: row.description,
                        date: row.expense_date,
                        type: "Update",
                        id: id
                    }
                }}>
                  <IconButton aria-label="delete" style={{ padding: "0" }} >
                      <EditIcon />
                  </IconButton>
                </Link>
            </>
        );
    }

    const columns = [
        {
            dataField: 'id',
            text: ' ID',
            hidden: true
        }, {
            dataField: 'expense_date',
            text: 'Date',
            sort: true
        }, {
            dataField: 'particular',
            text: 'Particular',
            sort: true
        },{
            dataField: 'amount',
            text: 'Amount',
            sort: true
        },{
            dataField: 'description',
            text: 'Description',
            sort: true
        }, {
            text: 'Actions',
```

```
        formatter: actionFormat,
        headerStyle: (colum, colIndex) => {
          return { width: '100px', textAlign: 'center' };
        },
        hidden : false

    }];

    function printContent(){
        const newcolumns = [
            {
                dataField: 'id',
                text: ' ID',
            }, {
                dataField: 'expense_date',
                text: 'Date'
            }, {
                dataField: 'particular',
                text: 'Particular'
            },{
                dataField: 'amount',
                text: 'Amount',
            },{
                dataField: 'description',
                text: 'Description',
            }];

        return(
            <>
            <BootstrapTable keyField="id" data={state.data}
columns={newcolumns} />
            </>
        )
    }

    var printdata = {
        title: 'One time expenses',
        content: printContent()
    }

  return (
    <>
        <Link style={{ float: "right" }} to="/form/ote">
            <Tooltip title="Create new one time expenses" aria-
label="add">
                <Fab size="medium" color="primary" aria-label="add">
                    <AddIcon />
                </Fab>
            </Tooltip>
        </Link>
        <Button style={{ float: 'left' }} variant="contained"
onClick={handlePrint} color="primary">Print</Button>
        <div style={{ display: "none" }}><ComponentToPrint
data={printdata} ref={componentRef} /></div>
        <br />
        <br />
        <br />

        <DateRangeGen dates={getdates} />
        <ReactTable columns={columns}  data={state.data}/>
    </>
  )
}

function Allitems() {
  const breadcrumps = [
    {
      label: "One time expenses",
      icon: <ListAltIcon fontSize='small' />,
      route: "/all/items"
    }
  ]

  return (
    <>
      <GenCont title="All one time expenses"
breadcrumps={breadcrumps} content={content()} />
      <br /><br />
    </>
  );
}

const mapStateToProps = (state) => ({
  isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(Allitems);
```

5.2 Form

```
import React, { useState } from 'react';
import { useLocation } from 'react-router-dom'
import { connect } from 'react-redux';
import GenCont from '../../components/gen_container';
import FormCont from '../../components/form_container';
import LaptopWindowsIcon from '@material-
ui/icons/LaptopWindows';
import AddIcon from '@material-ui/icons/Add';
import TextField from '@material-ui/core/TextField';
import SaveIcon from '@material-ui/icons/Save';
import Button from '@material-ui/core/Button';
import EditIcon from '@material-ui/icons/Edit';
import Http from '../../Http';
import { useSnackbar } from 'notistack';
import ListAltIcon from '@material-ui/icons/ListAlt';

var _isMounted = false;


function form() {
  _isMounted = true;
  const api = '/api/v1/ote';
  const { enqueueSnackbar } = useSnackbar();
  const location = useLocation();
  var particular = "";
  var amount = "";
  var desc = "";
  var date = "";
  var variant;
  var id;

  if (location.state) {
    particular = location.state.particular;
    amount = location.state.amount;
    desc = location.state.desc;
    date = location.state.date;
    id = location.state.id;
  }

  const [state, setState] = useState({
    particular: particular,
    amount: amount,
    desc: desc,
    date: date
  });

  function onChange(event) {//Sets the value of states
    const { name, value } = event.target;
    if (_isMounted) {
      setState(prevState => ({ ...prevState, [name]: value }));
    }
  }

  function submit(e) {
    e.preventDefault();
    const posts = {
      particular: state.particular,
      amount: state.amount,
      description: state.desc,
      expense_date: state.date
    }

    location.state ? submitUpdate(posts) : submitCreate(posts);

  }

  function submitCreate(posts) {

    Http.post(api, posts)
      .then(({ data }) => {
        variant = 'success';
        enqueueSnackbar('OTE successfully created', { variant })
        if (_isMounted) {
          setState({ particular: "", amount: "", desc: "", date: "" });
        }
      })
      .catch(() => {
        variant = 'error';
        console.log(error);
        enqueueSnackbar('Failed to create OTE', { variant })
      });
  }

  function submitUpdate(posts) {
```

```
        Http.patch(`${api}/${id}`, posts)
            .then(({ data }) => {
                variant = 'success';
                enqueueSnackbar('OTE successfully updated', { variant
})
            })
            .catch(() => {
                variant = 'error';
                enqueueSnackbar('Error updating OTE', { variant })
            });
    }


    return (
        <>
            <form onSubmit={submit} autoComplete="off">
                <TextField id="outlined-basic" required
value={state.particular} name="particular" onChange={onChange}
style={{ width: "100%" }} label="Particular" variant="outlined" /><br
/><br />
                <TextField id="outlined-basic" type="number" required
value={state.amount} name="amount" onChange={onChange}
style={{ width: "100%" }} label="Amount" variant="outlined" /><br
/><br />
                <TextField id="date" InputLabelProps={{
                    shrink: true,
                }}
                    type="date" required value={state.date} name="date"
onChange={onChange} style={{ width: "100%" }} label="Date"
variant="outlined" /><br /><br />
                <TextField

                    value={state.desc} name="desc"
onChange={onChange}
                    style={{ width: "100%" }}
                    id="outlined-multiline-static"
                    label="Description"
                    multiline
                    rows={4}
                    variant="outlined"
                /><br /><br />

                <Button
                    style={{ float: "right" }}
                    variant="contained"
                    color="primary"
                    size="medium"
                    startIcon={<SaveIcon />}

                    type="submit"
                // onClick={submit}
                >
                    Save
                </Button>
                <br /><br />
            </form>
        </>
    );

}


function content() {
    const formClasses = [{
        width: 50,
    }]
    return (
        <>
            <br />

            <FormCont
                formAttr={formClasses}
                form={form()}
            />

        </>
    );
}


function CategoryForm() {
    const location = useLocation();

    var label = "Create New";
    var icon = <AddIcon fontSize='small' />;
    var title = "Create New OTE";

    if (location.state) {
        label = "Update";
        icon = <EditIcon fontSize='small' />;
        title = "Update OTE";

    }

    const breadcrumps = [
        {
            label: "One time expenses",
            icon: <ListAltIcon fontSize='small' />,
            route: "/all/ote"
        },
        {
            label: label,
            icon: icon,
        }
    ]

    return (
        <>
            <GenCont disable_card="true" title={title}
breadcrumps={breadcrumps} content={content()} />

        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(CategoryForm);
```

5.3 Controller

```
<?php
namespace App\Http\Controllers;
use App\One_time_expense;
use Illuminate\Http\Request;

class OneTimeExpenseController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $ote = One_time_expense::select('one_time_expenses.*');
            if (request('from_date') && request('to_date')) {
                $ote->whereDate('one_time_expenses.expense_date',
'>=', request('from_date'));
                $ote->whereDate('one_time_expenses.expense_date',
'<=', request('to_date'));
            }
            $ote = $ote->get();
        } catch (Exception $e) {
        }

        return response()->json([
            'status' => 200,
            'ote' => $ote,
        ], 200);
    }


    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }
```

```
        // Warning: Data isn't being fully sanitized yet.
        try {
            $ote = One_time_expense::create([
                'particular' => request('particular'),
                'expense_date' => request('expense_date'),
                'description' => request('description'),
                'amount' => request('amount')
            ]);
            return response()->json([
                'status' => 201,
                'id' => $ote->id
            ], 201);
        } catch (Exception $e) {
            return $this->responseServerError('Error creating
resource.');
        }
    }


    /**
     * Show the form for editing the specified resource.
     *
     * @param  \App\One_time_expense  $one_time_expense
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $ote = One_time_expense::where('id', $id)->firstOrFail();

            if (request('particular')) {
                $ote->particular = request('particular');
            }

            if (request('description')) {
                $ote->description = request('description');
            }

            if (request('expense_date')) {
                $ote->expense_date = request('expense_date');
            }

            if (request('amount')) {
                $ote->amount = request('amount');
            }

            $ote->save();

            return response()->json([
                'status' => 200,
            ], 200);
        } catch (Exception $e) {
            return $this->responseServerError('Error updating
resource.');
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  \App\One_time_expense  $one_time_expense
     * @return \Illuminate\Http\Response
     */
    public function destroy(Request $request, $id)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $ote = One_time_expense::where('id', $id)->firstOrFail();
            $ote->delete();

            return response()->json([
                'status' => 204,
                'message' => "Deleted successfully"
            ], 204);
        } catch (Exception $e) {
            return $this->responseServerError('Error deleting
resource.');
        }
    }
}
```

6. Honorarium

6.1 View

```
import React, { useState, useEffect } from 'react';
import { connect } from 'react-redux';
import GenCont from '../../../components/gen_container';
import Fab from '@material-ui/core/Fab';
import AddIcon from '@material-ui/icons/Add';
import { Link } from 'react-router-dom';
import Tooltip from '@material-ui/core/Tooltip';
import Http from '../../../Http';
import BootstrapTable from 'react-bootstrap-table-next';
import 'react-bootstrap-table-next/dist/react-bootstrap-
table2.min.css';
import { useSnackbar } from 'notistack';
import DeleteIcon from '@material-ui/icons/Delete';
import IconButton from '@material-ui/core/IconButton';
import EditIcon from '@material-ui/icons/Edit';
import ListAltIcon from '@material-ui/icons/ListAlt';
import Button from '@material-ui/core/Button';
import ReactTable from '../components/reactTable';
import DateRangeGen from '../../../components/daterangeGen';
import { useReactToPrint } from 'react-to-print';
import { ComponentToPrint } from '../components/printTemplate';

var _isMounted = false;

function content() {
    _isMounted = true;
    const { enqueueSnackbar } = useSnackbar();

    const componentRef = React.useRef();
    const handlePrint = useReactToPrint({
        content: () => componentRef.current,
    });

    var localTerminal =
JSON.parse(localStorage.getItem("honorarium") || "[]");
    const [state, setState] = useState({ data: localTerminal });

    useEffect(() => {
        getData();
    }, []);


    function getData(post = {}){
        Http.post('/api/v1/honorarium/tracker', post)
        .then((response) => {
            localStorage.setItem('honorarium',
JSON.stringify(response.data.honoraria))
            localTerminal =
JSON.parse(localStorage.getItem("honorarium") || "[]");
            if (_isMounted) {
                setState({ data: localTerminal })
            }
        })
        .catch((error) => {
            var variant = 'error';
            console.log(error);
            enqueueSnackbar('Failed to fetch packages', { variant })
        });
    }

    function deleteItem(key) {
        var variant;

        if (confirm('Are you sure you want to proceed?')) {

            Http.post('/api/v1/honorarium/delete', {id: key})
                .then((response) => {

                    if (_isMounted) {
                        localStorage.setItem('honorarium',
JSON.stringify(response.data.honoraria))
                        localTerminal =
JSON.parse(localStorage.getItem("honorarium") || "[]");
                        if (_isMounted) {

                            setState({ data: localTerminal })
```

```
                    }
                }

                variant = 'success'
    enqueueSnackbar('Honorarium successfully deleted', { variant })
            })
            .catch((error) => {
                console.log(error);
                variant = 'error'
enqueueSnackbar('Error deleting Honorarium', { variant })
            });

        }

    }

    function actionFormat(cell, row) {
        var id = row.id;
        return (
            <>
                <IconButton aria-label="delete" style={{ padding: "0" }}
onClick={() => deleteItem(id)} >
                    <DeleteIcon />
                </IconButton>
                  
                <Link to={{
                    pathname: `/form/honorarium`,
                    state: {
                        date: row.date_paid,
                        amount: row.amount,
                        name: row.staff,
                        type: "Update",
                        id: id
                    }
                }}>
                  <IconButton aria-label="delete" style={{ padding: "0" }} >
                      <EditIcon />
                  </IconButton>
                </Link>
            </>
        );
    }

    function getdates(data) {

        if (data == "clear") {
            getData()
        } else {
            console.log("get dates")
            console.log(data)

            getData({ from_date: data.datefrom, to_date: data.dateto })
        }

    }

    const columns = [
        {
            dataField: 'id',
            text: ' ID',
            hidden: true
        },{
            dataField: 'date_paid',
            text: 'Date',
            sort:true
        },{
            dataField: 'staff',
            text: 'Staff',
            sort:true
        },{
            dataField: 'amount',
            text: 'Amount',
            sort:true
        },{
            text: 'Actions',
            formatter: actionFormat,
            headerStyle: (colum, colIndex) => {
                return { width: '100px', textAlign: 'center' };
            }
        }

    ];

    function printContent(){
        const newcolumns = [
            {
                dataField: 'id',
                text: ' ID',
                hidden: true
            },{
                dataField: 'date_paid',
                text: 'Date'
            },{
                dataField: 'staff',
                text: 'Staff'
            },{
                dataField: 'amount',
                text: 'Amount'
            }
        ];

        return(
            <><BootstrapTable keyField="id" data={state.data}
columns={newcolumns} /></>
        )
    }

    var printdata = {
        title: 'Honoriarium',
        content: printContent()
    }

    return (
        <>
            <Link style={{ float: "right" }} to="/form/honorarium/add">
                <Tooltip title="Create new honorarium" aria-label="add">
                    <Fab size="medium" color="primary" aria-label="add">
                        <AddIcon />
                    </Fab>
                </Tooltip>
            </Link>
            <Button style={{ float: 'left' }} variant="contained"
onClick={handlePrint} color="primary">Print</Button>
            <div style={{ display: "none" }}><ComponentToPrint
data={printdata} ref={componentRef} /></div>
            <br /> <br /> <br />
            <DateRangeGen dates={getdates} />
            <ReactTable columns={columns}  data={state.data}/>
        </>
    )
}

function AllPackages() {
    const breadcrumps = [
        {
            label: "Honoraria",
            icon: <ListAltIcon fontSize='small' />,
            route: "/all/packages"
        }
    ]

    return (
        <>
            <GenCont title="Honoraria" breadcrumps={breadcrumps}
content={content()} />
            <br /><br />
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllPackages);
```

6.2 Form

```
import React, { useState } from 'react';
import { useLocation } from 'react-router-dom'
import { connect } from 'react-redux';
import GenCont from '../../components/gen_container';
import FormCont from '../../components/form_container';
import TextField from '@material-ui/core/TextField';
import SaveIcon from '@material-ui/icons/Save';
import Button from '@material-ui/core/Button';
import EditIcon from '@material-ui/icons/Edit';
import Http from '../../Http';
import { useSnackbar } from 'notistack';
import ListAltIcon from '@material-ui/icons/ListAlt';

var _isMounted = false;
```

```
function form() {
    _isMounted = true;
    const api = '/api/v1/honorarium';
    const { enqueueSnackbar } = useSnackbar();
    const location = useLocation();
    var date = "";
    var amount = "";
    var variant;
    var id;

    if (location.state) {
        date = location.state.date;
        amount = location.state.amount;
        id = location.state.id;
    }

    const [state, setState] = useState({ date: date, amount: amount
});

    function onChange(event) {//Sets the value of states
        const { name, value } = event.target;
        if (_isMounted) {
            setState(prevState => ({ ...prevState, [name]: value }));
        }
    }

    function submit(e) {
        e.preventDefault();
        const posts = {
            date: state.date,
            amount: state.amount,
        }
        submitUpdate(posts)
    }

    function submitUpdate(posts) {
        Http.patch(`${api}/${id}`, posts)
            .then(({ data }) => {
                variant = 'success';
                enqueueSnackbar('Honorarium successfully updated', {
variant })
            })
            .catch(() => {
                variant = 'error';
                enqueueSnackbar('Error updating Honorarium', { variant
})
            });
    }

    return (
        <>
            <form onSubmit={submit} autoComplete="off">

                <TextField id="outlined-basic" type="date" required
value={state.date} name="date" onChange={onChange} style={{
width: "100%" }} label="Honorarium Date" variant="outlined" /><br
/><br />
                <TextField id="outlined-basic" required
value={state.amount} name="amount" onChange={onChange}
style={{ width: "100%" }} label="Amount" variant="outlined" /><br
/><br />

                <Button
                    style={{ float: "right" }}
                    variant="contained"
                    color="primary"
                    size="medium"
                    startIcon={<SaveIcon />}
                    type="submit"
                >
                    Save
                </Button>
                <br /><br />
            </form>
        </>
    );

}


function content() {
    const formClasses = [{
        width: 50,
    }]
    return (
        <>
            <br />
            <FormCont
                formAttr={formClasses}
                form={form()}
            />

        </>
    );
}


function PackageForm() {
    const location = useLocation();
    const { name, date } = location.state;


    const breadcrumps = [
        {
            label: "Honoraria",
            icon: <ListAltIcon fontSize='small' />,
            route: "/all/honoraria"
        },
        {
            label: "Update Honorarium",
            icon: <EditIcon fontSize='small' />,
        }
    ]

    return (
        <>
            <GenCont disable_card="true" title={"Update " + name + "
Honorarium"} breadcrumps={breadcrumps} content={content()} />

        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(PackageForm);
```

6.3 Controller

```
<?php

namespace App\Http\Controllers;

use App\Honorarium;
use Illuminate\Http\Request;
use DB;

class HonorariumController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $honor =  $this->getallHon();
        } catch (Exception $e) {
        }

        return response()->json([
            'status' => 200,
            'honoraria' => $honor,
        ], 200);
    }
    function getallHon()
    {
        $honor =      Honorarium::join('users',  'honoraria.staff_id',  '=',
'users.id');
        if (request('from_date') && request('to_date')) {
            $honor->whereDate('honoraria.date_paid',                  '>=',
request('from_date'));
            $honor->whereDate('honoraria.date_paid',                  '<=',
request('to_date'));
```

```
        }
        $honor->select('honoraria.*',
DB::raw('CONCAT(users.first_name," "  ,  users.last_name)  as
staff'));
        $honor=$honor->get();

        return $honor;
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }
        // Warning: Data isn't being fully sanitized yet.
        try {
            $hon = Honorarium::create([
                'staff_id' => request('staff_id'),
                'date_paid' => request('date'),
                'amount' => request('amount')
            ]);
            return response()->json([
                'status' => 201,
                'id' => $hon->id
            ], 201);
        } catch (Exception $e) {
            return        $this->responseServerError('Error        creating
resource.');
        }
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \App\Honorarium  $honorarium
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $hon = Honorarium::where('id', $id)->firstOrFail();

            if (request('date')) {
                $hon->date_paid = request('date');
            }

            if (request('amount')) {
                $hon->amount = request('amount');
            }


            $hon->save();

            return response()->json([
                'status' => 200,
            ], 200);
        } catch (Exception $e) {
            return        $this->responseServerError('Error        updating
resource.');
        }
    }


    public function deleteHonorarium(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $honor = Honorarium::where('id', request('id'))->firstOrFail();
            $honor->delete();

            return response()->json([
                'status' => 200,
                'message' => "Deleted successfully",
                'honoraria' =>   $this->getallHon(),

            ], 200);
        } catch (Exception $e) {
            return        $this->responseServerError('Error        deleting
resource.');
        }
    }
}
```

7. Company Operating Expense

7.1 View

```
import React, { useState, useEffect } from 'react';
import { connect } from 'react-redux';
import GenCont from '../../../components/gen_container';
import Fab from '@material-ui/core/Fab';
import AddIcon from '@material-ui/icons/Add';
import { Link } from 'react-router-dom';
import Tooltip from '@material-ui/core/Tooltip';
import Http from '../../../Http';
import BootstrapTable from 'react-bootstrap-table-next';
import                'react-bootstrap-table-next/dist/react-bootstrap-
table2.min.css';
import     'react-bootstrap-table2-toolkit/dist/react-bootstrap-table2-
toolkit.min.css';
import { useSnackbar } from 'notistack';
import DeleteIcon from '@material-ui/icons/Delete';
import IconButton from '@material-ui/core/IconButton';
import EditIcon from '@material-ui/icons/Edit';
import ListAltIcon from '@material-ui/icons/ListAlt';
import Button from '@material-ui/core/Button';
import ReactTable from '../components/reactTable';
import DateRangeGen from '../../../components/daterangeGen';
import { useReactToPrint } from 'react-to-print';
import { ComponentToPrint } from '../components/printTemplate';
import Dialog from '@material-ui/core/Dialog';
import ImportExp from '../../../components/ImportCompExp';
import ReactHTMLTableToExcel from 'react-html-table-to-excel';

var _isMounted = false;

function content() {
    _isMounted = true;
    const api = '/api/v1/opex/company';
    const { enqueueSnackbar } = useSnackbar();

    const componentRef = React.useRef();
    const handlePrint = useReactToPrint({
        content: () => componentRef.current,
    });

    var                    localTerminal                    =
JSON.parse(localStorage.getItem("honorarium") || "[]");
    const [state, setState] = useState({ data: localTerminal });

    useEffect(() => {
        getData();
    }, []);

    const [openImp, setOpenImp] = React.useState(false);

    function getData(post = {}) {
        Http.post('/api/v1/opex/company/report', post)
            .then((response) => {
                localStorage.setItem('company_opex',
JSON.stringify(response.data.opex))
                localTerminal                    =
JSON.parse(localStorage.getItem("company_opex") || "[]");
                if (_isMounted) {
                    setState({ data: localTerminal })
                }
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch packages', { variant })
            });
```

```
    }

    function deleteItem(key) {
        var variant;

        if (confirm('Are you sure you want to proceed?')) {
            Http.delete(`/api/v1/opex/company/${key}`)
                .then((response) => {

                    getData();

                    variant = 'success'
                    enqueueSnackbar('Expense successfully deleted', { variant })
                })
                .catch((error) => {
                    console.log(error);
                    variant = 'error'
                    enqueueSnackbar('Error deleting Expense', { variant })
                });
        }

    }

    function actionFormat(cell, row) {
        var id = row.id;
        return (
            <>
                <IconButton aria-label="delete" style={{ padding: "0" }} onClick={() => deleteItem(id)} >
                    <DeleteIcon />
                </IconButton>
                  
                <Link to={{
                    pathname: `/form/operating-expenses/company`,
                    state: {
                        date: row.date_transact,
                        amount: row.amount,
                        item_id: row.opex_item_id,
                        desc: row.remarks,
                        type: "Update",
                        id: id
                    }
                }}>
                    <IconButton aria-label="delete" style={{ padding: "0" }} >
                        <EditIcon />
                    </IconButton>
                </Link>
            </>
        );
    }

    function getdates(data) {

        if (data == "clear") {
            getData()
        } else {
            console.log("get dates")
            console.log(data)

            getData({ from_date: data.datefrom, to_date: data.dateto })
        }

    }

    function downloadexcel() {
        return (
            <>
                <table style={{ visibility: "hidden" }} id="gen" class="table table-bordered">
                    <tr>
                        <td colSpan="4"><center>Company Operating Expenses</center></td>
                    </tr>
                    <tr>
                        <td>Date</td>
                        <td>Particular</td>
                        <td>Amount</td>
                        <td>Remarks</td>
                    </tr>

                    {state.data.map((itm, index, arr) => (
                        <tr>
                            <td>{itm.date_transact}</td>
                            <td>{itm.particular}</td>
                            <td>{itm.amount}</td>
                            <td>{itm.remarks}</td>
                        </tr>
                    ))}
                </table>
            </>
        )
    }

    const columns = [
        {
            dataField: 'id',
            text: ' ID',
            hidden: true
        }, {
            dataField: 'date_transact',
            text: 'Date',
            sort: true
        }, {
            dataField: 'particular',
            text: 'Particular',
            sort: true
        }, {
            dataField: 'amount',
            text: 'Amount',
            sort: true
        }, {
            dataField: 'remarks',
            text: 'Remarks',
            sort: true
        }, {
            dataField: 'opex_item_id',
            text: '',
            hidden: true
        }, {
            text: 'Actions',
            formatter: actionFormat,
            headerStyle: (colum, colIndex) => {
                return { width: '100px', textAlign: 'center' };
            }
        }

    ];

    function printContent() {
        const newcolumns = [
            {
                dataField: 'id',
                text: ' ID',
                hidden: true
            }, {
                dataField: 'date_transact',
                text: 'Date',
                sort: true
            }, {
                dataField: 'particular',
                text: 'Particular',
                sort: true
            }, {
                dataField: 'amount',
                text: 'Amount',
                sort: true
            }, {
                dataField: 'remarks',
                text: 'Remarks',
                sort: true
            }];

        return (
            <>
                <BootstrapTable keyField="id" data={state.data} columns={newcolumns} />
            </>
        )
    }

    var printdata = {
        title: 'Company Operating Expenses',
        content: printContent()
    }

    return (
        <>
            <Link style={{ float: "right" }}
                to={{
                    pathname: `/form/operating-expenses/company`,
                    state: {
                        type: "Add"
```

```
                }
            }}
        >
            <Tooltip title="Create new expense" aria-label="add">
                <Fab size="medium" color="primary" aria-label="add">
                    <AddIcon />
                </Fab>
            </Tooltip>
        </Link>
        <Button style={{ float: 'left' }} variant="contained" onClick={handlePrint} color="primary">Print</Button>

        <div style={{ float: 'left' }}>   <Button variant="contained" onClick={(e => setOpenImp(!openImp))} color="primary">Import</Button></div>
        <div style={{ float: 'left' }}>
              
            <ReactHTMLTableToExcel
                id="allItems"
                className="MuiButtonBase-root MuiButton-root MuiButton-contained MuiButton-containedPrimary"
                table="gen"
                filename="excelReports"
                sheet="all items"
                buttonText="Download as Excel" />
        </div>
        <div style={{ display: "none" }}><ComponentToPrint data={printdata} ref={componentRef} /></div>
        <br /><br /><br />
        <DateRangeGen dates={getdates} />
        <ReactTable columns={columns} data={state.data} />
        <Dialog aria-labelledby="simple-dialog-title" open={openImp} onClose={(e => setOpenImp(!openImp))} >
            <ImportExp added={getData} />
        </Dialog>
        <div style={{ display: 'none' }}>
            {downloadexcel()}
        </div>

    </>
  )
}

function AllPackages() {
  const breadcrumps = [
    {
      label: "Company Operating Expenses",
      icon: <ListAltIcon fontSize='small' />,
      route: "/all/operating-expenses/company"
    }
  ]

  return (
    <>
      <GenCont title="Company Operating Expenses" breadcrumps={breadcrumps} content={content()} />
      <br /><br />
    </>
  );
}

const mapStateToProps = (state) => ({
  isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllPackages);
```

7.2 Form

```
import React, { useState, useEffect } from 'react';
import { useLocation } from 'react-router-dom'
import { connect } from 'react-redux';
import GenCont from '../../components/gen_container';
import FormCont from '../../components/form_container';
import TextField from '@material-ui/core/TextField';
import SaveIcon from '@material-ui/icons/Save';
import Button from '@material-ui/core/Button';
import EditIcon from '@material-ui/icons/Edit';
import Http from '../../Http';
import { useSnackbar } from 'notistack';
import ListAltIcon from '@material-ui/icons/ListAlt';
import MenuItem from '@material-ui/core/MenuItem';
var _isMounted = false;

function form() {
  _isMounted = true;
  const { enqueueSnackbar } = useSnackbar();
  const location = useLocation();
  var date = null;
  var amount = null;
  var item_id = null;
  var desc = null;
  var variant;
  var id;

  if (location.state) {
    date = location.state.date;
    amount = location.state.amount;
    id = location.state.id;
    item_id = location.state.item_id;
    desc = location.state.desc;
  }

  var localOpexItm = JSON.parse(localStorage.getItem("opex_items") || "[]");
  const [state, setState] = useState({
    date: date,
    amount: amount,
    opexitems: localOpexItm,
    desc: desc,
    item_id: item_id
  });

  function onChange(event) {//Sets the value of states
    const { name, value } = event.target;
    if (_isMounted) {
      setState(prevState => ({ ...prevState, [name]: value }));
    }
  }

  function submit(e) {
    e.preventDefault();
    const posts = {
      date: state.date,
      amount: state.amount,
      desc: state.desc,
      item_id: state.item_id
    }
    if (location.state.type == "Update") {
      submitUpdate(posts)
    } else {
      submitAdd(posts)
    }

  }

  useEffect(() => {
    getOpexItems();
  }, []);


  function submitUpdate(posts) {
    Http.patch(`/api/v1/opex/company/${id}`, posts)
      .then(({ data }) => {
        console.log("updated")
        variant = 'success';
        enqueueSnackbar('Expense successfully updated', { variant })
      })
      .catch(() => {
        variant = 'error';
        enqueueSnackbar('Error updating Expense', { variant })
      });
  }

  function submitAdd(posts) {
    Http.post(`/api/v1/opex/company`, posts)
      .then(({ data }) => {
        variant = 'success';
        enqueueSnackbar('Expense successfully add', { variant })
      })
      .catch(() => {
        variant = 'error';
        enqueueSnackbar('Error adding Expense', { variant })
      });
  }

  function getOpexItems() {
    Http.get('/api/v1/opex/items')
      .then((response) => {
        localStorage.setItem('opex_items', JSON.stringify(response.data.opex_items))
```

```
            var localTerminal = JSON.parse(localStorage.getItem("opex_items") || "[]");
            if (_isMounted) {
                setState(prevState => ({ ...prevState, opexitems: localTerminal }));
            }

        })
        .catch((error) => {
            var variant = 'error';
            console.log(error);
            enqueueSnackbar('Failed to fetch opex items', { variant })
        });
    }


    return (
        <>
            <form onSubmit={submit} autoComplete="off">

                <TextField id="outlined-basic" type="date"
                    InputLabelProps={{ shrink: true, }}
                    required value={state.date} name="date" onChange={onChange} style={{ width: "100%" }} label="Date" variant="outlined" /><br /><br />
                <TextField
                    id="outlined-select-currency"
                    select
                    label="Particular"
                    value={state.item_id}
                    onChange={onChange}
                    name="item_id"
                    variant="outlined"
                    style={{ width: "100%" }}

                >
                    {state.opexitems.map((option) => (
                        <MenuItem key={option.id} value={option.id}>
                            {option.particular}
                        </MenuItem>
                    ))}
                </TextField>
                <br /><br />
                <TextField id="outlined-basic" required value={state.amount} name="amount" onChange={onChange} style={{ width: "100%" }} label="Amount" variant="outlined" /><br /><br />
                <TextField

                    value={state.desc}
                    name="desc"
                    onChange={onChange}
                    style={{ width: "100%" }}
                    id="outlined-multiline-static"
                    label="Remarks"
                    multiline
                    rows={4}
                    variant="outlined"
                />

                <br /><br />

                <Button
                    style={{ float: "right" }}
                    variant="contained"
                    color="primary"
                    size="medium"
                    startIcon={<SaveIcon />}
                    type="submit"
                >
                    Save
                </Button>
                <br /><br />
            </form>
        </>
    );

}


function content() {
    const formClasses = [{
        width: 50,
    }]
    return (
        <>
            <br />
            <FormCont
                formAttr={formClasses}
                form={form()}
            />

        </>
    );
}

function PackageForm() {
    const breadcrumps = [
        {
            label: "Company Operating Expenses",
            icon: <ListAltIcon fontSize='small' />,
            route: "/all/operating-expenses/company"
        },
        {
            label: "Add Expense",
            icon: <EditIcon fontSize='small' />,
        }
    ]

    return (
        <>
            <GenCont disable_card="true" breadcrumps={breadcrumps} content={content()} />
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(PackageForm);
```

7.3 Controller

```
<?php

namespace App\Http\Controllers;

use App\CompanyOperatingExpenses;
use Illuminate\Http\Request;
use DB;
class CompanyOperatingExpensesController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $opex = CompanyOperatingExpenses::join('opex_items', 'company_operating_expenses.opex_item_id', '=', 'opex_items.id');
            if (request('from_date') && request('to_date')) {
                $opex->whereDate('company_operating_expenses.date_transact', '>=', request('from_date'));
                $opex->whereDate('company_operating_expenses.date_transact', '<=', request('to_date'));
            }
            $opex->select('company_operating_expenses.*', 'opex_items.particular' , DB::raw('MONTH(company_operating_expenses.date_transact) as month'));
            $opex =$opex->get();
        } catch (Exception $e) {
        }

        return response()->json([
            'status' => 200,
            'opex' => $opex,
        ], 200);
    }

    public function ImportExp(Request $request)
```

```
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }
        try {
            if (!is_null(request("compExp"))) {
                $compex = json_decode(request('compExp'), true); //new

                foreach ($compex as $itm) {
                    if ($itm["date"]) {
                        $partId                                                    =
OpexItemController::checkCreateOpexItem(strtolower($itm["particular"]));

                        $res = CompanyOperatingExpenses::create([
                            'date_transact'
=>date_format(date_create($itm["date"]),"Y-m-d"),
                            'amount' => $itm["amount"],
                            'remarks' => $itm["remarks"],
                            'opex_item_id' => $partId
                        ]);
                    }
                }

            }

            return response()->json([
                'status' => 200,

            ], 200);
        } catch (Exception $e) {
            return        $this->responseServerError('Error        creating
resource.');
        }
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }
        // Warning: Data isn't being fully sanitized yet.
        try {
            $res = CompanyOperatingExpenses::create([
                'date_transact' => request('date'),
                'amount' => request('amount'),
                'remarks' => request('desc'),
                'opex_item_id' => request('item_id')
            ]);
            return response()->json([
                'status' => 201,
                'id' => $res->id
            ], 201);
        } catch (Exception $e) {
            return        $this->responseServerError('Error        creating
resource.');
        }
    }



    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     *          @param                  \App\CompanyOperatingExpenses
$companyOperatingExpenses
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $exp    =    CompanyOperatingExpenses::where('id',    $id)-
>firstOrFail();

            if (request('date')) {
                $exp->date_transact = request('date');
            }

            if (request('amount')) {
                $exp->amount = request('amount');
            }

            if (request('desc')) {
                $exp->remarks = request('desc');
            }

            if (request('item_id')) {
                $exp->opex_item_id = request('item_id');
            }


            $exp->save();

            return response()->json([
                'status' => 200,
            ], 200);
        } catch (Exception $e) {
            return        $this->responseServerError('Error        updating
resource.');
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     *          @param                  \App\CompanyOperatingExpenses
$companyOperatingExpenses
     * @return \Illuminate\Http\Response
     */
    public function destroy(Request $request, $id)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $pos    =    CompanyOperatingExpenses::where('id',    $id)-
>firstOrFail();
            $pos->delete();

            return response()->json([
                'status' => 204,
                'message' => "Deleted successfully"
            ], 204);
        } catch (Exception $e) {
            return        $this->responseServerError('Error        deleting
resource.');
        }
    }
}
```

8. Clients

8.1 View

```
import React, { useState, useEffect } from 'react';
import { connect } from 'react-redux';
import GenCont from '../../../components/gen_container';
import Fab from '@material-ui/core/Fab';
import AddIcon from '@material-ui/icons/Add';
import { Link } from 'react-router-dom';
import Tooltip from '@material-ui/core/Tooltip';
import Http from '../../../Http';
import Button from '@material-ui/core/Button';
import BootstrapTable from 'react-bootstrap-table-next';
import                  'react-bootstrap-table-next/dist/react-bootstrap-
table2.min.css';
import      'react-bootstrap-table2-toolkit/dist/react-bootstrap-table2-
toolkit.min.css';
import { useSnackbar } from 'notistack';
import DeleteIcon from '@material-ui/icons/Delete';
import IconButton from '@material-ui/core/IconButton';
import EditIcon from '@material-ui/icons/Edit';
import ListAltIcon from '@material-ui/icons/ListAlt';
import ReactTable from '../components/reactTable';
import { useReactToPrint } from 'react-to-print';
import { ComponentToPrint } from '../components/printTemplate';
```

```
var _isMounted = false;

function content() {
    _isMounted = true;
    const api = '/api/v1/client';
    const { enqueueSnackbar } = useSnackbar();

    const componentRef = React.useRef();
    const handlePrint = useReactToPrint({
        content: () => componentRef.current,
    });

    var localTerminal = JSON.parse(localStorage.getItem("clients") ||
"[]");
    const [state, setState] = useState({ data: localTerminal });


    useEffect(() => {
        Http.get(api)
            .then((response) => {
                localStorage.setItem('clients',
JSON.stringify(response.data.clients))
                localTerminal                                          =
JSON.parse(localStorage.getItem("clients") || "[]");
                if (_isMounted) {
                    setState({ data: localTerminal })
                }

            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch clients', { variant })
            });
    }, []);

    function deleteCl(key) {
        var variant;
        if (confirm('Are you sure you want to proceed?')) {

            Http.post('/api/v1/client/delete', {id: key})
                .then((response) => {
                        if (_isMounted) {
                            localStorage.setItem('clients',
JSON.stringify(response.data.clients))
                            localTerminal                              =
JSON.parse(localStorage.getItem("clients") || "[]");
                            if (_isMounted) {

                                setState({ data: localTerminal })
                            }
                        } // }
                    variant = 'success'
                    enqueueSnackbar('client    successfully    deleted',    {
variant })
                })
                .catch((error) => {
                    console.log(error);
                    variant = 'error'
                    enqueueSnackbar('Error deleting client', { variant })
                });

        }

    }

    function actionFormat(cell, row) {
        var id = row.id;
        return (
            <>
                <IconButton  aria-label="delete"  style={{  padding:  "0"  }}
onClick={() => deleteCl(id)} >
                    <DeleteIcon />
                </IconButton>
                  
                <Link to={{
                    pathname: `/form/client`,
                    state: {
                        first_name: row.first_name,
                        middle_name: row.middle_name,
                        last_name: row.last_name,
                        contact_no: row.contact_no,
                        address: row.address,
                        email: row.email,
                        type: "Update",
                        id: id
                    }
                }}>
                    <IconButton aria-label="delete" style={{ padding: "0" }}
>
                        <EditIcon />
                    </IconButton>
                </Link>
            </>
        );
    }
function nameFormat(cell, row) {

        return (
            <>{row.first_name  +  "  "  +  row.middle_name  +  "  "  +
row.last_name} </>
        );
    }

    const columns = [
        {
            dataField: 'id',
            text: ' ID',
            hidden: true
        }, {
            dataField: 'id',
            text: 'Name',
            formatter: nameFormat,
            sort: true
        }, {
            dataField: 'first_name',
            text: 'h',
            hidden: true
        }, {
            dataField: 'middle_name',
            text: 'h',
            hidden: true
        }, {
            dataField: 'last_name',
            text: 'h',
            hidden: true
        },{
            dataField: 'contact_no',
            text: 'Contact',
            sort: true
        },{
            dataField: 'address',
            text: 'Address',
            sort: true
        },{
            dataField: 'email',
            text: 'Email',
            sort: true
        },{
            dataField: 'balance',
            text: 'Balance',
            sort: true
        },{
            text: 'Actions',
            formatter: actionFormat,
            headerStyle: (colum, colIndex) => {
                return { width: '100px', textAlign: 'center' };
            }
        }];

        function printContent(){
            const newcolumns = [
                {
                    dataField: 'id',
                    text: ' ID',
                    hidden: true
                }, {
                    dataField: 'id',
                    text: 'Name',
                    formatter: nameFormat,
                    sort: true
                }, {
                    dataField: 'first_name',
                    text: 'h',
                    hidden: true
                }, {
                    dataField: 'middle_name',
                    text: 'h',
                    hidden: true
                }, {
                    dataField: 'last_name',
                    text: 'h',
                    hidden: true
                },{
```

```
            dataField: 'contact_no',
            text: 'Contact',
            sort: true
        },{
            dataField: 'address',
            text: 'Address',
            sort: true
        },{
            dataField: 'email',
            text: 'Email',
            sort: true
        },{
            dataField: 'balance',
            text: 'Balance',
            sort: true
        }];

    return(
        <>  <BootstrapTable keyField="id" data={state.data}
columns={newcolumns} /> </>
    )
  }

  var printdata = {
    title: 'Clients',
    content: printContent()
  }

 return (
  <>
    <Link style={{ float: "right" }}
      to={{
        pathname: `/form/client`,
        state: {
          type: "Add",
        }
      }} >
      <Tooltip title="Add new client" aria-label="add">
        <Fab size="medium" color="primary" aria-label="add">
          <AddIcon />
        </Fab>
      </Tooltip>
    </Link>
    <Button style={{ float: 'left' }} variant="contained"
onClick={handlePrint} color="primary">Print</Button>
    <div style={{ display: "none" }}><ComponentToPrint
data={printdata} ref={componentRef} /></div>
    <br /><br /> <br />

    <ReactTable columns={columns}  data={state.data}/>
  </>
 )
}

function AllItems() {
 const breadcrumps = [
  {
    label: "Clients",
    icon: <ListAltIcon fontSize='small' />,
    route: "/all/clients"
  }
 ]

 return (
  <>
    <GenCont title="All Clients" breadcrumps={breadcrumps}
content={content()} />
    <br /><br />
  </>
 );
}

const mapStateToProps = (state) => ({
 isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllItems);
```

8.2 Form

```
import React, { useState } from 'react';
import { useLocation } from 'react-router-dom';
import { connect } from 'react-redux';
import GenCont from '../../components/gen_container';
import FormCont from '../../components/form_container';
import AddIcon from '@material-ui/icons/Add';
import TextField from '@material-ui/core/TextField';
import SaveIcon from '@material-ui/icons/Save';
import Button from '@material-ui/core/Button';
import EditIcon from '@material-ui/icons/Edit';
import Http from '../../Http';
import { useSnackbar } from 'notistack';
import ListAltIcon from '@material-ui/icons/ListAlt';
var _isMounted = false;

function PackageForm(props) {
  const [stateId, setId] = useState(null);
  function form() {
    _isMounted = true;
    const api = '/api/v1/client';
    const { enqueueSnackbar } = useSnackbar();
    const location = useLocation();
    var first_name = "";
    var middle_name = "";
    var last_name = "";
    var contact_no = "";
    var address = "";
    var email = "";


    var variant;
    var id;

    if (location.state.type == "Update") {
      first_name = location.state.first_name;
      middle_name = location.state.middle_name;
      last_name = location.state.last_name;
      contact_no = location.state.contact_no;
      address = location.state.address;
      email = location.state.email;
      id = location.state.id;
    }

    const [state, setState] = useState({
      first_name: first_name,
      middle_name: middle_name,
      last_name: last_name,
      contact_no: contact_no,
      address: address,
      email: email,
    });

    function onChange(event) {//Sets the value of states
      const { name, value } = event.target;
      if (_isMounted) {
        setState(prevState => ({ ...prevState, [name]: value }));
      }
    }

    function submit(e) {
      e.preventDefault();
      const posts = {
        first_name: state.first_name,
        middle_name: state.middle_name,
        last_name: state.last_name,
        contact_no: state.contact_no,
        address: state.address,
        email: state.email,
      }

      location.state.type == "Update" ? submitUpdate(posts) :
submitCreate(posts);

    }

    function submitCreate(posts) {

      Http.post(api, posts)
        .then(({ data }) => {
          variant = 'success';
          enqueueSnackbar('Client successfully created', {
variant })
          if (_isMounted) {
            setState(prevState => ({
              ...prevState,
              first_name: "",
              middle_name: "",
              last_name: "",
              contact_no: "",
              address: "",
              email: ""
```

```
                }));

                setId(data.id)
            }
        })
        .catch((error) => {
            variant = 'error';
            console.log(error);
            enqueueSnackbar('Failed to create client', { variant })
        });
    }

    function submitUpdate(posts) {
        Http.patch(`${api}/${id}`, posts)
        .then(({ data }) => {
            variant = 'success';
            enqueueSnackbar('Client successfully updated', { variant })
        })
        .catch(() => {
            variant = 'error';
            enqueueSnackbar('Error updating client', { variant })
        });
    }


    return (
        <>
            <form onSubmit={submit} autoComplete="off">

                <TextField id="outlined-basic" required value={state.first_name} name="first_name" onChange={onChange} style={{ width: "100%" }} label="First Name" variant="outlined" /><br /><br />
                <TextField id="outlined-basic" required value={state.middle_name} name="middle_name" onChange={onChange} style={{ width: "100%" }} label="Middle Name" variant="outlined" /><br /><br />
                <TextField id="outlined-basic" required value={state.last_name} name="last_name" onChange={onChange} style={{ width: "100%" }} label="Last Name" variant="outlined" /><br /><br />
                <TextField id="outlined-basic" required value={state.contact_no} name="contact_no" onChange={onChange} style={{ width: "100%" }} label="Conntact No" variant="outlined" /><br /><br />
                <TextField id="outlined-basic" required value={state.address} name="address" onChange={onChange} style={{ width: "100%" }} label="Address" variant="outlined" /><br /><br />
                <TextField id="outlined-basic" required value={state.email} name="email" onChange={onChange} style={{ width: "100%" }} type="email" label="Email" variant="outlined" /><br /><br />

                <Button
                    style={{ float: "right" }}
                    variant="contained"
                    color="primary"
                    size="medium"
                    startIcon={<SaveIcon />}
                    type="submit"
                >
                    Save
            </Button>
                <br /><br />
            </form>
        </>
    );

}


function content() {
    const formClasses = [{
        width: 50,
    }]
    return (
        <>
            <br />

            <FormCont
                formAttr={formClasses}
                form={form()}
            />

        </>
    );
}

const location = useLocation();
const { type } = location.state;

var label = "Add New Client";
var icon = <AddIcon fontSize='small' />;
var title = "Add New Client";

if (type == "Update") {
    label = "Update";
    icon = <EditIcon fontSize='small' />;
    title = "Update Client";
}

var breadcrumps = [
    {
        label: "Clients",
        icon: <ListAltIcon fontSize='small' />,
        route: "/all/clients"
    },
    {
        label: label,
        icon: icon,
        states: { type: type }
    }
]

if (type == "Order") {
    breadcrumps = [
        {
            label: "Order",
            icon: <ListAltIcon fontSize='small' />,
            route: "/form/order/basic_details",
            states: { type: "form", client_id: stateId }
        },
        {
            label: label,
            icon: icon,
            states: { type: type }
        }
    ]
}

return (
    <><GenCont disable_card="true" title={title} breadcrumps={breadcrumps} content={content()} /></>
);
}

const mapStateToProps = (state) => ({
isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(PackageForm);
```

8.3 Controller

```
<?php

namespace App\Http\Controllers;

use App\Client;
use App\Order;
use Illuminate\Http\Request;

class ClientController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {

            return response()->json([
                'status' => 200,
                'clients' => $this->getallclient(),
```

```
            ], 200);
        } catch (Exception $e) {
        }


    }

    public static function checkCreateClient($fname, $mname, $lname){

        $check = Client::whereRaw('lower(first_name)  like ?',$fname)
        ->whereRaw('lower(middle_name)  like ?', $mname)
        ->whereRaw('lower(last_name)  like ?',$lname)
        ->first();

        $id = null;
        if(!is_null($check)){
            $id = $check->id;
        }else{
            $cl = Client::create([
                'first_name' => $fname,
                'middle_name' => $mname,
                'last_name' =>$lname,
                'contact_no' => '',
                'address' => '',
                'email' => '',
                'balance' => 0.00
            ]);

            $id = $cl->id;
        }

        return $id;


    }


    public function singleClient(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $cl = Client::where('clients.id', '=',request('client_id'))->get();

            return response()->json([
                'status' => 200,
                'client' => $cl,
            ], 200);
        } catch (Exception $e) {
        }


    }
    function getallclient()
    {
        $cl = Client::select('clients.*')->get();

        return $cl;
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }
        // Warning: Data isn't being fully sanitized yet.
        try {
            $cl = Client::create([
                'first_name' => request('first_name'),
                'middle_name' => request('middle_name'),
                'last_name' => request('last_name'),
                'contact_no' => request('contact_no'),
                'address' => request('address'),
                'email' => request('email'),
                'balance' => 0.00
            ]);

            return response()->json([
                'status' => 201,
                'id' => $cl->id
            ], 201);
        } catch (Exception $e) {
            return $this->responseServerError('Error creating resource.');
        }
    }



    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \App\Client  $client
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $cl = Client::where('id', $id)->firstOrFail();

            if (request('first_name')) {
                $cl->first_name = request('first_name');
            }

            if (request('last_name')) {
                $cl->last_name = request('last_name');
            }

            if (request('middle_name')) {
                $cl->middle_name = request('middle_name');
            }

            if (request('contact_no')) {
                $cl->contact_no = request('contact_no');
            }

            if (request('email')) {
                $cl->email = request('email');
            }

            if (request('address')) {
                $cl->address = request('address');
            }

            $cl->save();

            return response()->json([
                'status' => 200,
            ], 200);
        } catch (Exception $e) {
            return $this->responseServerError('Error updating resource.');
        }
    }


    public function deleteClient(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $cl = Client::where('id', request('id'))->firstOrFail();

            $order_exist = Order::where('client_id', request('id'))->first();

            if(is_null($order_exist)){
                $cl->delete();

                return response()->json([
                    'status' => 200,
                    'message' => "Deleted successfully",
                    'clients' =>  $this->getallclient(),

                ], 200);
```

```
            }else{
                return response()->json([
                    'status' => 400,

                ], 400);
            }

        } catch (Exception $e) {
            return $this->responseServerError('Error deleting
resource.');
        }
    }
}
```

9. Orders

9.1 Components (Form)

A. Progress Step

```
import React, { useState, useEffect } from 'react';
import { makeStyles, withStyles } from '@material-ui/core/styles';
import { connect } from 'react-redux';
import Http from '../../../Http';
import { useLocation } from 'react-router-dom';
import PropTypes from 'prop-types';
import clsx from 'clsx';
import Stepper from '@material-ui/core/Stepper';
import Step from '@material-ui/core/Step';
import StepLabel from '@material-ui/core/StepLabel';
import Check from '@material-ui/icons/Check';
import StepConnector from '@material-ui/core/StepConnector';
import MenuBookIcon from '@material-ui/icons/MenuBook';
import ListAltIcon from '@material-ui/icons/ListAlt';
import AttachMoneyIcon from '@material-ui/icons/AttachMoney';
import VerifiedUserIcon from '@material-ui/icons/VerifiedUser';
import HourglassFullIcon from '@material-ui/icons/HourglassFull';

var _isMounted = false;

// ------------------------

  const useQontoStepIconStyles = makeStyles({
    root: {
      color: '#eaeaf0',
      display: 'flex',
      height: 22,
      alignItems: 'center',
    },
    active: {
      color: '#784af4',
    },
    circle: {
      width: 8,
      height: 8,
      borderRadius: '50%',
      backgroundColor: 'currentColor',
    },
    completed: {
      color: '#784af4',
      zIndex: 1,
      fontSize: 18,
    },
  });

  function QontoStepIcon(props) {
    const classes = useQontoStepIconStyles();
    const { active, completed } = props;

    return (
      <div
        className={clsx(classes.root, {
          [classes.active]: active,
        })}
      >
        {completed ? <Check className={classes.completed} /> :
<div className={classes.circle} />}
      </div>
    );
  }

  QontoStepIcon.propTypes = {
    active: PropTypes.bool,
    completed: PropTypes.bool,
  };

  const ColorlibConnector = withStyles({
    alternativeLabel: {
      top: 22,
    },
    active: {
      '& $line': {
        backgroundImage:
          'linear-gradient(  95deg,rgb(242,113,33)  0%,rgb(233,64,87)
50%,rgb(138,35,135) 100%)',
      },
    },
    completed: {
      '& $line': {
        backgroundImage:
          'linear-gradient(  95deg,rgb(242,113,33)  0%,rgb(233,64,87)
50%,rgb(138,35,135) 100%)',
      },
    },
    line: {
      height: 3,
      border: 0,
      backgroundColor: '#eaeaf0',
      borderRadius: 1,
    },
  })(StepConnector);

  const useColorlibStepIconStyles = makeStyles({
    root: {
      backgroundColor: '#ccc',
      zIndex: 1,
      color: '#fff',
      width: 50,
      height: 50,
      display: 'flex',
      borderRadius: '50%',
      justifyContent: 'center',
      alignItems: 'center',
    },
    active: {
      backgroundImage:
        'linear-gradient( 136deg, rgb(242,113,33) 0%, rgb(233,64,87)
50%, rgb(138,35,135) 100%)',
      boxShadow: '0 4px 10px 0 rgba(0,0,0,.25)',
    },
    completed: {
      backgroundImage:
        'linear-gradient( 136deg, rgb(242,113,33) 0%, rgb(233,64,87)
50%, rgb(138,35,135) 100%)',
    },
  });

  function ColorlibStepIcon(props) {
    const classes = useColorlibStepIconStyles();
    const { active, completed } = props;

    const icons = {
      1: <ListAltIcon />,
      2: <MenuBookIcon />,
      3: <AttachMoneyIcon />,
      4: <HourglassFullIcon />,
      5: <VerifiedUserIcon />,
    };

    return (
      <div
        className={clsx(classes.root, {
          [classes.active]: active,
          [classes.completed]: completed,
        })}
      >
        {icons[String(props.icon)]}
      </div>
    );
  }

  ColorlibStepIcon.propTypes = {
    active: PropTypes.bool,
    completed: PropTypes.bool,
    icon: PropTypes.node,
  };

// ------------------------


function getSteps() {
```

```
    return ['Basic Order Details', 'Service Selection', 'Costing', 'For
Verification', 'Verified'];
}

function content(c) {
  _isMounted = true
  const location = useLocation();
  var num = 0;
  const [state, setState] = useState({
    status: " ",
    num: 0
});


  useEffect(() => {
    Http.post('/api/v1/order/specific', { id: location.state.id })
        .then((response) => {

            var data = response.data.order;
            if (_isMounted) {
              if(data[0].transaction_status == "Service Selection"){num =
1}
              if(data[0].transaction_status == "Costing"){num = 2}
              if(data[0].transaction_status == "For Verification"){num = 3}
              if(data[0].transaction_status == "Verified"){num = 4}

              setState(prevState      =>      ({      ...prevState,      status:
data[0].transaction_status, num: num }));

            }
        })
        .catch((error) => {
            console.log(error);
        });
}, []);

  const steps = getSteps();
    _isMounted = true;

    return (
        <>
            <Stepper        alternativeLabel        activeStep={state.num}
connector={<ColorlibConnector />}>
                {steps.map((label) => (
                    <Step key={label}>
                        <StepLabel
StepIconComponent={ColorlibStepIcon}>{label}</StepLabel>
                    </Step>
                ))}
            </Stepper>
        </>
    )
}

function AllCats(props) {
    return (
        <>
        {content(props.transStat)}
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllCats);
```

B. Basic Details

```
import React, { useState, useEffect } from 'react';
import { useLocation } from 'react-router-dom';
import { connect } from 'react-redux';
import OrderFormContainer from '../orders/order_form_container';
import { useSnackbar } from 'notistack';
import TextField from '@material-ui/core/TextField';
import Button from '@material-ui/core/Button';
import NavigateNextIcon from '@material-ui/icons/NavigateNext';
import MenuItem from '@material-ui/core/MenuItem';
import PersonAddIcon from '@material-ui/icons/PersonAdd';
import { makeStyles } from '@material-ui/core/styles';
import { Link, useHistory } from 'react-router-dom';
import Tooltip from '@material-ui/core/Tooltip';
import Http from '../../../Http';
import FormLabel from '@material-ui/core/FormLabel';
var _isMounted = false;

const useStyles = makeStyles((theme) => ({
    root: {
        '& > *': {
            margin: theme.spacing(1),
        },
    },
    extendedIcon: {
        marginRight: theme.spacing(1),
    },
}));

function content(props) {
    const location = useLocation();
    var variant;
    _isMounted = true;
    const api = '/api/v1/order';
    const { enqueueSnackbar } = useSnackbar();

    var localTerminal = JSON.parse(localStorage.getItem("clients") ||
"[]");
    var                    localTerminalP                    =
JSON.parse(localStorage.getItem("packages") || "[]");


    const [state, setState] = useState({
        first_name: null,
        middle_name: null,
        last_name: null,
        allclients: localTerminal,
        allpackages: localTerminalP,
        client_id: location.state.client_id,
        package_id: location.state.package_id,
        payment_method: location.state.payment_status,
        id: location.state.id,
        locType: location.state.type,
        readonly: location.state.readonly,
        code: location.state.code,
        added: null,
        transStat: null,
    });

    useEffect(() => {
        Http.get('/api/v1/client')
            .then((response) => {
                localStorage.setItem('clients',
JSON.stringify(response.data.clients))
                localTerminal                                        =
JSON.parse(localStorage.getItem("clients") || "[]");
                if (_isMounted) {
                    setState(prevState    =>    ({    ...prevState,    allclients:
localTerminal }));
                }
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch clients', { variant })
            });
    }, []);


    useEffect(() => {
        Http.get('/api/v1/packages')
            .then((response) => {
                localStorage.setItem('packages',
JSON.stringify(response.data.packages))
                localTerminalP                                       =
JSON.parse(localStorage.getItem("packages") || "[]");
                if (_isMounted) {
                    setState(prevState    =>    ({    ...prevState,    allpackages:
localTerminalP }));
                }
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch packages', { variant })
            });
    }, []);

    if (state.locType == "update" && state.added == null) {
        useEffect(() => {
            Http.post('/api/v1/order/specific', { id: state.id })
                .then((response) => {


                    var data = response.data.order;
```

```
                if (_isMounted) {
                    setState(prevState => ({
                        ...prevState,
                        client_id: data[0].client_id,
                        package_id: data[0].package_id,
                        payment_method: data[0].payment_status,
                        code: data[0].code,
                        transStat: data[0].transaction_status
                    }));


                }
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch packages', { variant
})
            });
    }, []);
}



function onChange(event) {//Sets the value of states
    const { name, value } = event.target;
    if (_isMounted) {
        setState(prevState => ({ ...prevState, [name]: value }));
    }
}


function submit(e) {
    e.preventDefault();
    const posts = {
        client_id: state.client_id,
        package_id: state.package_id,
        payment_status: state.payment_method,
    }
    state.locType == "update" ? submitUpdate(posts) :
submitCreate(posts);

}

function submitCreate(posts) {
    var variant;
    Http.post(api, posts)
        .then(({ data }) => {
            variant = 'success';
            enqueueSnackbar('Order successfully created', { variant
})

            setState(prevState => ({
                ...prevState,
                id: data.id,
                added: "yes",
                locType: "update",
                code: data.code,
                client_id: posts.client_id
            }));

        })
        .catch((error) => {
            variant = 'error';
            console.log(error);
            enqueueSnackbar('Failed to create order', { variant })
        });
}

function submitUpdate(posts) {
    Http.patch(`${api}/${state.id}`, posts)
        .then(({ data }) => {
            variant = 'success';
            enqueueSnackbar('Order successfully updated', { variant
})
        })

        .catch(() => {
            variant = 'error';
            enqueueSnackbar('Failed updating Order', { variant })
        });
}


const allclients = state.allclients
const allpackages = state.allpackages
const pmtmethod = ["Full Payment", "Partial Payment"]


return (
    <>
        <center>
            <FormLabel variant="h6" gutterBottom>
                Basic Order Details
            </FormLabel>
        </center>
        <hr />
        <form
            onSubmit={submit}
            autoComplete="off">
            {
                state.locType != "update" ?
                    <>
                        <Link to={{
                            pathname: `/form/client`,
                            state: {
                                type: "Order",
                            }
                        }}>
                            <Tooltip title="Add new client" aria-
label="add">
                                <Button color="primary" >
                                    <PersonAddIcon />
                                </Button>
                            </Tooltip>
                        </Link>
                        <br /> <br />
                    </>
                    : <>
                        <small>
                            <FormLabel  gutterBottom>
                                {state.code}
                            </FormLabel>
                        </small>
                        <br /><br />
                    </>

            }

            <TextField
                disabled={state.transStat == "Verified" || state.readonly
== true  ? true : false}
                id="outlined-select-currency"
                select
                label="Client"
                value={state.client_id ? state.client_id : null}
                onChange={onChange}
                name="client_id"
                variant="outlined"
                style={{ width: "100%" }}
                required
                InputLabelProps={
                    state.client_id ?
                        {
                            shrink: true,
                        }
                        : {}
                }
            >
                {allclients.map((option) => (
                    <MenuItem key={option.id} value={option.id}>
                        {option.first_name + " " + option.last_name}
                    </MenuItem>
                ))}
            </TextField>

            <br /><br />

            <TextField
                disabled={state.transStat == "Verified" || state.readonly
== true ? true : false}
                id="outlined-select-currency"
                select
                label="Package"
                InputLabelProps={
                    state.package_id ?
                        {
                            shrink: true,
                        } :
                        {}
                }
                value={state.package_id ? state.package_id : null}

                onChange={onChange}
                name="package_id"
```

```
                variant="outlined"
                style={{ width: "100%" }}
                required
            >
                {allpackages.map((option) => (
                    <MenuItem key={option.id} value={option.id}>
                        {option.name}
                    </MenuItem>
                ))}
            </TextField><br /><br />
            <TextField
             disabled={state.transStat == "Verified" || state.readonly
== true  ? true : false}
                InputLabelProps={
                    state.payment_method ?
                        {
                            shrink: true,
                        }
                        : {}
                }
                id="outlined-select-currency"
                select
                label="Payment Method"
                value={state.payment_method                              ?
state.payment_method : null}
                onChange={onChange}
                name="payment_method"
                variant="outlined"
                style={{ width: "100%" }}
                required
            >
                {pmtmethod.map((option) => (
                    <MenuItem key={option} value={option}>
                        {option}

                    </MenuItem>
                ))}
            </TextField>

            <br /><br />

            <Button
                style={{ float: "left" }}
                variant="contained"
                color="primary"
                size="medium"
                type="submit"
                disabled={state.transStat == "Verified" || state.readonly
== true ? true : false}
            >
                Save
            </Button>
            {
                state.id ?
                    <div style={{ float: "right" }}>
                        <Link to={{
                            pathname: `/form/order/service_selection`,
                            state: {
                                package_id: state.package_id,
                                id: state.id,
                                type: "update",
                                readonly: state.readonly
                            }
                        }} >

                            <Button

                                variant="contained"
                                color="primary"
                                size="medium"
                                startIcon={<NavigateNextIcon />}
                            >
                                Next
                            </Button>
                        </Link>
                    </div>
                    : <></>
            }

            <br /><br />
        </form>
    </>
  )
}

function BasicDetails(props) {
  const ordprops = {
    cont: content(),
    type: "form"
  }

  return (
    <>
      <OrderFormContainer ordprops={ordprops} />
      <br /><br />
    </>
  );
}

const mapStateToProps = (state) => ({
  isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(BasicDetails);
```

C. Service Selection

```
import React, { useState, useEffect } from 'react';
import { useLocation } from 'react-router-dom';
import { connect } from 'react-redux';
import OrderFormContainer from '../orders/order_form_container';
import { useSnackbar } from 'notistack';
import TextField from '@material-ui/core/TextField';
import Button from '@material-ui/core/Button';
import NavigateNextIcon from '@material-ui/icons/NavigateNext';
import NavigateBeforeIcon from '@material-
ui/icons/NavigateBefore';
import Fab from '@material-ui/core/Fab';
import { Link } from 'react-router-dom';
import Tooltip from '@material-ui/core/Tooltip';
import Http from '../../../Http';
import AddUserdef from './add_userdefined_items';
import CreateIcon from '@material-ui/icons/Create';
import Checkbox from '@material-ui/core/Checkbox';
import FormGroup from '@material-ui/core/FormGroup';
import FormControlLabel from '@material-
ui/core/FormControlLabel';
import FormControl from '@material-ui/core/FormControl';
import FormLabel from '@material-ui/core/FormLabel';
import Typography from '@material-ui/core/Typography';
import Accordion from '@material-ui/core/Accordion';
import AccordionSummary from '@material-
ui/core/AccordionSummary';
import AccordionDetails from '@material-ui/core/AccordionDetails';
import ExpandMoreIcon from '@material-ui/icons/ExpandMore';
import DeleteIcon from '@material-ui/icons/Delete';
import Dialog from '@material-ui/core/Dialog';
import DialogActions from '@material-ui/core/DialogActions';
import DialogContent from '@material-ui/core/DialogContent';
import DialogTitle from '@material-ui/core/DialogTitle';

var _isMounted
function ServiceSelc(props) {
  const location = useLocation();
  _isMounted = true;
  const  local_user  =  JSON.parse(localStorage.getItem("user")  ||
"[]");
  const { enqueueSnackbar } = useSnackbar();
  const [state, setState] = useState({
    items: [],
    package_id: null,
    selId: null,
    selName: null,
    selRemark: null,
    package_name: null,
    selected: [],
    selectedUsdef: [],
    userdefItems: [],
    transStat: "Verified",
    role: local_user.role,
    readonly: location.state.readonly,

  });

  useEffect(() => {
    Http.post('/api/v1/order/specific', { id: location.state.id })
      .then((response) => {


        var data = response.data.order;
        if (_isMounted) {
          setState(prevState => ({
```

```
                    ...prevState,
                    package_id: data[0].package_id,
                    package_name: data[0].package,
                    transStat: data[0].transaction_status
                }));

                Http.post('/api/v1/packitms/index',  {  package_id:
data[0].package_id })
                    .then((response) => {

                        if (_isMounted) {
                            setState(prevState  =>  ({  ...prevState,  items:
response.data.items }));
                        }
                    })
                    .catch((error) => {
                        var variant = 'error';
                        console.log(error);
                        enqueueSnackbar('Failed to fetch items', { variant
})
                    });
            }
        })
        .catch((error) => {
            var variant = 'error';
            console.log(error);
            enqueueSnackbar('Failed to fetch packages', { variant })
        });
    }, []);

    useEffect(() => {
        getUserDef();
    }, []);

    function getUserDef() {
        Http.post('/api/v1/order/userdefOrderItem',  {  order_id:
location.state.id })
            .then((response) => {
                if (_isMounted) {
                    setState(prevState => ({
                        ...prevState,
                        userdefItems: response.data.items,
                    }));
                }
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch packages', { variant })
            });
    }

    useEffect(() => {
        Http.post('/api/v1/order/item/all', { order_id: location.state.id })
            .then((response) => {

                var sel = response.data.items
                var items = [];

                sel.map((option) => {
                    var it = {}

                    it.item_id = option.item_id
                    it.checked = true
                    it.particular = option.particular
                    items.push(it);
                })

                if (_isMounted) {
                    setState(prevState  =>  ({  ...prevState,  selected:  items
}));
                }
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch items', { variant })
            });

    }, []);

    function handleInputChange(event) {
        const target = event.target;
        var value = target.value;
        var seltem = state.selected
        if (target.checked) {
            var subs = {
                item_id: parseInt(value),
                checked: true
            }
            seltem.push(subs);
        } else {
            const index = seltem.findIndex(
                (sc) => parseInt(sc.item_id, 10) === parseInt(value, 10),
            );
            seltem = [...seltem.slice(0, index), ...seltem.slice(index + 1)];
        }
        setState(prevState => ({ ...prevState, selected: seltem }));
    }

    function handleUserdef(event) {
        const target = event.target;
        var value = target.value;
        var seltem = state.selectedUsdef
        if (target.checked) {
            var subs = {
                id: parseInt(value),
                checked: true
            }
            seltem.push(subs);
        } else {
            const index = seltem.findIndex(
                (sc) => parseInt(sc.id, 10) === parseInt(value, 10),
            );
            seltem = [...seltem.slice(0, index), ...seltem.slice(index + 1)];
        }
        setState(prevState  =>  ({  ...prevState,  selectedUsdef:  seltem
}));
    }

    function submit(e) {
        e.preventDefault();
        const posts = {
            order_id: location.state.id,
            items: JSON.stringify(state.selected),
        }

        Http.post('/api/v1/order/item', posts)
            .then((response) => {

                var variant = 'success';
                enqueueSnackbar('Client successfully created', { variant
})
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch items', { variant })
            });

    }

    function submitDelete() {
        if (confirm("Are you sure you want to delete these services?"))
{
            Http.post('/api/v1/order/deleteUserdefinedItems',  {  items:
JSON.stringify(state.selectedUsdef) })
                .then((response) => {
                    getUserDef();
                    var variant = 'success';
                    enqueueSnackbar('Client  successfully  created',  {
variant })
                })
                .catch((error) => {
                    var variant = 'error';
                    console.log(error);
                    enqueueSnackbar('Failed to fetch items', { variant })
                });
        }
    }

    function submitupdate() {
        if (confirm("Are you sure you want to update this service?")) {
            const subs = {
                id: state.selId,
                item_name: state.selName,
                remarks: state.selRemark

            }

            Http.post('/api/v1/order/updateUserdefinedItems', subs)
                .then((response) => {
                    getUserDef();
                    var variant = 'success';
```

```
            enqueueSnackbar('Item successfully updated', { variant })
          })
          .catch((error) => {
            var variant = 'error';
            console.log(error);
            enqueueSnackbar('Failed to update item', { variant })
          });
    }
  }

  function content() {

    var items = state.items;
    var userdefItems = state.userdefItems;
    var allsel = state.selected;

    function systemDefine() {
      return (
        <>
          <form onSubmit={submit} autoComplete="off">
            <FormControl component="fieldset">


              <FormGroup aria-label="position" >
                {items.map((option) => (
                  <>

                    <FormControlLabel
                      disabled={state.transStat == "Verified" || state.readonly == true ? true : false}
                      onChange={handleInputChange}
                      value={option.id}
                      checked={allsel.filter(e2 => e2.item_id == option.id).length > 0 ? true : false}
                      control={<Checkbox color="primary" />}
                      label={option.particular}
                      labelPlacement="end"
                    />

                  </>
                ))}

              </FormGroup>
            </FormControl>
            <br />
            <Button
              style={{ float: "left" }}
              variant="contained"
              color="primary"
              size="medium"
              type="submit"
              disabled={state.transStat == "Verified" || state.readonly == true ? true : false}
            >
              Save
          </Button>
            </form>

          </>
      );
    }
    function onChange(event) {//Sets the value of states
      const { name, value } = event.target;
      _isMounted = true;
      if (_isMounted) {
        setState(prevState => ({ ...prevState, [name]: value }));
      }
    }

    const [open, setOpen] = React.useState(false);

    function handleClickOpen(e) {
      const { id, name, remark } = e.target.dataset;
      setState(prevState => ({ ...prevState, selId: id, selName: name, selRemark: remark }));
      setOpen(true);
    };

    const handleClose = () => {
      setOpen(false);
    };

    function userDefine() {
      return (
        <>
          <form style={{ display: "block" }} onSubmit={submit} autoComplete="off" style={{ width: "100%" }}>
            <AddUserdef newUd={getUserDef} />
            <Tooltip title="Delete user defined" aria-label="delete">
              <Fab style={{ float: "right" }} size="small" color="secondary" aria-label="delete"
                onClick={submitDelete}
              >
                <DeleteIcon />
              </Fab>
            </Tooltip>
            <br />
            <FormControl component="fieldset">
              <FormGroup aria-label="position" >
                <table width="100%" style={{ width: "100%" }} >
                  {userdefItems.map((option) => (
                    <>
                      <tr>
                        <td>
                          <FormControlLabel
                            disabled={state.transStat == "Verified" || state.readonly == true ? true : false}
                            onChange={handleUserdef}
                            value={option.id}
                            checked={option.checked}
                            control={<Checkbox color="primary" />}
                            label={option.item_name}
                            labelPlacement="end"
                          />
                        </td>
                        <td>
                          <div data-id={option.id} data-name={option.item_name} data-remark={option.remarks} onClick={handleClickOpen}>
                            <Button data-id={option.id} data-name={option.item_name} data-remark={option.remarks} onClick={handleClickOpen} >
                              <CreateIcon data-id={option.id} data-name={option.item_name} data-remark={option.remarks} onClick={handleClickOpen} />
                            </Button>
                          </div>
                        </td>
                      </tr>
                    </>
                  ))}
                </table>
              </FormGroup>
            </FormControl>
          </form>

          <form
            onSubmit={submitupdate}
            autoComplete="off">
            <Dialog open={open} onClose={handleClose} aria-labelledby="form-dialog-title">
              <DialogTitle id="form-dialog-title">Update user defined service</DialogTitle>
              <DialogContent>
                <TextField
                  onChange={onChange}
                  name="selName"
                  defaultValue={state.selName}
                  margin="dense"
                  id="item_name"
                  label="Particular"
                  type="text"
                  fullWidth
                />
                <TextField
                  name="selRemark"
                  onChange={onChange}
                  defaultValue={state.selRemark}
                  margin="dense"
                  id="remarks"
                  label="Remarks"
                  type="text"
                  fullWidth
                />
              </DialogContent>
              <DialogActions>
                <Button onClick={handleClose} color="primary">
                  Cancel
                </Button>
```

```
                        <Button
                            onClick={submitupdate}
                            type="submit" color="primary">
                            Save
                        </Button>
                        </DialogActions>
                    </Dialog>
                </form>

            </>
        );
    }

    return (<>
        <center>
            <FormLabel variant="h6" gutterBottom>
                Service Selection
            </FormLabel>
        </center>
        <hr />
        <FormLabel component="legend">{state.package_name +
" Package"}</FormLabel>
        <br />
        <Accordion>
            <AccordionSummary
                expandIcon={<ExpandMoreIcon />}
                aria-controls="panel1a-content"
                id="panel1a-header"
            >
                <Typography>System Defined Items</Typography>
            </AccordionSummary>
            <AccordionDetails>
                {systemDefine()}
            </AccordionDetails>
        </Accordion>
        {
            state.package_name == "Customize" ?
                <Accordion>
                    <AccordionSummary
                        expandIcon={<ExpandMoreIcon />}
                        aria-controls="panel1a-content"
                        id="panel1a-header"
                    >
                        <Typography>User Defined Items</Typography>
                    </AccordionSummary>
                    <AccordionDetails>
                        {userDefine()}
                    </AccordionDetails>
                </Accordion>
                : <></>
        }
        <br />
        {
            state.role != 'Production'?
            <div style={{ float: "right" }}>
                <Link to={{
                    pathname: `/form/order/costing`,
                    state: {
                        id: location.state.id,
                        type: "update",
                        readonly: state.readonly
                    }
                }} >
                    <Button
                        variant="contained"
                        color="primary"
                        size="medium"
                        startIcon={<NavigateNextIcon />}
                    >
                        Next
                            </Button>
                </Link>
            </div>
            :<></>
        }
        <div style={{ float: "right" }}>
              
            </div>
        <Link to={{
            pathname: `/form/order/basic_details`,
            state: {

                id: location.state.id,
                type: "update",
                readonly: state.readonly
            }
        }} >
            <Button
                style={{ float: "right" }}
                variant="contained"
                color="primary"
                size="medium"
                startIcon={<NavigateBeforeIcon />}
            >
                Back
                </Button>
        </Link>
    </>)
}
const ordprops = {
    cont: content(),
    type: "form"
}
return (
    <>
        <OrderFormContainer                type={location.state.type}
ordprops={ordprops} />
        <br /><br />
    </>
);
}
const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(ServiceSelc);
```

D. Add Userdefined Items

```
import React, { useState } from 'react';
import { useLocation } from 'react-router-dom';
import { connect } from 'react-redux';
import Button from '@material-ui/core/Button';
import Fab from '@material-ui/core/Fab';
import AddIcon from '@material-ui/icons/Add';
import Tooltip from '@material-ui/core/Tooltip';
import { useSnackbar } from 'notistack';
import Http from '../../../Http';
import TextField from '@material-ui/core/TextField';
import Dialog from '@material-ui/core/Dialog';
import DialogActions from '@material-ui/core/DialogActions';
import DialogContent from '@material-ui/core/DialogContent';
import DialogTitle from '@material-ui/core/DialogTitle';
var _isMounted = false;
var variant;
function UserDef(props) {
    _isMounted = true;
    const location = useLocation();
    const [open, setOpen] = React.useState(false);

    const handleClickOpen = () => { setOpen(true); };
    const handleClose = () => { setOpen(false); };

    const { enqueueSnackbar } = useSnackbar();
    const [state, setState] = useState({
        item_name: null,
        remarks: null,
        timeout: 0,
    });

    const handleSave = (e) => {
        e.preventDefault();
        const posts = {
            item_name: state.item_name,
            remarks: state.remarks,
            order_id: location.state.id,
        }

        if (state.item_name && state.remarks) {

            Http.post(`/api/v1/order/addUserdefinedItems`, posts)
                .then((response) => {

                    var variant = 'success';
                    props.newUd()
                    enqueueSnackbar('Service added', { variant })
                })
                .catch((error) => {
                    var variant = 'error';
                    console.log(error);
                    enqueueSnackbar('Failed to save service', { variant })
                });
```

```
    } else {
      variant = 'error';
      enqueueSnackbar('Incomplete data', { variant })
    }
  };

  function onChange(event) {//Sets the value of states
    const { name, value } = event.target;

    if (_isMounted) {
      setState(prevState => ({ ...prevState, [name]: value }));
    }
  }

  return (
    <>
      <div style={{ float: "right" }} >
          
        <Tooltip title="Add user defined" aria-label="add">
          <Fab size="small" color="primary" aria-label="add"
onClick={handleClickOpen}>
            <AddIcon />
          </Fab>
        </Tooltip>

        <form onSubmit={handleSave} autoComplete="off">
          <Dialog open={open} onClose={handleClose} aria-
labelledby="form-dialog-title">
            <DialogTitle id="form-dialog-title">Add new user
defined services</DialogTitle>
            <DialogContent>

              <TextField
                onChange={onChange}
                name="item_name"

                margin="dense"
                id="item_name"
                label="Particular"
                type="text"
                fullWidth
              />
              <TextField
                name="remarks"
                onChange={onChange}

                margin="dense"
                id="remarks"
                label="Remarks"
                type="text"
                fullWidth
              />
            </DialogContent>
            <DialogActions>
              <Button onClick={handleClose} color="primary">
                Cancel
              </Button>
              <Button onClick={handleSave} type="submit"
color="primary">
                Save
              </Button>
            </DialogActions>
          </Dialog>
        </form>
      </div>
    </>
  );
}

const mapStateToProps = (state) => ({
  isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(UserDef);
```

E. Service Costing

```
import React, { useState, useEffect } from 'react';
import { useLocation } from 'react-router-dom';
import { connect } from 'react-redux';
import { useSnackbar } from 'notistack';
import TextField from '@material-ui/core/TextField';
import Button from '@material-ui/core/Button';
import Http from '../../../Http';
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableContainer from '@material-ui/core/TableContainer';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import Paper from '@material-ui/core/Paper';
import update from 'immutability-helper';
var _isMounted = false;
function ServCost(props) {
  const location = useLocation();
  _isMounted = true;

  const { enqueueSnackbar } = useSnackbar();
  const [state, setState] = useState({
    items: [],
    package_id: null,
    package_name: null,
    selected: [],
    timeout: 0,


  });

  useEffect(() => {
    Http.post('/api/v1/order/item/all', { order_id: location.state.id })
      .then((response) => {


        var sel = response.data.items
        var items = [];

        sel.map((option) => {
          var it = {}

          it.id = option.id
          it.item_id = option.item_id
          it.checked = true
          it.particular = option.particular
          it.original_price = option.original_price
          it.markup = option.markup
          items.push(it);
        })


        if (_isMounted) {
          setState(prevState => ({ ...prevState, items: items }));
        }
      })
      .catch((error) => {
        var variant = 'error';
        console.log(error);
        enqueueSnackbar('Failed to fetch items', { variant })
      });

  }, []);

  function submit(e) {
    e.preventDefault();
    const posts = {
      order_id: location.state.id,
      items: JSON.stringify(state.items),
    }

    Http.post('/api/v1/order/item/costing', posts)
      .then((response) => {

        var variant = 'success';
        enqueueSnackbar('Costing saved', { variant })
      })
      .catch((error) => {
        var variant = 'error';
        console.log(error);
        enqueueSnackbar('Failed to save costing', { variant })
      });
  }

  function modServices(e) {

    const { ordid } = e.target.dataset;
    const { name, value } = e.target;
    console.log(value)
    console.log(ordid)
    if (state.timeout) clearTimeout(state.timeout);
    state.timeout = setTimeout(() => {

      var items = state.items;
      var commentIndex = items.findIndex(function (c) {
        return c.id == ordid;
```

```
        });

        var updatedComment = update(items[commentIndex], { [name]: { $set: parseFloat(value) } });

        var newData = update(items, {
            $splice: [[commentIndex, 1, updatedComment]]
        });
        if (_isMounted) {
            setState(prevState => ({ ...prevState, items: newData }));
        }

    }, 300);

}

function content() {

    var items = state.items;
    var total_org_price_inc = 0;
    var total_markup = 0;
    var sale_price = 0;
    items.map((opt) => {

        total_org_price_inc += Number.isInteger(opt.original_price) ? opt.original_price : 0
        total_markup += Number.isInteger(opt.markup) ? opt.markup : 0
    })

    sale_price = parseFloat(total_org_price_inc + total_markup);

    return (<>
        <div style={{ width: "100%" }}>

            <div style={{ width: "100%" }} >
                <form onSubmit={submit} autoComplete="off">

                    <TableContainer component={Paper}>
                        <Table aria-label="simple table" aria-label="caption table">
                            <caption>Service Inclusion Costing</caption>
                            <TableHead>
                                <TableRow>
                                    <TableCell align="center" colSpan="3">Service Costing</TableCell>
                                </TableRow>
                                <TableRow>
                                    <TableCell><TextField id="outlined-basic" value={sale_price} disabled style={{ width: "45%" }} size="small" label="Sale Price" variant="outlined" /></TableCell>
                                    <TableCell align="center"><TextField id="outlined-basic" value={total_org_price_inc} disabled style={{ width: "45%" }} size="small" label="Total Cost" variant="outlined" /></TableCell>
                                    <TableCell align="center"><TextField id="outlined-basic" value={total_markup} disabled style={{ width: "45%" }} size="small" label="Total Mark Up" variant="outlined" /></TableCell>
                                </TableRow>
                                <TableRow>
                                    <TableCell>Particular</TableCell>
                                    <TableCell align="center">Original Price</TableCell>
                                    <TableCell align="center">Mark up</TableCell>
                                </TableRow>
                            </TableHead>
                            <TableBody>
                                {items.map((opt) => (
                                    <TableRow key={opt.item_id}>
                                        <TableCell component="th" scope="row">
                                            {opt.particular}
                                        </TableCell>
                                        <TableCell align="center"><TextField id="outlined-basic" inputProps={{ 'data-ordid': opt.id }} onChange={modServices} name="original_price" data-ordid={opt.id} defaultValue={opt.original_price} style={{ width: "40%" }} size="small" label="Org. Price" variant="outlined" /></TableCell>
                                        <TableCell align="center"><TextField id="outlined-basic" name="markup" inputProps={{ 'data-ordid': opt.id }} onChange={modServices} defaultValue={opt.markup} style={{ width: "40%" }} size="small" label="Mark Up" variant="outlined" /></TableCell>
                                    </TableRow>
                                ))}
                            </TableBody>
                        </Table>
                    </TableContainer>
                    <br />
                    {props.readonly != true ? <Button
                        style={{ float: "right" }}
                        variant="contained"
                        color="primary"
                        size="medium"
                        type="submit"
                        disabled={props.status == "Verified" ? true : false}
                    >
                        Save
                    </Button> : <></>}
                </form>
            </div>
        </div>
    </>)
}
return (
    <>
        {content()}
    </>
);
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(ServCost);
```

F. Costing Container

```
import React , { useState, useEffect } from 'react';
import { useLocation } from 'react-router-dom';
import { connect } from 'react-redux';
import OrderTax from './order_tax';
import OrderDisc from './order_discount';
import OrderFormContainer from '../orders/order_form_container';
import ServiceCosting from './service_costing';
import { useSnackbar } from 'notistack';
import Opex from './order_operating_expense';
import Button from '@material-ui/core/Button';
import NavigateBeforeIcon from '@material-ui/icons/NavigateBefore';
import { Link } from 'react-router-dom';
import FormLabel from '@material-ui/core/FormLabel';
import NavigateNextIcon from '@material-ui/icons/NavigateNext';
import Accordion from '@material-ui/core/Accordion';
import AccordionSummary from '@material-ui/core/AccordionSummary';
import AccordionDetails from '@material-ui/core/AccordionDetails';
import Typography from '@material-ui/core/Typography';
import ExpandMoreIcon from '@material-ui/icons/ExpandMore';
import Http from '../../../Http';

var _isMounted = false;

function AllCats(props) {
    const { enqueueSnackbar } = useSnackbar();
  var variant;
    const location = useLocation();
    _isMounted = true;

    useEffect(() => {
        basicDetails();
    }, []);
    const [state, setState] = useState({
        status: "Verified",
    });

    function basicDetails() {
        Http.post('/api/v1/order/specific', { id: location.state.id })
            .then((response) => {


                var data = response.data.order;
                if (_isMounted) {
                    setState(prevState => ({
                        ...prevState,
                        status: data[0].transaction_status,
                    }));
```

```
            }
        })
        .catch((error) => {
            var variant = 'error';
            console.log(error);
            enqueueSnackbar('Failed to fetch details', { variant })
        });
    }

    function setforverf() {
        const posts = {
            order_id: location.state.id,
        }


if(confirm("Are you sure you want to set this order for verfifcation?")){
        Http.post('/api/v1/order/forVerifyOrder', posts)
            .then((response) => {

                var variant = 'success';
                enqueueSnackbar('Order successfully created', { variant
})
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to update order', { variant })
            });}

    }



    function content() {

        return (<>

            <center>

                <FormLabel variant="h6" >
                    Costing

                </FormLabel>
                <br/>
                <br/>
                <Accordion>
                    <AccordionSummary
                    expandIcon={<ExpandMoreIcon />}
                    aria-controls="panel1a-content"
                    id="panel1a-header"
                    >
                        <Typography>Service Costing</Typography>
                    </AccordionSummary>
                    <AccordionDetails>
                        <ServiceCosting                    status={state.status}
readonly={location.state.readonly} />
                    </AccordionDetails>
                </Accordion>
            </center>
            <br/>
        <Accordion>
                    <AccordionSummary
                    expandIcon={<ExpandMoreIcon />}
                    aria-controls="panel1a-content"
                    id="panel1a-header"
                    >
                        <Typography>Order                              Operating
Expenses</Typography>
                    </AccordionSummary>
                    <AccordionDetails>
                            <Opex                              status={state.status}
readonly={location.state.readonly} />
                    </AccordionDetails>
        </Accordion>
        <br/>
        <OrderTax       id={location.state.id}       status={state.status}
readonly={location.state.readonly}/>  
        <OrderDisc       id={location.state.id}       status={state.status}
readonly={location.state.readonly}/>

         <br/>
         <br/>
             <Link to={{
                 pathname: `/form/order/service_selection`,
                 state: {
                     id: location.state.id,
                     type: "update",
                     readonly: location.state.readonly
                 }
             }} >
                 <Button
                     style={{ float: "left" }}
                     variant="contained"
                     color="primary"
                     size="medium"
                     startIcon={<NavigateBeforeIcon />}
                 // onClick={submit}
                 >
                     Back
                 </Button>




                 </Link>
                 <div style={{ float: "right" }}>

                     {/* {
                     state.status != "Verified" ?
                     <>
                     <Button
                         onClick={setforverf}
                         variant="contained"
                         color="primary"
                         size="medium"
                     >
                         Mark for verification
                             </Button>
                                
                             </> : <>
</>                              } */}
                     <Link to={{
                         pathname: `/form/order/for_verification`,
                         state: {
                             // package_id: state.package_id,
                             id: location.state.id,
                             type: "update",
                             readonly: location.state.readonly
                         }
                     }} >

                         <Button

                             variant="contained"
                             color="primary"
                             size="medium"
                             startIcon={<NavigateNextIcon />}
                         >
                             Next
                                 </Button>
                     </Link>
                 </div>
             </>)
    }


    const ordprops = {
        cont: content(),
        type: "multi_form"
    }

    return (
        <>
            <OrderFormContainer                type={location.state.type}
ordprops={ordprops} />



            <br /><br />
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllCats);
```

G. Add Operating Expenses

```
import React, { useState } from 'react';
import { useLocation } from 'react-router-dom';
import { connect } from 'react-redux';
import Button from '@material-ui/core/Button';
import Fab from '@material-ui/core/Fab';
import AddIcon from '@material-ui/icons/Add';
import Tooltip from '@material-ui/core/Tooltip';
import { useSnackbar } from 'notistack';
import Http from '../../../Http';
import TextField from '@material-ui/core/TextField';
import Dialog from '@material-ui/core/Dialog';
import DialogActions from '@material-ui/core/DialogActions';
import DialogContent from '@material-ui/core/DialogContent';
import DialogTitle from '@material-ui/core/DialogTitle';
var _isMounted = false;
var variant;
function AddOpex(props) {
    _isMounted = true;
    const location = useLocation();
    const [open, setOpen] = React.useState(false);

    const handleClickOpen = () => {
        setOpen(true);
    };

    const handleClose = () => {
        setOpen(false);
    };

    const { enqueueSnackbar } = useSnackbar();
    const [state, setState] = useState({
        particular: null,
        amount: null,
        timeout: 0,
    });

    const handleSave = (e) => {
        e.preventDefault();
        const posts = {
            particular: state.particular,
            amount: state.amount,
            order_id: location.state.id,
        }

        if (state.particular && state.amount) {

            Http.post(`/api/v1/order/opex`, posts)
                .then((response) => {

                    var variant = 'success';
                    props.newOpex(response.data.id)
                    enqueueSnackbar('Costing saved', { variant })
                })
                .catch((error) => {
                    var variant = 'error';
                    console.log(error);
                    enqueueSnackbar('Failed to save costing', { variant })
                });

        } else {
            variant = 'error';
            enqueueSnackbar('Empty OpEx', { variant })
        }
        setOpen(false);
    };

    function onChange(event) {//Sets the value of states
        const { name, value } = event.target;

        if (_isMounted) {
            setState(prevState => ({ ...prevState, [name]: value }));
        }
    }

    return (
        <>
            <Tooltip title="Add OpEx" aria-label="add">
                <Fab size="medium" color="primary" aria-label="add"
onClick={handleClickOpen}>
                    <AddIcon />
                </Fab>
            </Tooltip>
            <form onSubmit={handleSave} autoComplete="off">
                <Dialog open={open} onClose={handleClose} aria-
labelledby="form-dialog-title">
                    <DialogTitle id="form-dialog-title">Add new operating
expense</DialogTitle>
                    <DialogContent>

                        <TextField
                            onChange={onChange}
                            name="particular"

                            margin="dense"
                            id="particular"
                            label="Particular"
                            type="text"
                            fullWidth
                        />
                        <TextField
                            name="amount"
                            onChange={onChange}

                            margin="dense"
                            id="amount"
                            label="Amount"
                            type="number"
                            fullWidth
                        />
                    </DialogContent>
                    <DialogActions>
                        <Button onClick={handleClose} color="primary">
                            Cancel
                        </Button>
                        <Button onClick={handleSave} type="submit"
color="primary">
                            Save
                        </Button>
                    </DialogActions>
                </Dialog>
            </form>
            <br />
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AddOpex);
```

H. Operating Expenses Container

```
import React, { useState, useEffect } from 'react';
import { useLocation } from 'react-router-dom';
import { connect } from 'react-redux';
import { useSnackbar } from 'notistack';
import TextField from '@material-ui/core/TextField';
import Button from '@material-ui/core/Button';
import Fab from '@material-ui/core/Fab';
import Http from '../../../Http';
import AddOpex from './add_opex';
import DeleteIcon from '@material-ui/icons/Delete';
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableContainer from '@material-ui/core/TableContainer';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import Paper from '@material-ui/core/Paper';
import update from 'immutability-helper';
var _isMounted = false;
function Opex(props) {
    const location = useLocation();
    _isMounted = true;

    const { enqueueSnackbar } = useSnackbar();
    const [state, setState] = useState({
        items: [],
        package_id: null,
        package_name: null,
        selected: [],
        timeout: 0,
    });

    function refreshOpex() {
        Http.post('/api/v1/order/item/costing/opex', { order_id:
location.state.id })
```

```
        .then((response) => {

          if (_isMounted) {
            setState(prevState => ({ ...prevState, items: response.data.items }));
          }
        })
        .catch((error) => {
          var variant = 'error';
          console.log(error);
          enqueueSnackbar('Failed to fetch items', { variant })
        });

  }

  useEffect(() => {

    refreshOpex()

  }, []);

  function submit(e) {
    e.preventDefault();
    const posts = {
      order_id: location.state.id,
      items: JSON.stringify(state.items),
    }

    Http.patch(`/api/v1/order/opex/${location.state.id}`, posts)
      .then((response) => {

        var variant = 'success';
        enqueueSnackbar('Costing saved', { variant })
      })
      .catch((error) => {
        var variant = 'error';
        console.log(error);
        enqueueSnackbar('Failed to save costing', { variant })
      });
  }

  function modServices(e) {

    const { ordid } = e.target.dataset;
    const { name, value } = e.target;
    if (state.timeout) clearTimeout(state.timeout);
    state.timeout = setTimeout(() => {

      var items = state.items;
      var commentIndex = items.findIndex(function (c) {
        return c.id == ordid;
      });

      var updatedComment = update(items[commentIndex], { [name]: { $set: name == "particular" ? value : parseFloat(value) } });

      var newData = update(items, {
        $splice: [[commentIndex, 1, updatedComment]]
      });
      if (_isMounted) {
        setState(prevState => ({ ...prevState, items: newData }));
      }

    }, 300);

  }

  function deleteItem(key) {
    var variant;
    const itm = state.items;
    if (confirm('Are you sure you want to proceed?')) {

      Http.delete(`/api/v1/order/opex/${key}`)
        .then((response) => {
          if (response.status === 204) {
            const index = itm.findIndex(
              (item) => parseInt(item.id, 10) === parseInt(key, 10),
            );
            const update = [...itm.slice(0, index), ...itm.slice(index + 1)];
            if (_isMounted) {

              if (_isMounted) {
                setState({ items: update })
              }
            }
          }
          variant = 'success'
          enqueueSnackbar('OpEx successfully deleted', { variant })
        })
        .catch((error) => {
          console.log(error);
          variant = 'error'
          enqueueSnackbar('Error deleting OpEx', { variant })
        });

    }

  }

  function content() {

    var items = state.items;
    var total_amount = 0;
    items.map((opt) => {
      total_amount += Number.isInteger(opt.amount) ? opt.amount : 0
    })

    return (<>

      <div style={{ width: "100%" }}>
        <div style={{ width: "100%" } } >
          <form onSubmit={submit} autoComplete="off">
            {props.readonly != true ? <AddOpex newOpex={refreshOpex} /> : <></>}
            <TableContainer component={Paper}>
              <Table aria-label="simple table" aria-label="caption table">

                <TableHead>
                  <TableRow>
                    <TableCell align="center" colSpan="4">Order Operating Expenses</TableCell>
                  </TableRow>
                  <TableRow>
                    <TableCell></TableCell>
                    <TableCell align="center"><TextField id="outlined-basic" value={total_amount} disabled style={{ width: "45%" }} size="small" label="Total Cost" variant="outlined" /></TableCell>
                    <TableCell align="center"></TableCell>
                  </TableRow>
                  <TableRow>
                    <TableCell>Particular</TableCell>
                    <TableCell align="center">Original Price</TableCell>
                    <TableCell align="center"></TableCell>
                  </TableRow>
                </TableHead>
                <TableBody>
                  {items.map((opt) => (
                    <TableRow key={opt.id}>
                      <TableCell component="th" scope="row">
                        <TextField id="outlined-basic" inputProps={{ 'data-ordid': opt.id }} disabled={opt.particular == "Commission" ? true : false} onChange={modServices} name="particular" data-ordid={opt.id} defaultValue={opt.particular} style={{ width: "40%" }} size="small" label="Particular" variant="outlined" />
                        {/* {opt.particular} */}
                      </TableCell>
                      <TableCell align="center"><TextField id="outlined-basic" inputProps={{ 'data-ordid': opt.id }} onChange={modServices} name="amount" data-ordid={opt.id} defaultValue={opt.amount} style={{ width: "40%" }} size="small" label="Org. Price" variant="outlined" /></TableCell>
                      <TableCell align="center">
                        {opt.particular == "Commission" ?
                          <></>
                          : <Fab size="small" color="primary" aria-label="add" onClick={() => deleteItem(opt.id)} >
                            <DeleteIcon />
                          </Fab>
                        }
                      </TableCell>
                    </TableRow>
                  ))}
                </TableBody>
              </Table>
```

```
                </TableContainer>
                <br />
                {props.readonly != true ?
                    <Button
                        style={{ float: "right" }}
                        variant="contained"
                        color="primary"
                        size="medium"
                        type="submit"
                        disabled={props.status == "Verified" ? true : false}
                    >
                        Save
                    </Button>
                    : <></>}

            </form>
        </div>
    </div>
    </>)
  }

  return (
    <>
      {content()}
    </>
  );
}

const mapStateToProps = (state) => ({
  isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(Opex);
```

I. Order Tax

```
import React, { useState, useEffect } from 'react';
import { connect } from 'react-redux';
import { useSnackbar } from 'notistack';
import Http from '../../../Http';
import TextField from '@material-ui/core/TextField';
import InputAdornment from '@material-ui/core/InputAdornment';

var _isMounted = false;
var variant;
function OrderTax(props) {
  _isMounted = true;

  const { enqueueSnackbar } = useSnackbar();
  const [state, setState] = useState({
    tax_rate: null,
    timeout: 0,
  });

  function onChange(event) {//Sets the value of states
    const { name, value } = event.target;

    if (_isMounted) {
      setState(prevState => ({ ...prevState, [name]: value }));
    }
  }

  useEffect(() => {
    Http.post('/api/v1/order/specific', { id: props.id })
      .then((response) => {
        var data = response.data.order;
        if (_isMounted) {
          setState(prevState => ({
            ...prevState,
            tax_rate: data[0].tax_rate,
          }));
        }
      })
      .catch((error) => {
        var variant = 'error';
        console.log(error);
        enqueueSnackbar('Failed to fetch packages', { variant })
      });
  }, []);

  function submitUpdate() {
    if (confirm("Are you sure you want to update tax rate?")) {
      Http.patch(`/api/v1/order/${props.id}`, { tax_rate: state.tax_rate })
        .then(({ data }) => {
          variant = 'success';
          enqueueSnackbar('Order successfully updated', { variant })
        })

        .catch(() => {
          variant = 'error';
          enqueueSnackbar('Failed updating Order', { variant })
        });
    }
  }

  return (
    <>
      <TextField id="outlined-basic"
        name="tax_rate"
        onChange={onChange}

        InputLabelProps={
          state.tax_rate ?
            { shrink: true, } :
            {}
        }
        value={state.tax_rate} label="Tax Rate" variant="outlined" InputProps={{
          endAdornment: <InputAdornment position="end">%</InputAdornment>,
        }}
        disabled={props.status == "Verified" || props.readonly == true ? true : false}
        onBlur={submitUpdate}
      /> </>
  );
}

const mapStateToProps = (state) => ({
  isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(OrderTax);
```

J. Order Discount

```
import React, { useState, useEffect } from 'react';
import { useLocation } from 'react-router-dom';
import { connect } from 'react-redux';
import { useSnackbar } from 'notistack';
import Http from '../../../Http';
import TextField from '@material-ui/core/TextField';

var _isMounted = false;
var variant;
function OrderDisc(props) {
  _isMounted = true;

  const { enqueueSnackbar } = useSnackbar();
  const [state, setState] = useState({
    deduction: null,
    timeout: 0,
  });

  function onChange(event) {//Sets the value of states
    const { name, value } = event.target;

    if (_isMounted) {
      setState(prevState => ({ ...prevState, [name]: value }));
    }
  }

  useEffect(() => {
    Http.post('/api/v1/order/specific', { id: props.id })
      .then((response) => {

        var data = response.data.order;
        if (_isMounted) {
          setState(prevState => ({
            ...prevState,
            deduction: data[0].deduction,
          }));
        }
      })
      .catch((error) => {
        var variant = 'error';
```

```
            console.log(error);
            enqueueSnackbar('Failed to fetch packages', { variant })
        });
    }, []);

    function submitUpdate() {
        if (confirm("Are you sure you want to update tax rate?")) {
            Http.patch(`/api/v1/order/${props.id}`, { deduction: state.deduction })
            .then(({ data }) => {
                variant = 'success';
                enqueueSnackbar('Order successfully updated', { variant })
            })

            .catch(() => {
                variant = 'error';
                enqueueSnackbar('Failed updating Order', { variant })
            });
        }
    }

    return (
        <>
            <TextField id="outlined-basic"
                name="deduction"
                onChange={onChange}
                InputLabelProps={
                    state.deduction || state.deduction == 0 ?
                        {
                            shrink: true,
                        } :
                        {}
                } value={state.deduction} label="Discount" variant="outlined"
                disabled={props.status == "Verified" || props.readonly == true ? true : false}
                onBlur={submitUpdate}
            />
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(OrderDisc);
```

K. For Verification

```
import React, { useState, useEffect } from 'react';
import { useLocation } from 'react-router-dom';
import { connect } from 'react-redux';
import PaymentForm from '../payment_form';
import OrderFormContainer from '../orders/order_form_container';
import { useSnackbar } from 'notistack';
import Http from '../../../Http';
import Button from '@material-ui/core/Button';
import NavigateBeforeIcon from '@material-ui/icons/NavigateBefore';
import { Link } from 'react-router-dom';
import FormLabel from '@material-ui/core/FormLabel';
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableContainer from '@material-ui/core/TableContainer';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import Paper from '@material-ui/core/Paper';
import { useReactToPrint } from 'react-to-print';
import { ComponentToPrint } from '../../../pages/tables/components/printTemplate';
import InvoicePrintComp from './invoice_print';

var _isMounted = false;
function ForVerf(props) {
    const location = useLocation();
    _isMounted = true;

    const componentRef = React.useRef();
    const handlePrint = useReactToPrint({
        content: () => componentRef.current,
    });

    const { enqueueSnackbar } = useSnackbar();
    const [state, setState] = useState({
        items: [],
        itemsOpex: [],
        timeout: 0,
        tax_rate: null,
        deduction: null,
        status: null,
        payment_status: null,
        client_id: null,
        loaded: null,
        keyPayForm: 0,
        allBasic: {},
        packageItems: [],
        opex: []
    });

    useEffect(() => {
        setforverf()

        serviceCosting();
        opex();
        basicDetails();
        costingDetails();
        getOpex();


    }, []);

    function setforverf() {
        const posts = {
            order_id: location.state.id,
        }
        Http.post('/api/v1/order/forVerifyOrder', posts)
            .then((response) => {
            })
            .catch((error) => {
                console.log(error);
            });
    }

    function basicDetails() {
        Http.post('/api/v1/order/specific', { id: location.state.id })
            .then((response) => {

                var data = response.data.order;
                console.log("all data")
                console.log(data)
                if (_isMounted) {
                    setState(prevState => ({
                        ...prevState,
                        tax_rate: data[0].tax_rate,
                        deduction: data[0].deduction,
                        status: data[0].transaction_status,
                        payment_status: data[0].payment_status,
                        client_id: data[0].client_id,
                        loaded: true,
                        allBasic: data[0]
                    }));


                }
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch details', { variant })
            });
    }
    function costingDetails() {
        Http.post('/api/v1/order/item/all', { order_id: location.state.id })
            .then((response) => {
                var sel = response.data.items
                if (_isMounted) {
                    setState(prevState => ({ ...prevState, packageItems: sel }));
                }
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch items', { variant })
            });
    }

    function getOpex() {
```

```
        Http.post('/api/v1/order/item/costing/opex', { order_id: location.state.id })
            .then((response) => {

                var opex = response.data.items;
                var totalOpex = 0;
                opex.map((o) => {
                    totalOpex += o.amount
                })

                var opexarr = [];
                var opexspec ={};
                opexspec.name = "Operation Fee";
                opexspec.amount = totalOpex;

                opexarr.push(opexspec);

                if (_isMounted) {
                    setState(prevState => ({ ...prevState, opex: opexarr }));
                }
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch items', { variant })
            });

    }

    function serviceCosting() {
        Http.post('/api/v1/order/item/all', { order_id: location.state.id })
            .then((response) => {
                var sel = response.data.items
                var items = [];

                sel.map((option) => {
                    var it = {}

                    it.id = option.id
                    it.original_price = option.original_price
                    it.markup = option.markup
                    items.push(it);
                })

                if (_isMounted) {
                    setState(prevState => ({ ...prevState, items: items }));
                }

            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch items', { variant })
            });
    }

    function opex() {
        Http.post('/api/v1/order/item/costing/opex', { order_id: location.state.id })
            .then((response) => {

                if (_isMounted) {
                    setState(prevState => ({ ...prevState, itemsOpex: response.data.items }));
                }
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch items', { variant })
            });
    }

    function setverf() {
        const posts = {
            order_id: location.state.id,
        }

        if (confirm("Are you sure you want to verify order?")) {
            Http.post('/api/v1/order/verifyOrder', posts)
                .then((response) => {
                    basicDetails();
                    var variant = 'success';
                    if (_isMounted) {
                        setState(prevState => ({ ...prevState, keyPayForm: state.keyPayForm + 1 }));
                    }
                    enqueueSnackbar('Order successfully created', { variant })
                })
                .catch((error) => {
                    var variant = 'error';
                    console.log(error);
                    enqueueSnackbar('Failed to update order', { variant })
                });
        }
    }

    function generateCalculations(subtotal, payable, tax){
        var calcArr =[];

        var subt = {};
        subt.name = "Subtotal";
        subt.amount = subtotal;
        calcArr.push(subt);

        var disc = {};
        disc.name = "Deduction/Discount";
        disc.amount = state.deduction;
        calcArr.push(disc);

        var tx = {};
        tx.name = "Tax";
        tx.amount = tax
        calcArr.push(tx);

        var amtDue = {};
        amtDue.name = "Amount Due";
        amtDue.amount = payable
        calcArr.push(amtDue);

        return calcArr
    }

    function content() {
        var items = state.items;
        var itemsOp = state.itemsOpex;
        var total_opex = 0;
        var total_org_price_inc = 0;
        var total_markup = 0;
        var sale_price = 0;

        items.map((opt) => {
            total_org_price_inc += Number.isInteger(opt.original_price) ? opt.original_price : 0
            total_markup += Number.isInteger(opt.markup) ? opt.markup : 0
        })

        itemsOp.map((opt) => {
            total_opex += Number.isInteger(opt.amount) ? opt.amount : 0
        })

        sale_price = parseFloat(total_org_price_inc + total_markup);

        // subtotal

        var subtotal = sale_price + total_opex;
        var tax_converted = state.tax_rate / 100;
        var subtotal_discounted = subtotal - state.deduction;

        var subtotal_discounted_tax = subtotal_discounted * tax_converted;
        var payable = (subtotal - state.deduction) + subtotal_discounted_tax;

        var printdata = {
            title: 'Order Invoice',
            content: <InvoicePrintComp basic={state.allBasic} packageItems={state.packageItems} opex={state.opex} calculations={generateCalculations(subtotal, payable, subtotal_discounted_tax.toFixed(2))}/>
        }

        return (<>
            <center>
                <FormLabel variant="h4" >
                    <b>Verification</b>

                </FormLabel>
            </center>
            <hr />
            <FormLabel variant="h6" >
```

```
            Sale price

          </FormLabel>
          <hr />
          <TableContainer component={Paper}>
            <Table    aria-label="simple    table"    aria-label="caption
table">
              <TableBody>
                <TableRow >
                  <TableCell component="th" scope="row">
                    SRP
                  </TableCell>
                  <TableCell component="th" scope="row">
                    {total_org_price_inc}
                  </TableCell>
                </TableRow>
                <TableRow >
                  <TableCell component="th" scope="row">
                    Mark Up
                  </TableCell>
                  <TableCell component="th" scope="row">
                    {total_markup}
                  </TableCell>

                </TableRow>
                <TableRow >
                  <TableCell component="th" scope="row">
                    <b>Total</b>
                  </TableCell>
                  <TableCell component="th" scope="row">
                    {sale_price}
                  </TableCell>
                </TableRow>
              </TableBody>
            </Table>
          </TableContainer>
          <br />
          <FormLabel variant="h6" >
            Operating Expenses
          </FormLabel>
          <hr />
          <TableContainer component={Paper}>
            <Table    aria-label="simple    table"    aria-label="caption
table">
              <TableHead>
                <TableRow>
                  <TableCell><b>Particular</b></TableCell>
                  <TableCell                 align="center"><b>Original
Price</b></TableCell>
                </TableRow>
              </TableHead>
              <TableBody>
                {itemsOp.map((opt) => (
                  <TableRow key={opt.id}>
                    <TableCell component="th" scope="row">
                      {opt.particular}
                    </TableCell>
                    <TableCell
align="center">{opt.amount}</TableCell>
                  </TableRow>
                ))}
                <TableRow >
                  <TableCell component="th" scope="row">
                    <b>Total</b>
                  </TableCell>
                  <TableCell      align="center"      component="th"
scope="row">
                    {total_opex}
                  </TableCell>
                </TableRow>
              </TableBody>
            </Table>
          </TableContainer>
          <br />
          <FormLabel variant="h6" >
            Calculations
          </FormLabel>
          <hr />
          <TableContainer component={Paper}>
            <Table    aria-label="simple    table"    aria-label="caption
table">
              <TableBody>
                <TableRow >
                  <TableCell component="th" scope="row">
                    Subtotal
                  </TableCell>
                  <TableCell component="th" scope="row">
                    {subtotal}
                  </TableCell>
                </TableRow>
                <TableRow >
                  <TableCell component="th" scope="row">
                    Deduction/Discount
                  </TableCell>
                  <TableCell component="th" scope="row">
                    {state.deduction}
                  </TableCell>
                </TableRow>
                <TableRow >
                  <TableCell component="th" scope="row">
                    Tax
                  </TableCell>
                  <TableCell component="th" scope="row">
                    {subtotal_discounted_tax.toFixed(2)}
                  </TableCell>
                </TableRow>
                <TableRow >
                  <TableCell component="th" scope="row">
                    <b>   Amount Due </b>
                  </TableCell>
                  <TableCell component="th" scope="row">
                    {payable}
                  </TableCell>
                </TableRow>
              </TableBody>
            </Table>
          </TableContainer>
          <hr />
          {
            state.loaded == true && location.state.readonly != true ?
              <div key={state.keyPayForm}>
                <PaymentForm
                  amount={state.payment_status      ==      "Partial
Payment" ? null : payable}
                  status={state.payment_status}
                  client_id={state.client_id}
                />
              </div>
              : <></>
          }
          <br />  <br />
          <hr />
          <Link to={{
            pathname: `/form/order/costing`,
            state: {
              id: location.state.id,
              type: "update",
              readonly : location.state.readonly
            }
          }} >
            <Button
              style={{ float: "left" }}
              variant="contained"
              color="primary"
              size="medium"
              startIcon={<NavigateBeforeIcon />}
            >
              Back
            </Button>
          </Link>
          {
          location.state.readonly != true ?
           <div style={{ float: "right" }}>
             {
               state.status  ==  "For  Verification"  ||  state.status  ==
"Verified" ?
                 <Button
                   onClick={state.status == "Verified" ? {} : setverf}
                   variant="contained"
                   color={state.status  ==  "Verified"  ?  "secondary"  :
"primary"}
                   size="medium"
                 >
                   {state.status  ==  "Verified"  ?  "Order  Verified"  :
"Verify Order"}
                 </Button>
                 : <></>
             }
           </div> : <></>}
             
          <Button       variant="contained"       onClick={handlePrint}
color="primary">Print</Button>
          <div  style={{  display:  "none"  }}><ComponentToPrint
data={printdata} ref={componentRef} /></div>
```

```
        </>)
    }

    const ordprops = {
        cont: content(),
        type: "form"
    }

    return (
        <>
            <OrderFormContainer                type={location.state.type}
ordprops={ordprops} />
            <br /><br />
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(ForVerf);
```

L. Invoice Print

```
import React from 'react';
import { useLocation } from 'react-router-dom';
import { connect } from 'react-redux';
import BootstrapTable from 'react-bootstrap-table-next';

var _isMounted = false;
function InvoicePrintComp(props) {
    _isMounted = true;

    function leftHeadCont() {
        return (
            <>
                <tr>
                    <td><b>Name of Client</b></td>
                    <td> : </td>
                    <td>{props.basic.client}</td>
                </tr>
                <tr>
                    <td><b>Date of Order</b></td>
                    <td> : </td>
                    <td>{props.basic.date_order}</td>
                </tr>
                <tr>
                    <td><b>Date Verified</b></td>
                    <td> : </td>
                    <td>{props.basic.date_verified}</td>
                </tr>
            </>
        )
    }

    function rigthHeadCont() {
        return (
            <>
                <tr>
                    <td><b>Ref#</b></td>
                    <td> : </td>
                    <td>{props.basic.code}</td>
                </tr>
                <tr>
                    <td><b>Package Name</b></td>
                    <td> : </td>
                    <td>{props.basic.package}</td>
                </tr>
                <tr>
                    <td><b>Payment Method</b></td>
                    <td> : </td>
                    <td>{props.basic.payment_status}</td>
                </tr>
            </>
        )
    }

    function packageItmSrp(cell, row) {
        return (row.original_price + row.markup)
    }

    function packageItems(){
        const columnsv = [{
            dataField: 'particular',
            text: 'Particular',
            headerStyle: (colum, colIndex) => {
                return {textAlign: 'center' };
            },
        }, {
            dataField: 'original_price',
            text: 'SRP',
            formatter: packageItmSrp,
            headerStyle: (colum, colIndex) => {
                return { textAlign: 'right' };
            },align: (colum, colIndex) => {
                return 'right';
            }
        }, {
            dataField: 'markup',
            text: 'markup',
            hidden: true
        }];

        return (<>
            <div>
            <BootstrapTable
            keyField='id'
            data={props.packageItems}
            columns={columnsv}
            caption={"Package Items"}
            />
            </div>
            </>)
    }

    function otherExp(){
        const columnsv = [{
            dataField: 'name',
            text: 'Particular',
            headerStyle: (colum, colIndex) => {
                return {textAlign: 'center' };
            },
        },{
            dataField: 'amount',
            text: 'Amount',
            headerStyle: (colum, colIndex) => {
                return { textAlign: 'right' };
            },align: (colum, colIndex) => {
                return 'right';
            }
        }];

        return (<>
            <div>
            <BootstrapTable
            keyField='id'
            data={props.opex}
            columns={columnsv}
            caption={"Other Fees"}
            />
            </div>

            </>)
    }

    function calculations(){
        const columnsv = [{
            dataField: 'name',
            text: ' ',
            headerStyle: (colum, colIndex) => {
                return {textAlign: 'center' };
            },
        },{
            dataField: 'amount',
            text: 'Amount',
            headerStyle: (colum, colIndex) => {
                return { textAlign: 'right' };
            },align: (colum, colIndex) => {
                return 'right';
            }
        }];

        return (<>
            <div>
            <BootstrapTable
            keyField='id'
            data={props.calculations}
            columns={columnsv}
            caption={"Calculations"}
            />
            </div>
            </>)
```

```
    }

    return (
        <>
            <table style={{ width: '100%' }}>
                <tr>
                    <td>
                        {leftHeadCont()}
                    </td>
                    <td>
                        {rigthHeadCont()}
                    </td>
                </tr>
            </table>
            <br/>
            {packageItems()}
            {otherExp()}
            {calculations()}
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(InvoicePrintComp);
```

M. Order Container

```
import React from 'react';
import { connect } from 'react-redux';
import GenCont from '../../../components/gen_container';
import Progress from './progress_step';
import AddIcon from '@material-ui/icons/Add';
import EditIcon from '@material-ui/icons/Edit';
import ListAltIcon from '@material-ui/icons/ListAlt';
import { useLocation } from 'react-router-dom';

var _isMounted = false;

function content(c, stat) {
    _isMounted = true;

    const form_padLR = c.type == "form" ? "30%": c.type == "multi_form" ? "5%" : " ";

    return (
        <>
            <Progress  />
            <div style={{paddingLeft: form_padLR, paddingRight: form_padLR, paddingTop: "5%", paddingBottom: "5%"}}>
                {c.cont}
            </div>
        </>
    )
}

function AllOrder(props) {
    const location = useLocation();
    const { type } = location.state;

    var label = "Create New";
    var icon = <AddIcon fontSize='small' />;

    if (type == "update") {
        label = "Update";
        icon = <EditIcon fontSize='small' />;

    }

    const breadcrumps = [
        {
            label: "Orders",
            icon: <ListAltIcon fontSize='small' />,
            route: "/all/orders"
        },
        {
            label: label,
            icon: icon,
            states: { type: type }
        }
    ]

    return (
        <>
            <GenCont title=" " breadcrumps={breadcrumps} content={content(props.ordprops, props.transStat)} />
            <br /><br />
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllOrder);
```

9.2 Controller

A. Order Items Controller

```
<?php

namespace App\Http\Controllers;

use App\Client;
use App\Helper\Helper;
use App\Item;
use App\Order;
use App\OrderItem;
use Illuminate\Http\Request;
use DB;

class OrderItemController extends Controller
{

    public function specificOrderItem(Request $request)
    {
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $or = OrderItem::leftJoin('items', 'order_items.item_id', '=', 'items.id')
                ->where('order_items.order_id', request("order_id"))
                ->select(
                    // 'items.particular',
                    DB::raw('(CASE WHEN order_items.item_id is NULL THEN order_items.item_name ELSE items.particular END) AS particular'),
                    'order_items.*'
                )
                ->get();
        } catch (Exception $e) {
        }

        return response()->json([
            'status' => 200,
            'items' => $or,
        ], 200);
    }

    public function addUserdefinedItems(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {

            $item = OrderItem::create([
                'item_name' => request('item_name'),
                'remarks' => request('remarks'),
                'order_id' => request('order_id')
            ]);
        } catch (Exception $e) {
        }

        return response()->json([
            'status' => 200,
        ], 200);
    }

    public function deleteUserdefinedItems(Request $request)
    {
```

```
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {

            $items = json_decode(request('items'), true); //new

            foreach ($items as $i) {
                OrderItem::where('id', $i['id'])->delete();
            }
        } catch (Exception $e) {
        }

        return response()->json([
            'status' => 200,
        ], 200);
    }

    public function updateUserdefinedItems(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {

            $item = OrderItem::where('id', request("id"))->firstorfail();

            if (request('item_name')) {
                $item->item_name = request('item_name');
            }
            if (request('remarks')) {
                $item->remarks = request('remarks');
            }

            $item->save();
        } catch (Exception $e) {
        }

        return response()->json([
            'status' => 200,
        ], 200);
    }

    public function userdefOrderItem(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $or = OrderItem::whereNull('item_id')
                ->where('order_items.order_id', request("order_id"))
                ->select('order_items.*')
                ->get();
        } catch (Exception $e) {
        }

        return response()->json([
            'status' => 200,
            'items' => $or,
        ], 200);
    }

    public function modify(Request $request)
    {
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {

            $old = OrderItem::where('order_id', request("order_id"))->get();

            OrderItem::where('order_id', request("order_id"))->delete();
            $items = json_decode(request('items'), true);
            foreach ($items as $itm) {

                $getItm = Item::where('id', $itm["item_id"])->firstOrFail();
                $price = $getItm->price;
                $uprice = null;

                foreach ($old as $ol) {
                    if ($itm['item_id'] == $ol['item_id']) {
                        $price = $ol['original_price'];
                        $uprice = $ol['markup'];
                    }
                }


                $item = OrderItem::create([
                    'item_id' => $itm["item_id"],
                    'order_id' => request('order_id'),
                    'original_price' => $price,
                    'markup' => $uprice,

                ]);
            }


            $ote = Order::where('id', request("order_id"))->firstOrFail();
            if ($ote->transaction_status == "Basic Details") {
                $ote->transaction_status = "Service Selection";
            }
            $ote->save();
        } catch (Exception $e) {
            return $this->responseServerError('Error updating resource.');
        }
    }

    public function orderItemCosting(Request $request)
    {
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {

            $items = json_decode(request('items'), true);
            foreach ($items as $itm) {
                $oi = OrderItem::where('id', $itm['id'])->firstOrFail();

                if ($itm['original_price']) {
                    $oi->original_price = $itm['original_price'];
                }
                if ($itm['markup']) {
                    $oi->markup = $itm['markup'];
                }

                $oi->save();
            }

            $ote = Order::where('id', request("order_id"))->firstOrFail();
            if ($ote->transaction_status == "Service Selection") {
                $ote->transaction_status = "Costing";
            }
            $ote->save();
        } catch (Exception $e) {
            return $this->responseServerError('Error updating resource.');
        }
    }

    public function verifyOrder(Request $request)
    {
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {


            $ote = Order::where('id', request("order_id"))->firstOrFail();
            if ($ote->transaction_status == "For Verification") {
                $ote->transaction_status = "Verified";
                $ote->date_verified = date("Y-m-d");
            }

            $var = new Helper();
            $amt = $var->calculateOrderPayable(request("order_id"));

            $client = Client::where('id', $ote->client_id)->firstOrFail();
            $client->balance += $amt;

            $client->save();
            $ote->save();

        } catch (Exception $e) {
```

```
            return $this->responseServerError('Error updating resource.');
        }
    }

    public function forVerifyOrder(Request $request)
    {
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {


            $ote = Order::where('id', request("order_id"))->firstOrFail();
            if ($ote->transaction_status == "Costing") {
                $ote->transaction_status = "For Verification";
            }
            $ote->save();
        } catch (Exception $e) {
            return $this->responseServerError('Error updating resource.');
        }
    }

}
```

B. Order Transaction Controller

```
<?php

namespace App\Http\Controllers;

use App\Order;
use App\OrderItem;
use App\OrderOperatingExpense;
use App\Tax;
use Illuminate\Http\Request;
use DB;

class OrderController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $or =   Order::join('clients', 'orders.client_id', '=', 'clients.id')
                ->join('packages', 'orders.package_id', '=', 'packages.id')
                ->join('order_items', 'orders.id', '=', 'order_items.order_id')
                ->select(
                    'orders.*',
                    'packages.name as package',
                    DB::raw('"" as comm'),
                    DB::raw('CONCAT(clients.first_name," " , clients.last_name) as name'),
                    DB::raw('(SUM(order_items.original_price + order_items.markup))as amount ')

                )
                ->groupBy('orders.code');

            $opex =  Order::join('clients', 'orders.client_id', '=', 'clients.id')
                ->join('packages', 'orders.package_id', '=', 'packages.id')
                ->join('order_operating_expenses', 'orders.id', '=', 'order_operating_expenses.order_id')
                ->select(
                    'orders.*',
                    'packages.name as package',
                    DB::raw('(CASE WHEN order_operating_expenses.particular = "Commission" THEN order_operating_expenses.amount ELSE CAST(6 AS float) END) as comm '),
                    DB::raw('CONCAT(clients.first_name," " , clients.last_name) as name'),

DB::raw('(SUM(order_operating_expenses.amount))as amount '),
                )

                ->groupBy('orders.code');


            $result = $or
                ->union($opex);

            $combined = DB::query()->fromSub($result, 'i_t');
            if (request("type") == "tracker") {
                $combined->where('transaction_status', '=', "Verified");
            }

            if (request('from_date') && request('to_date')) {
                $combined->where('transaction_status', '=', "Verified");
                $combined->whereDate('date_verified', '>=', request('from_date'));
                $combined->whereDate('date_verified', '<=', request('to_date'));
            }

            if (request('from_date_mas') && request('to_date_mas') && request('transaction_status') == "master") {

                $combined->whereDate('date_order', '>=', request('from_date_mas'));
                $combined->whereDate('date_order', '<=', request('to_date_mas'));
            }

            if (request('year')) {
                $combined->whereYear('date_verified', '=', request('year'));
            }

            $combined->select(
                'i_t.*',
                DB::raw('MONTH(i_t.date_verified) as month'),
                DB::raw('SUM(comm) as commission'),
                DB::raw('
            (
                (
                 (SUM(CASE WHEN amount is null THEN 0 Else amount END)) -
                 (CASE WHEN deduction is null THEN 0 Else deduction END)
                )
                   +
                (
                     (
                      (SUM(CASE WHEN amount is null THEN 0 Else amount END)) -
                      (CASE WHEN deduction is null THEN 0 Else deduction END)
                     )
                     *
                     (
                        tax_rate / 100
                     )
                )

             ) as amount_due '),
                DB::raw('
            (
                (
                    (
                      (SUM(CASE WHEN amount is null THEN 0 Else amount END)) -
                      (CASE WHEN deduction is null THEN 0 Else deduction END)
                    )
                       +
                    (
                        (
                          (SUM(CASE WHEN amount is null THEN 0 Else amount END)) -
                          (CASE WHEN deduction is null THEN 0 Else deduction END)
                        )
                        *
                        (
                           tax_rate / 100
                        )
                    )

                ) - SUM(comm)
             ) as net ')
```

```
            );
            $combined->groupBy('code');

            $combined =   $combined->get();
        } catch (Exception $e) {
        }

        return response()->json([
            'status' => 200,
            'orders' => $combined
        ], 200);
    }

    public function specificOrder(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $or  =  Order::join('packages',  'orders.package_id',  '=',
'packages.id')
                ->join('clients', 'orders.client_id', '=', 'clients.id')
                ->select(
                    'orders.*',
                    'packages.name as package',
                    DB::raw('CONCAT(clients.first_name,"  "  ,
clients.last_name) as client'),
                )
                ->where('orders.id', request("id"))
                ->get();
        } catch (Exception $e) {
        }

        return response()->json([
            'status' => 200,
            'order' => $or,
        ], 200);
    }


    public function ImportOrder(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }
        try {
            $clientId = null;
            $servs =[];
            if (!is_null(request("orderDetails"))) {
                $ordDet  =  json_decode(request('orderDetails'),  true);
//new
                $ordServ  =  json_decode(request('ordServices'),  true);
//new
                $ordExp = json_decode(request('ordExp'), true); //new
                $tax = Tax::select('taxes.*')->firstOrFail();

                foreach ($ordDet as $itm) {
                    if ($itm["order_id"]) {
                        $clientId                                        =
ClientController::checkCreateClient(strtolower($itm["client_firstnam
e"]),                                strtolower($itm["client_middlename"]),
strtolower($itm["client_lastname"]));
                        $packageId                                       =
PackageController::checkCreatePackage(strtolower($itm["package
_name"]));
                        $paymeth  =  (strtolower($itm["payment_method"])
== "full payment" ? "Full Payment" : "Partial Payment");

                        $code =  "ORD-IMP".rand(1111111, 9999999);

                        while (!is_null($check_order = Order::where('code',
$code)->first())) {
                            $code =  "ORD-IMP".rand(1111111, 9999999);
                        }

                        $ord = Order::create([
                            'client_id' =>  $clientId,
                            'package_id' => $packageId,
                            'payment_status' => $paymeth,
                            'tax_rate' => $itm["tax"] ? $itm["tax"] : $tax->rate,
                            'transaction_status' => "Verified",
                            'date_order'                                 =>
date_format(date_create($itm["date_order"]),"Y-m-d"),
                            'date_verified'                              =>
date_format(date_create($itm["date_verified"]),"Y-m-d") ,
                            'deduction' => $itm["discount"],
                            'isImported' => 1,
                            'code' => $code,

                        ]);


                        $servs  =  array_keys(array_column($ordServ,
'order_id'), $itm["order_id"]);

                        foreach($servs as $ss){
                            $obs = $ordServ[$ss];
                            $item = OrderItem::create([
                                'item_id'                                =>
ItemController::checkCreateItem($obs["service_name"]),
                                'order_id' => $ord->id,
                                'original_price' => $obs["service_org_price"],
                                'markup' =>  $obs["service_markup"],
                            ]);
                        }


                        $exps  =  array_keys(array_column($ordExp,
'order_id'), $itm["order_id"]);

                        foreach($exps as $es){
                            $eps = $ordExp[$es];
                            $opex = OrderOperatingExpense::create([
                                'particular' => $eps["particular"],
                                'amount' => $eps["cost"],
                                'order_id' => $ord->id
                            ]);
                        }


                    }
                }
            }

            return response()->json([
                'status' => 200,
                'clientId' => $clientId,
                'packageId' =>  $packageId,
                'paymeth' =>  $paymeth,
                'ord' =>  $ord,
                'servs' => $servs,

            ], 200);
        } catch (Exception $e) {
            return  $this->responseServerError('Error  creating
resource.');
        }
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }
        try {
            $tax = Tax::select('taxes.*')->firstOrFail();


            $code  =  "ORD"  .  date("y")  .  date("m")  .  date("d")  .
rand(11111, 99999);

            while (!is_null($check_order = Order::where('code', $code)-
>first())) {
                $code  =  "ORD"  .  date("y")  .  date("m")  .  date("d")  .
rand(11111, 99999);
            }

            $ord = Order::create([
                'client_id' => request('client_id'),
                'package_id' => request('package_id'),
                'payment_status' => request('payment_status'),
                'tax_rate' => $tax->rate,
```

```
            'transaction_status' => "Basic Details",
            'date_order' => date("Y-m-d"),
            'code' => $code,

        ]);

        $opex = OrderOperatingExpense::create([
            'particular' => "Commission",
            'amount' => 0,
            'order_id' => $ord->id
        ]);

        return response()->json([
            'status' => 201,
            'id' => $ord->id,
            'code' => $code
        ], 201);
    } catch (Exception $e) {
        return $this->responseServerError('Error creating resource.');
    }
}


/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @param  \App\Order  $order
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    // Get user from $request token.
    if (!$user = auth()->setRequest($request)->user()) {
        return $this->responseUnauthorized();
    }

    try {
        $ote = Order::where('id', $id)->firstOrFail();

        if (request('client_id')) {
            $ote->client_id = request('client_id');
        }

        if (request('package_id')) {
            $ote->package_id = request('package_id');
        }

        if (request('payment_status')) {
            $ote->payment_status = request('payment_status');
        }

        if (request('tax_rate')) {
            $ote->tax_rate = request('tax_rate');
        }

        if (request('deduction')) {
            $ote->deduction = request('deduction');
        }

        $ote->save();

        return response()->json([
            'status' => 200,
        ], 200);
    } catch (Exception $e) {
        return $this->responseServerError('Error updating resource.');
    }
}

/**
 * Remove the specified resource from storage.
 *
 * @param  \App\Order  $order
 * @return \Illuminate\Http\Response
 */
public function destroy(Request $request, $id)
{
    // Get user from $request token.
    if (!$user = auth()->setRequest($request)->user()) {
        return $this->responseUnauthorized();
    }

    try {
        $order = Order::where('id', $id)->firstOrFail();

        if ($order->transaction_status == "Verified") {
            return response()->json(['status' => 400,], 400);
        } else {
            $orderitem = OrderItem::where('order_id', $id)->delete();
            $orderoe = OrderOperatingExpense::where('order_id', $id)->delete();

            $order->delete();
        }
    } catch (Exception $e) {
        return $this->responseServerError('Error deleting resource.');
    }
}
}
```

10. Payments

10.1 View

```
import React, { useState, useEffect } from 'react';
import { connect } from 'react-redux';
import GenCont from '../../../components/gen_container';
import Fab from '@material-ui/core/Fab';
import AddIcon from '@material-ui/icons/Add';
import { Link } from 'react-router-dom';
import Tooltip from '@material-ui/core/Tooltip';
import Http from '../../../Http';
import Button from '@material-ui/core/Button';
import PaymentForm from '../../forms/payment_form';
import BootstrapTable from 'react-bootstrap-table-next';
import 'react-bootstrap-table-next/dist/react-bootstrap-table2.min.css';
import 'react-bootstrap-table2-toolkit/dist/react-bootstrap-table2-toolkit.min.css';
import { useSnackbar } from 'notistack';
import ListAltIcon from '@material-ui/icons/ListAlt';
import Dialog from '@material-ui/core/Dialog';
import TextField from '@material-ui/core/TextField';
import MenuItem from '@material-ui/core/MenuItem';
import ReactTable from '../components/reactTable';
import DateRangeGen from '../../../components/daterangeGen';
import { useReactToPrint } from 'react-to-print';
import { ComponentToPrint } from '../components/printTemplate';
var _isMounted = false;

function content() {
    _isMounted = true;

    const { enqueueSnackbar } = useSnackbar();

    const componentRef = React.useRef();
    const handlePrint = useReactToPrint({
        content: () => componentRef.current,
    });

    var localTerminal = JSON.parse(localStorage.getItem("payments") || "[]");
    var localTerminalClients = JSON.parse(localStorage.getItem("clients") || "[]");
    const [state, setState] = useState({ data: localTerminal, allclients: localTerminalClients, loaded: null, client_id: null });


    useEffect(() => {
        getAllPAyments();
    }, []);

    function getAllPAyments (post = {}){
        Http.post('/api/v1/payment/index',post)
        .then((response) => {
            localStorage.setItem('payments', JSON.stringify(response.data.payments))
            localTerminal = JSON.parse(localStorage.getItem("payments") || "[]");
            if (_isMounted) {
                setState(prevState => ({ ...prevState, data: localTerminal }));
            }
        })
        .catch((error) => {
```

```
            var variant = 'error';
            console.log(error);
            enqueueSnackbar('Failed to fetch payments', { variant })
        });
    }

    useEffect(() => {
        Http.get('/api/v1/client')
            .then((response) => {
                localStorage.setItem('clients',
JSON.stringify(response.data.clients))
                var localTerminalcl =
JSON.parse(localStorage.getItem("clients") || "[]");
                if (_isMounted) {
                    setState(prevState => ({ ...prevState, allclients:
localTerminalcl, loaded: true }));
                }
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch clients', { variant })
            });
    }, []);

    function onChange(event) {//Sets the value of states
        const { name, value } = event.target;
        if (_isMounted) {
            setState(prevState => ({ ...prevState, [name]: value }));
        }
    }
    function newPayment(){
        getAllPAyments();
    }

    const [open, setOpen] = React.useState(false);

    const handleClickOpen = () => {
        setOpen(true);
    };
    const handleClose = () => {
        setOpen(false);
    };
    const allclients = state.allclients
    function addNewPayment() {
        return (
            <>
                <div style={{ width: '300px' }}>
                    <label>Add new payment</label>

                    {
                        state.loaded == true ?
                            <TextField

                                id="outlined-select-currency"
                                select
                                label="Client"

                                onChange={onChange}
                                name="client_id"
                                variant="outlined"
                                style={{ width: "100%" }}
                                required
                                InputLabelProps={{ shrink: true, }}
                            >
                                {allclients.map((option) => (
                                    <MenuItem key={option.id}
value={option.id}>
                                        {option.first_name + " " +
option.last_name}
                                    </MenuItem>
                                ))}
                            </TextField>
                            : <></>
                    }
                    <br/>
                    <br/>
                    {
                        state.client_id ?
                        <PaymentForm
                        added={newPayment}
                        amount={null}
                        status="Partial Payment"
                        client_id={state.client_id}

                    />
                    : <></>
                    }
                    </div>
                </>

            );

    }

    function getdates(data) {

        if (data == "clear") {
            getAllPAyments()
        } else {
            console.log("get dates")
            console.log(data)

            getAllPAyments({ from_date: data.datefrom, to_date:
data.dateto })
        }

    }

    const columns = [
        {
            dataField: 'date_paid',
            text: ' Date',
            sort:true
        },
        {
            dataField: 'id',
            text: ' ID',
            hidden: true
        }, {
            dataField: 'name',
            text: 'Name',
            sort:true
        }, {
            dataField: 'amount',
            text: 'Amount',
            sort:true
        }, {
            dataField: 'beg_balance',
            text: 'Beg. Balance',
            sort:true
        }, {
            dataField: 'end_balance',
            text: 'End. Balance',
            sort:true
        }];

    function printContent(){

        return(
            <>
            <BootstrapTable keyField="id" data={state.data}
columns={columns} />
            </>
        )
    }

    var printdata = {
        title: 'Payments',
        content: printContent()
    }
    return (
        <>
            <Link style={{ float: "right" }}
            >
                <Tooltip title="Add new client" aria-label="add">
                    <Fab size="medium" color="primary" aria-label="add"
onClick={handleClickOpen}>
                        <AddIcon />
                    </Fab>
                </Tooltip>
            </Link>
            <Button style={{ float: 'left' }} variant="contained"
onClick={handlePrint} color="primary">Print</Button>
            <div style={{ display: "none" }}><ComponentToPrint
data={printdata} ref={componentRef} /></div>
            <br />
            <br />
            <Dialog aria-labelledby="simple-dialog-title" open={open}
onClose={handleClose} >
                <div style={{ padding: "30px", width: '100%' }}>
                    {addNewPayment()}
                </div>
            </Dialog>
```

```
            <br />
            <DateRangeGen dates={getdates} />
            <ReactTable columns={columns}  data={state.data}/>
        </>
    )
}

function AllItems() {
    const breadcrumps = [
        {
            label: "Payments",
            icon: <ListAltIcon fontSize='small' />,
            route: "/all/payments"
        }
    ]

    return (
        <>
            <GenCont                title="All                Payments"
breadcrumps={breadcrumps} content={content()} />
            <br /><br />
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllItems);
```

10.2 Form

```
import React, { useState, useEffect } from 'react';
import { connect } from 'react-redux';
import TextField from '@material-ui/core/TextField';
import Button from '@material-ui/core/Button';
import Http from '../../Http';
import { useSnackbar } from 'notistack';

var _isMounted = false;
function PosForm(props) {
    const { enqueueSnackbar } = useSnackbar();
    _isMounted = true;

    const [state, setState] = useState({
        items: [],
        item_id: null,
        name: null,
        amount: props.amount,
        date_paid: null,
        cur_balance: null
    });

    useEffect(() => {
        getClientData();
    }, []);

    function getClientData() {
        var variant;

        Http.post('/api/v1/client/single', { client_id: props.client_id })
            .then((response) => {
                console.log(response.data.client)
                var cl = response.data.client;
                if (_isMounted) {
                    setState(prevState  =>  ({  ...prevState,  cur_balance:
cl[0].balance }));
                }
            })
            .catch((error) => {
                variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch client', { variant })
            });
    }

    function onChange(event) {//Sets the value of states
        const { name, value } = event.target;
        if (_isMounted) {
            setState(prevState => ({ ...prevState, [name]: value }));
        }
    }

    function submit(e) {
        e.preventDefault();
        if (confirm("Are you sure you want to create payment
transaction?")) {
            const posts = {
                client_id: props.client_id,
                amount: state.amount,
                date_paid: state.date_paid
            }
            if (state.amount) {
                submitPayment(posts)
            }
        }
    }

    function submitPayment(posts) {
        var variant;

        Http.post('/api/v1/payment', posts)
            .then(({ data }) => {
                variant = 'success';
                enqueueSnackbar('Payment  successfully  added',  {
variant })
                getClientData();
                if (props.added) {
                    props.added("");
                }
            })
            .catch((error) => {
                variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to create payment', { variant })
            });
    }

    const formatter = new Intl.NumberFormat('fil', {
        style: 'currency',
        currency: 'PHP',
        minimumFractionDigits: 2
    })

    return (
        <>
            {
                state.cur_balance > 0 ?
                <form onSubmit={submit} autoComplete="off">

                <label>Payment</label>  <label  style={{  float:  "right"
}}>Client Balance: {formatter.format(state.cur_balance)}</label>
                <TextField id="outlined-basic"
                    disabled={props.status == "Partial Payment" ? false :
true}
                    value={props.amount}
                    InputLabelProps={{ shrink: true, }}
                    required  name="amount"  onChange={onChange}
style={{  width:  "100%"  }}  label="Amount"  type="number"
variant="outlined" />
                <br /><br />

                <label>Date Paid</label>
                <TextField id="outlined-basic" InputLabelProps={{ shrink:
true, }}
                    required  name="date_paid"  onChange={onChange}
style={{  width:  "100%"  }}  label="Amount"  type="date"
variant="outlined" />
                <br /><br />

                <Button  style={{  float:  "right"  }}  type="submit"
variant="contained" color="primary">
                    Add payment
                </Button>
            </form>
            :<></>
            }
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(PosForm);
```

10.3 Controller

```
<?php

namespace App\Http\Controllers;

use App\Client;
use App\Payment;
use Illuminate\Http\Request;
use DB;
class PaymentController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {


        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {

            $payment = Payment::join('clients', 'payments.client_id', '=',
'clients.id')
                ->select(
                    'payments.*',
                    DB::raw('CONCAT(clients.first_name," "
clients.last_name) as name')
                );

                if (request('from_date') && request('to_date')) {
                    $payment->whereDate('payments.date_paid',
request('from_date'));
                    $payment->whereDate('payments.date_paid',
request('to_date'));
                }
             $payment =  $payment->get();



            return response()->json([
                'status' => 200,
                'payments' => $payment
            ], 200);
        } catch (Exception $e) {
            return
resource.');
        }
    }


    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {

            $client = Client::where('id', request('client_id'))->firstorfail();

            if ($client->balance < request('amount')) {
                return response()->json(['status' => 400,], 400);
            } else {
                $beg_bal = $client->balance;

                $client->balance -= request('amount');

                $payment = Payment::create([
                    'client_id' => request('client_id'),
                    'amount' => request('amount'),
                    'beg_balance' => $beg_bal,
                    'end_balance' => $client->balance,
                    'date_paid'
request('date_paid') : date('Y-m-d'),
                ]);

                $client->save();
            }

            return response()->json([
                'status' => 200,
            ], 200);
        } catch (Exception $e) {
            return
resource.');
        }
    }


}
```

11. Users

11.1 View

```
import React, { useState, useEffect } from 'react';
import { makeStyles } from '@material-ui/core/styles';
import { connect } from 'react-redux';
import GenCont from '../../../components/gen_container';
import
ui/icons/LaptopWindows';
import Fab from '@material-ui/core/Fab';
import AddIcon from '@material-ui/icons/Add';
import { Link } from 'react-router-dom';
import Tooltip from '@material-ui/core/Tooltip';
import Http from '../../../Http';
import BootstrapTable from 'react-bootstrap-table-next';
import
table2.min.css';
import paginationFactory from 'react-bootstrap-table2-paginator';
import
table2-paginator.min.css';
import ToolkitProvider, { Search, CSVExport } from 'react-bootstrap-
table2-toolkit';
import
toolkit.min.css';
import { useSnackbar } from 'notistack';
import DeleteIcon from '@material-ui/icons/Delete';
import IconButton from '@material-ui/core/IconButton';
import EditIcon from '@material-ui/icons/Edit';
import PeopleAltIcon from '@material-ui/icons/PeopleAlt';
import VisibilityIcon from '@material-ui/icons/Visibility';

import ReactTable from '../components/reactTable';

const { SearchBar, ClearSearchButton } = Search;
const { ExportCSVButton } = CSVExport;
var _isMounted = false;


function content() {
    _isMounted = true;
    const api = '/api/v1/user';
    const { enqueueSnackbar } = useSnackbar();

    var localTerminal = JSON.parse(localStorage.getItem("users") ||
"[]");
    const [state, setState] = useState({ data: localTerminal });

    useEffect(() => {
        Http.get(api)
            .then((response) => {
                localStorage.setItem('users',
JSON.stringify(response.data.users))
                localTerminal =
JSON.parse(localStorage.getItem("users") || "[]");
                if (_isMounted) {
                    setState({ data: localTerminal })
                }
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch users', { variant })
            });
    }, []);
```

```
    function deleteItem(key) {
        var variant;
        const itm = state.data;
        if (confirm('Are you sure you want to proceed?')) {

            Http.delete(`${api}/${key}`)
                .then((response) => {
                    if (response.status === 204) {
                        const index = itm.findIndex(
                            (item) => parseInt(item.id, 10) === parseInt(key,
10),
                        );
                        const update = [...itm.slice(0, index),
...itm.slice(index + 1)];
                        if (_isMounted) {
                            localStorage.setItem('users',
JSON.stringify(update))
                            localTerminal =
JSON.parse(localStorage.getItem("users") || "[]");
                            if (_isMounted) {
                                setState({ data: localTerminal })
                            }
                        }
                    }
                    variant = 'success'
                enqueueSnackbar('User successfully deleted', { variant })
                })
                .catch((error) => {
                    console.log(error);
                    variant = 'error'
                    enqueueSnackbar('Error deleting user', { variant })
                });

        }

    }

    function actionFormat(cell, row) {
        var id = row.id;
        return (
            <>
                <IconButton aria-label="delete" style={{ padding: "0" }}
onClick={() => deleteItem(id)} >
                    <DeleteIcon />
                </IconButton>
                  
                <Link to={{
                    pathname: `/form/users`,
                    state: {
                        name: row.name,
                        firstname: row.first_name,
                        midname: row.middle_name,
                        lastname: row.last_name,
                        email: row.email,
                        role: row.role,
                        isActive: row.isActive,
                        type: "Update",
                        id: id
                    }
                }}>
                  <IconButton aria-label="delete" style={{ padding: "0" }} >
                      <EditIcon />
                    </IconButton>
                </Link>
                   
                <Link to={{
                    pathname: `/form/users`,
                    state: {
                        name: row.name,
                        firstname: row.first_name,
                        midname: row.middle_name,
                        lastname: row.last_name,
                        email: row.email,
                        role: row.role,
                        type: "Update",
                        id: id,
                        isActive: row.isActive,
                        readonly: true
                    }
                }}>
                  <IconButton aria-label="delete" style={{ padding: "0" }} >
                      <VisibilityIcon />
                    </IconButton>
                </Link>
            </>
        );
    }

    const columns = [
        {
            dataField: 'id',
            text: ' ID',
            hidden: true
        }, {
            dataField: 'name',
            text: 'Name',
            sort: true
        }, {
            dataField: 'email',
            text: 'Email',
            sort: true
        }, {
            dataField: 'first_name',
            hidden: true
        }, {
            dataField: 'middle_name',
            hidden: true
        }, {
            dataField: 'last_name',
            hidden: true
        }, {
            dataField: 'role',
            text: 'Role',
            sort: true
        }, {
            dataField: 'isActive',
            text: 'isActive',
            sort: true,
            hidden: true
        }, {
            text: 'Actions',
            formatter: actionFormat,
            headerStyle: (colum, colIndex) => {
                return { width: '115px', textAlign: 'center' };
            }
        }];

    return (
        <>
            <Link style={{ float: "right" }} to="/form/users">
                <Tooltip title="Add new user" aria-label="add">
                    <Fab size="medium" color="primary" aria-label="add">
                        <AddIcon />
                    </Fab>
                </Tooltip>
            </Link>
            <br />

            <ReactTable columns={columns}  data={state.data}/>
        </>
    )
}

function AllUsers() {
    const breadcrumps = [
        {
            label: "Users",
            icon: <PeopleAltIcon fontSize='small' />,
            route: "/all/users"
        }
    ]

    return (
        <>
            <GenCont title="All Users" breadcrumps={breadcrumps}
content={content()} />
            <br /><br />
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllUsers);
```

11.2 Form

```
import React, { useState } from 'react';
import { useLocation } from 'react-router-dom'
import { connect } from 'react-redux';
import GenCont from '../../components/gen_container';
import FormCont from '../../components/form_container';
import LaptopWindowsIcon from '@material-ui/icons/LaptopWindows';
import AddIcon from '@material-ui/icons/Add';
import TextField from '@material-ui/core/TextField';
import SaveIcon from '@material-ui/icons/Save';
import Button from '@material-ui/core/Button';
import EditIcon from '@material-ui/icons/Edit';
import Http from '../../Http';
import { useSnackbar } from 'notistack';
import MenuItem from '@material-ui/core/MenuItem';
import Switch from '@material-ui/core/Switch';

var _isMounted = false;

function form() {
    _isMounted = true;
    const { enqueueSnackbar } = useSnackbar();
    const location = useLocation();
    var name = "";
    var firstname = "";
    var midname = "";
    var lastname = "";
    var email = "";
    var role = "";
    var password = "";
    var confirm = "";
    var isActive = false;
    var variant;
    var id;

    if (location.state) {
        name = location.state.name;
        firstname = location.state.firstname;
        midname = location.state.midname;
        lastname = location.state.lastname;
        email = location.state.email;
        role = location.state.role;
        id = location.state.id;
        isActive = location.state.isActive == 1 ? true : false;
    }

    const [state, setState] = useState({
        name: name,
        firstname: firstname,
        midname: midname,
        lastname: lastname,
        email: email,
        role: role,
        password: password,
        confirm: confirm,
        isActive: isActive,
        readonly: location.state && location.state.readonly ? location.state.readonly : false,
        error: false

    });

    function onChange(event) {//Sets the value of states
        const { name, value } = event.target;
        if (_isMounted) {
            setState(prevState => ({ ...prevState, [name]: value }));

            if (name === "confirm") {
                if (state.password !== value) {
                    setState(prevState => ({ ...prevState, error: true }))
                } else if (state.password === value) {
                    setState(prevState => ({ ...prevState, error: false }))
                }
            }
        }
    }

    function onChangeActive(event) {//Sets the value of states

        if (_isMounted) {
            setState(prevState => ({ ...prevState, isActive: !state.isActive }));
        }
    }

    function submit(e) {
        e.preventDefault();
        const posts = {
            first_name: state.firstname,
            last_name: state.lastname,
            middle_name: state.midname,
            email: state.email,
            role: state.role,
            isActive: state.isActive == true ? 1 : 0,
            password: state.password
        }
        if (location.state) {
            if (state.password) {
                if (state.password !== state.confirm) {
                    variant = 'error';
                  enqueueSnackbar('Passwords not matched', { variant })
                } else {
                    location.state ? submitUpdate(posts) : submitCreate(posts);
                }
            } else {
                location.state ? submitUpdate(posts) : submitCreate(posts);
            }

        } else {
            if (state.password !== state.confirm) {
                variant = 'error';
                enqueueSnackbar('Passwords not matched', { variant })
            } else {
                location.state ? submitUpdate(posts) : submitCreate(posts);
            }

        }
    }

    function submitCreate(posts) {

        Http.post(`/api/v1/auth/register`, posts)
            .then(({ data }) => {
                variant = 'success';
                enqueueSnackbar('User successfully created', { variant })
                if (_isMounted) {
                    setState({
                        name: "",
                        firstname: "",
                        midname: "",
                        lastname: "",
                        email: "",
                        password: "",
                        confirm: "",
                        error: false
                    });
                }
            })
            .catch((error) => {
                variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to create user', { variant })
            });
    }

    function submitUpdate(posts) {

        console.log("posts")
        console.log(posts)
        Http.patch(`/api/v1/user/${id}`, posts)
            .then(({ data }) => {
                variant = 'success';
                enqueueSnackbar('Terminal successfully updated', { variant })
            })
            .catch(() => {
                variant = 'error';
                enqueueSnackbar('Error updating terminal', { variant })
            });
    }

    const allroles = ['Production', "Finance", "Admin"];


    return (
        <>
            <form onSubmit={submit}
```

```
autoComplete="off">
<TextField
    id="outlined-basic"
    disabled={state.readonly == true ? true : false}
    required
    value={state.firstname}
    name="firstname"
    onChange={onChange}
    style={{ width: "100%" }}
    label="First name"
    variant="outlined"
    size="small" /><br /><br />
<TextField
    disabled={state.readonly == true ? true : false}
    id="outlined-basic"
    required
    value={state.midname}
    name="midname"
    onChange={onChange}
    style={{ width: "100%" }}
    label="Middle Name"
    variant="outlined"
    size="small" /><br /><br />
<TextField
    disabled={state.readonly == true ? true : false}
    id="outlined-basic"
    required
    value={state.lastname}
    name="lastname"
    onChange={onChange}
    style={{ width: "100%" }}
    label="Last Name"
    variant="outlined"
    size="small" /><br /><br />
<TextField
    disabled={state.readonly == true ? true : false}
    id="outlined-basic"
    required
    value={state.email}
    name="email"
    type="email"
    onChange={onChange}
    style={{ width: "100%" }}
    label="Email"
    variant="outlined"
    size="small" /><br /><br />
<TextField
    disabled={state.readonly == true ? true : false}
    id="outlined-select-currency"
    select
    label="Role"
    value={state.role}
    onChange={onChange}
    name="role"
    variant="outlined"
    style={{ width: "100%" }}
    size="small"
>
    {allroles.map((option) => (
        <MenuItem key={option} value={option}>
            {option}
        </MenuItem>
    ))}
</TextField><br /><br />

<TextField
    disabled={state.readonly == true ? true : false}
    required={location.state ? false : true}
    value={state.password}
    name="password"
    id="outlined-password-input"
    label="Password"
    type="password"
    variant="outlined"
    onChange={onChange}
    style={{ width: "100%" }}
    size="small"
/>
<br /><br />
<TextField
    disabled={state.readonly == true ? true : false}

    error={state.error} r
    required={location.state ? false : true}
    value={state.confirm}
    name="confirm"
    id="outlined-password-input"
                label="Confirm Password"
                type="password"
                variant="outlined"
                onChange={onChange}
                name="confirm"
                style={{ width: "100%" }}
                helperText={state.error === true ? "Mismatched
password" : ""}
                size="small"
            />
            <br /><br />
             Activate Account
            <Switch
                checked={state.isActive == true ? true :false}
                onChange={onChangeActive}
                color="primary"
                name="checkedB"
                inputProps={{ 'aria-label': 'primary checkbox' }}
            />

            {state.readonly != true ?
                <Button
                    style={{ float: "right" }}
                    variant="contained"
                    color="primary"
                    size="medium"
                    startIcon={<SaveIcon />}
                    type="submit"

                >
                    Save
                </Button> : <></>}
            <br /><br />
        </form>
    </>
  );

}


function content() {
    const formClasses = [{
        width: 50,

    }]
    return (
        <>
            <br />

            <FormCont
                formAttr={formClasses}
                form={form()}
            />

        </>
    );
}


function FormUser() {
    const location = useLocation();

    var label = "Create New";
    var icon = <AddIcon fontSize='small' />;
    var title = "Create New User";

    if (location.state) {
        label = "Update";
        icon = <EditIcon fontSize='small' />;
        title = location.state.readonly == true ? "View User" : "Update
User";

    }

    const breadcrumps = [
        {
            label: "Users",
            icon: <LaptopWindowsIcon fontSize='small' />,
            route: "/all/users"
        },
        {
            label: label,
            icon: icon,
        }
    ]

    return (
```

```
        <>
            <GenCont disable_card="true" title={title}
breadcrumps={breadcrumps} content={content()} />

        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(FormUser);
```

11.3 Controller

```
<?php

namespace App\Http\Controllers;

use App\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use DB;

class UserController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $users =   User::select(
                'users.*',
                DB::raw('CONCAT(users.first_name," "  ,
users.last_name) as name')
            )->get();
        } catch (Exception $e) {
        }

        return response()->json([
            'status' => 200,
            'users' => $users,
        ], 200);
    }

    public function specUser(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $users =   User::select(
                'users.role',
            )
                ->where('users.id', $user->id)
                ->first();
        } catch (Exception $e) {
        }

        return response()->json([
            'status' => 200,
            'role' => $users->role,
        ], 200);
    }


    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $us = User::where('id', $id)->firstOrFail();

            if (request('name')) {
                $us->name = request('name');
            }
            if (request('email')) {
                $us->email = request('email');
            }
            if (request('first_name')) {
                $us->first_name = request('first_name');
            }
            if (request('last_name')) {
                $us->last_name = request('last_name');
            }
            if (request('middle_name')) {
                $us->middle_name = request('middle_name');
            }
            if (request('role')) {
                $us->role = request('role');
            }
             $us->isActive = request('isActive');
            if (request('password')) {
                $us->password = Hash::make(request('password'));
            }

            $us->save();

            return response()->json([
                'status' => 200,
            ], 200);
        } catch (Exception $e) {
            return        $this->responseServerError('Error        updating
resource.');
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy(Request $request, $id)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $us = User::where('id', $id)->firstOrFail();
            $us->delete();

            return response()->json([
                'status' => 204,
                'message' => "Deleted successfully"
            ], 204);
        } catch (Exception $e) {
            return        $this->responseServerError('Error        deleting
resource.');
        }
    }
}
```

12. Reports

12.1 View

12.1.1 Components

A. Honorarium Report

```
import React, { useState, useEffect } from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableContainer from '@material-ui/core/TableContainer';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import Paper from '@material-ui/core/Paper';
import { useSnackbar } from 'notistack';
import { connect } from 'react-redux';
import Http from '../../../../Http';
var _isMounted = false;
function AllCats(props) {
  _isMounted = true;

  const { enqueueSnackbar } = useSnackbar();
  const [state, setState] = useState({
    items: [],
    package_id: null,
    package_name: null,
    selected: [],
    timeout: 0,
    total: 0,
  });

  const useRowStyles = makeStyles({
    root: {
      '& > *': {
        borderBottom: 'unset',
      },
    },
  });

  useEffect(() => {
    const subs = {}
    getData(subs);
  }, []);

  useEffect(() => {
    const subs = {
      type: "tracker",
      from_date: props.from_date,
      to_date: props.to_date
    }
    setState(prevState => ({ ...prevState, items: [] }));
    getData(subs);
  }, [props.searchChange]);

  function getData(subs){
    Http.post(`/api/v1/honorarium/tracker`, subs)
    .then((response) => {

      if (_isMounted) {
        setState(prevState => ({ ...prevState, items: [] }));
        props.totalHon(0);

        var honorarium = response.data.honoraria;
        setState(prevState => ({ ...prevState, items: honorarium
}));
        var totalHon = 0;
        honorarium.map((o) => {
          totalHon += o.amount;
        })
        var send ={
          field: "honorarium_data",
          value:honorarium
        }
        props.printing(send)
        props.totalHon(totalHon);
        setState(prevState => ({ ...prevState, total: totalHon }));
      }
    })
    .catch((error) => {
      var variant = 'error';
      console.log(error);
      enqueueSnackbar('Failed to fetch packages', { variant })
    });
  }

  function Row(props) {
    const { row } = props;
    const classes = useRowStyles();
    return (
      <React.Fragment>
        <TableRow className={classes.root}>

          <TableCell align="right">
            {row.date_paid}
          </TableCell>
          <TableCell align="right">{row.staff}</TableCell>
          <TableCell
align="right">{row.amount.toFixed(2)}</TableCell>
        </TableRow>
      </React.Fragment>
    );
  }
  return (
    <TableContainer component={Paper}>
      <label style={{float: "right", padding: "10px"}}>Total :
{state.total.toFixed(2)}</label>
      <Table aria-label="collapsible table">
        <TableHead>
          <TableRow>

            <TableCell align="right" ><b>Date</b></TableCell>
            <TableCell align="right" ><b>Staff</b></TableCell>
            <TableCell                                    align="right"
><b>Amount</b></TableCell>
          </TableRow>
        </TableHead>
        <TableBody>
          {state.items.map((row) => (
            <Row key={row.particular} row={row} />
          ))}
        </TableBody>
      </Table>
    </TableContainer>
  );

}

const mapStateToProps = (state) => ({
  isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllCats);
```

B. One Time Expenses Report

```
import React, { useState, useEffect } from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableContainer from '@material-ui/core/TableContainer';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import Paper from '@material-ui/core/Paper';
import { useSnackbar } from 'notistack';
import { connect } from 'react-redux';
import Http from '../../../../Http';
var _isMounted = false;
function AllCats(props) {
  _isMounted = true;
  const { enqueueSnackbar } = useSnackbar();
  const [state, setState] = useState({
    items: [],
    total: 0,
    package_id: null,
    package_name: null,
    selected: [],
    timeout: 0,
  });
  const useRowStyles = makeStyles({
    root: {
      '& > *': {
        borderBottom: 'unset',
      },
    },
  });
```

```
    useEffect(() => {
        const subs = {}
        getData(subs);
    }, []);

    useEffect(() => {
        const subs = {
            type: "tracker",
            from_date: props.from_date,
            to_date: props.to_date
        }
        setState(prevState => ({ ...prevState, items: [] }));
        getData(subs);
    }, [props.searchChange]);
    function getData(subs){
        Http.post(`/api/v1/ote/tracker`, subs)
        .then((response) => {
            if (_isMounted) {
                setState(prevState => ({ ...prevState, items: [] }));
                props.totalOtex(0);

                var ote = response.data.ote;
                setState(prevState => ({ ...prevState, items: ote }));
                var totalOte = 0;
                ote.map((o) => {
                    totalOte += o.amount;
                })

                var send ={
                    field: "otEx_data",
                    value:ote
                }

                props.printing(send)

                props.totalOtex(totalOte);
                setState(prevState => ({ ...prevState, total: totalOte }));
            }
        })
        .catch((error) => {
            var variant = 'error';
            console.log(error);
            enqueueSnackbar('Failed to fetch OTEx', { variant })
        });
    }

    function Row(props) {
        const { row } = props;
        const classes = useRowStyles();
        return (
            <React.Fragment>
                <TableRow className={classes.root}>

                    <TableCell
align="right">{row.expense_date}</TableCell>
                    <TableCell align="right">{row.particular}</TableCell>
                    <TableCell
align="right">{row.amount.toFixed(2)}</TableCell>
                </TableRow>
            </React.Fragment>
        );
    }
    return (
        <TableContainer component={Paper}>
                <label  style={{float:  "right",  padding:  "10px"}}>Total  :
{state.total.toFixed(2)}</label>
            <Table aria-label="collapsible table">
                <TableHead>
                    <TableRow>

                        <TableCell align="right" ><b>Date</b></TableCell>
                        <TableCell                                          align="right"
><b>Particular</b></TableCell>
                        <TableCell              align="right"              ><b>Amt.
Due</b></TableCell>
                    </TableRow>
                </TableHead>
                <TableBody>
                    {state.items.map((row) => (
                        <Row key={row.particular} row={row} />
                    ))}
                </TableBody>
            </Table>
        </TableContainer>
    );

}
```

```
const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllCats);
```

C. Operating Expenses Report

```
import React, { useState, useEffect } from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableContainer from '@material-ui/core/TableContainer';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import Paper from '@material-ui/core/Paper';
import { useSnackbar } from 'notistack';
import { connect } from 'react-redux';
import Http from '../../../../Http';
var _isMounted = false;
function AllCats(props) {
    _isMounted = true;

    const { enqueueSnackbar } = useSnackbar();
    const [state, setState] = useState({
        items: [],
        package_id: null,
        package_name: null,
        selected: [],
        timeout: 0,
        total: 0,
    });

    const useRowStyles = makeStyles({
        root: {
            '& > *': {
                borderBottom: 'unset',
            },
        },
    });

    useEffect(() => {
        const subs = {}
        getData(subs);
    }, []);

    useEffect(() => {
        const subs = {

            from_date: props.from_date,
            to_date: props.to_date
        }
        setState(prevState => ({ ...prevState, items: [] }));
        getData(subs);
    }, [props.searchChange]);

    function getData(subs){
        Http.post(`/api/v1/opex/company/report`, subs)
        .then((response) => {

            if (_isMounted) {
                props.totalcompopex(0);

                var opex = response.data.opex;
                setState(prevState => ({ ...prevState, items: opex }));
                var totalOpex = 0;
                opex.map((o) => {
                    totalOpex += o.amount;
                })
                props.totalcompopex(totalOpex);
                var send ={
                    field: "opEx_data",
                    value:opex
                }
                props.printing(send)
                setState(prevState => ({ ...prevState, total: totalOpex }));
            }
        })
        .catch((error) => {
            var variant = 'error';
            console.log(error);
            enqueueSnackbar('Failed to fetch packages', { variant })
        });
```

```
    }

    function Row(props) {
        const { row } = props;

        const classes = useRowStyles();
        return (
            <React.Fragment>
                <TableRow className={classes.root}>
                    <TableCell align="right">
                        {row.date_transact}
                    </TableCell>
                    <TableCell align="right">{row.particular}</TableCell>
                    <TableCell
align="right">{row.amount.toFixed(2)}</TableCell>
                </TableRow>
            </React.Fragment>
        );
    }
    return (
        <TableContainer component={Paper}>
            <label style={{float: "right", padding: "10px"}}>Total :
{state.total.toFixed(2)}</label>
            <Table aria-label="collapsible table">
                <TableHead>
                    <TableRow>

                        <TableCell align="right" ><b>Date</b></TableCell>
                        <TableCell                                    align="right"
><b>Particular</b></TableCell>
                        <TableCell              align="right"              ><b>Amt.
Due</b></TableCell>
                    </TableRow>
                </TableHead>
                <TableBody>
                    {state.items.map((row) => (
                        <Row key={row.particular} row={row} />
                    ))}
                </TableBody>
            </Table>
        </TableContainer>
    );

}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllCats);
```

D. Order/Sales Report

```
import React, { useState, useEffect } from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Box from '@material-ui/core/Box';
import Collapse from '@material-ui/core/Collapse';
import IconButton from '@material-ui/core/IconButton';
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableContainer from '@material-ui/core/TableContainer';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import Typography from '@material-ui/core/Typography';
import Paper from '@material-ui/core/Paper';
import        KeyboardArrowDownIcon        from        '@material-
ui/icons/KeyboardArrowDown';
import        KeyboardArrowUpIcon        from        '@material-
ui/icons/KeyboardArrowUp';
import { useSnackbar } from 'notistack';
import { connect } from 'react-redux';
import Http from '../../../../Http';
var _isMounted = false;
function AllCats(props) {
    _isMounted = true;

    const { enqueueSnackbar } = useSnackbar();
    const [state, setState] = useState({
        items: [],
        total: 0,
        package_id: null,
        package_name: null,
        selected: [],
        timeout: 0,
    });

    const useRowStyles = makeStyles({
        root: {
            '& > *': {
                borderBottom: 'unset',
            },
        },
    });

    useEffect(() => {
        const subs = { type: "tracker" }
        getOrder(subs);
    }, []);

    useEffect(() => {
        const subs = {
            type: "tracker",
            from_date: props.from_date,
            to_date: props.to_date
        }
        console.log(subs)
        setState(prevState => ({ ...prevState, items: [] }));
        getOrder(subs);
    }, [props.searchChange]);
    function getOrder(subs) {
        Http.post('/api/v1/order/index', subs)
            .then((response) => {
                if (_isMounted) {
                    props.totalsale(0);
                    var order = response.data.orders;

                    setState(prevState => ({ ...prevState, items: order }));

                    var totalNet = 0;
                    order.map((o) => {
                        totalNet += o.net;
                    })
                    props.totalsale(totalNet);
                    setState(prevState => ({ ...prevState, total: totalNet }));

                    var send ={
                        field: "revenue_data",
                        value:order
                    }
                    props.printing(send)
                }


            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch items', { variant })
            });

    }

    function Row(props) {
        const { row } = props;
        const [open, setOpen] = React.useState(false);
        const classes = useRowStyles();

        return (
            <React.Fragment>

                <TableRow className={classes.root}>
                    <TableCell>
                        <IconButton aria-label="expand row" size="small"
onClick={() => setOpen(!open)}>
                            {open ? <KeyboardArrowUpIcon /> :
<KeyboardArrowDownIcon />}
                        </IconButton>
                    </TableCell>
                    <TableCell
align="right">{row.date_verified}</TableCell>
                    <TableCell align="right">{row.name}</TableCell>
                    <TableCell
align="right">{row.amount_due.toFixed(2)}</TableCell>
                    <TableCell
align="right">{row.commission.toFixed(2)}</TableCell>
                    <TableCell
align="right">{row.net.toFixed(2)}</TableCell>
                </TableRow>
                <TableRow>
                    <TableCell style={{ paddingBottom: 0, paddingTop: 0 }}
colSpan={6}>
```

```
                    <Collapse in={open} timeout="auto"
unmountOnExit>
                      <Box margin={1}>
                        <Typography variant="h6" gutterBottom
component="div">
                          Details
                        </Typography>
                        <Table size="small" aria-label="purchases">

                          <TableBody>
                            <TableRow>
                              <TableCell key component="th"
scope="row">Code </TableCell>
                              <TableCell>{row.code}</TableCell>
                            </TableRow>
                            <TableRow>
                              <TableCell component="th"
scope="row"> Date Verified </TableCell>

<TableCell>{row.date_verified}</TableCell>
                            </TableRow>
                            <TableRow>
                              <TableCell component="th"
scope="row">Deduction</TableCell>

<TableCell>{row.deduction}</TableCell>
                            </TableRow>
                            <TableRow>
                              <TableCell component="th"
scope="row">Tax Rate</TableCell>

<TableCell>{row.tax_rate}%</TableCell>
                            </TableRow>
                            <TableRow>
                              <TableCell component="th"
scope="row">Package</TableCell>
                              <TableCell>{row.package}</TableCell>
                            </TableRow>


                          </TableBody>
                        </Table>
                      </Box>
                    </Collapse>
                  </TableCell>
                </TableRow>
              </React.Fragment>
          );
    }

    return (
        <TableContainer component={Paper}>
            <label style={{float: "right", padding: "10px"}}>Total :
{state.total.toFixed(2)}</label>
          <Table aria-label="collapsible table">
            <TableHead>
              <TableRow>
                <TableCell />
                <TableCell align="right" ><b>Date</b></TableCell>
                <TableCell align="right"
><b>Particular</b></TableCell>
                <TableCell align="right" ><b>Amt.
Due</b></TableCell>
                <TableCell align="right"
><b>Commission</b></TableCell>
                <TableCell align="right" ><b>Net</b></TableCell>
              </TableRow>
            </TableHead>
            <TableBody>
              {state.items.map((row) => (
                <Row key={row.name} row={row} />
              ))}
            </TableBody>
          </Table>
        </TableContainer>
    );

}


const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllCats);
```

E. Production Cost

```
import React, { useState, useEffect } from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Box from '@material-ui/core/Box';
import Collapse from '@material-ui/core/Collapse';
import IconButton from '@material-ui/core/IconButton';
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableContainer from '@material-ui/core/TableContainer';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import Typography from '@material-ui/core/Typography';
import Paper from '@material-ui/core/Paper';
import KeyboardArrowDownIcon from '@material-
ui/icons/KeyboardArrowDown';
import KeyboardArrowUpIcon from '@material-
ui/icons/KeyboardArrowUp';
import { useSnackbar } from 'notistack';
import { connect } from 'react-redux';
import Http from '../../../../Http';
var _isMounted = false;
function AllCats(props) {
    _isMounted = true;

    const { enqueueSnackbar } = useSnackbar();
    const [state, setState] = useState({
        items: [],
        package_id: null,
        total: 0,
        package_name: null,
        selected: [],
        timeout: 0,
    });

    const useRowStyles = makeStyles({
        root: {
            '& > *': {
                borderBottom: 'unset',
            },
        },
    });

    useEffect(() => {
        const subs = {}
        getOpex(subs);
    }, []);

    useEffect(() => {
        const subs = {

            from_date: props.from_date,
            to_date: props.to_date
        }
        setState(prevState => ({ ...prevState, items: [] }));
        getOpex(subs);
    }, [props.searchChange]);

    function getOpex(subs) {
        Http.post('/api/v1/order/opex/tracker', subs)
            .then((response) => {

                if (_isMounted) {
                    setState(prevState => ({ ...prevState, items: [] }));
                    var opex = response.data.opex
                    props.totalopex(0);
                    setState(prevState => ({ ...prevState, items: opex }));
                    var totalOpex = 0;
                    opex.map((o) => {
                        totalOpex += o.amount;
                    })
                    props.totalopex(totalOpex);
                    setState(prevState => ({ ...prevState, total: totalOpex
}));

                    var send ={
                        field: "prodCost_data",
                        value:opex
                    }

                    props.printing(send)
                }
            })
            .catch((error) => {
                var variant = 'error';
```

```
            console.log(error);
            enqueueSnackbar('Failed to fetch items', { variant })
        });

    }

    function Row(props) {
        const { row } = props;
        const [open, setOpen] = React.useState(false);

        const classes = useRowStyles();
        var items = JSON.parse(row.items);
        return (
            <React.Fragment>
                <TableRow className={classes.root}>
                    <TableCell>
                        <IconButton aria-label="expand row" size="small"
onClick={() => setOpen(!open)}>
                            {open ? <KeyboardArrowUpIcon /> :
<KeyboardArrowDownIcon />}
                        </IconButton>
                    </TableCell>
                    <TableCell align="right">
                        {row.date_verified}
                    </TableCell>
                    <TableCell align="right">{row.name}</TableCell>
                    <TableCell
align="right">{row.amount.toFixed(2)}</TableCell>
                </TableRow>
                <TableRow>

                    <TableCell style={{ paddingBottom: 0, paddingTop: 0 }}
colSpan={6}>
                        <Collapse in={open} timeout="auto"
unmountOnExit>
                            <Box margin={1}>
                                <Typography variant="h6" gutterBottom
component="div">
                                    Items
                                    </Typography>
                                {
                                    open ?
                                        <Table size="small" aria-
label="purchases">

                                            <TableBody>
                                                <TableRow>
                                                    <TableCell component="th"
scope="row">Particular </TableCell>
                                                    <TableCell component="th"
scope="row"> Amount </TableCell>
                                                </TableRow>

                                                {
                                                    open == true ?
                                                        <>
                                                            {
                                                                items.map((itm) => (

                                                                    <TableRow>
                                                                        <TableCell
align="left">{itm.particular}</TableCell>
                                                                        <TableCell
align="left">{itm.amount}</TableCell>
                                                                    </TableRow>

                                                                ))
                                                            }
                                                        </>
                                                        : <></>
                                                }

                                            </TableBody>
                                        </Table>
                                        : <></>
                                }
                            </Box>
                        </Collapse>
                    </TableCell>

                </TableRow>

            </React.Fragment>
        );
    }

    return (
        <TableContainer component={Paper}>
            <label style={{float: "right", padding: "10px"}}>Total :
{state.total.toFixed(2)}</label>
            <Table aria-label="collapsible table">
                <TableHead>
                    <TableRow>
                        <TableCell />
                        <TableCell align="right" ><b>Date</b></TableCell>
                        <TableCell align="right"
><b>Particular</b></TableCell>
                        <TableCell align="right" ><b>Amt.
Due</b></TableCell>
                    </TableRow>
                </TableHead>
                <TableBody>
                    {state.items.map((row) => (
                        <Row key={row.name} row={row} />
                    ))}
                </TableBody>
            </Table>
        </TableContainer>
    );

}


const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllCats);
```

F. Report Main View

```
import React, { useState, useEffect, useRef } from 'react';
import { makeStyles } from '@material-ui/core/styles';
import { connect } from 'react-redux';
import GenCont from '../../../components/gen_container';
import Http from '../../../Http';
import Orders from '../../tables/masters/tracker/order';
import ProdCost from '../../tables/masters/tracker/production_cost';
import Opex from '../../tables/masters/tracker/operating_expenses';
import Otex from '../../tables/masters/tracker/one_time_exp';
import Honorarium from '../../tables/masters/tracker/honorarium';
import BootstrapTable from 'react-bootstrap-table-next';
import 'react-bootstrap-table-next/dist/react-bootstrap-
table2.min.css';
import 'react-bootstrap-table2-paginator/dist/react-bootstrap-
table2-paginator.min.css';
import { Search, CSVExport } from 'react-bootstrap-table2-toolkit';
import 'react-bootstrap-table2-toolkit/dist/react-bootstrap-table2-
toolkit.min.css';
import { useSnackbar } from 'notistack';
import ListAltIcon from '@material-ui/icons/ListAlt';
import Accordion from '@material-ui/core/Accordion';
import AccordionSummary from '@material-
ui/core/AccordionSummary';
import AccordionDetails from '@material-ui/core/AccordionDetails';
import Typography from '@material-ui/core/Typography';
import ExpandMoreIcon from '@material-ui/icons/ExpandMore';
import TextField from '@material-ui/core/TextField';
import Button from '@material-ui/core/Button';
import { useReactToPrint } from 'react-to-print';
import { ComponentToPrint } from '../components/printTemplate';
import '../../../css/style.css'

var _isMounted = false;

const useStyles = makeStyles((theme) => ({
    root: {
        width: '100%',
    },
    heading: {
        fontSize: theme.typography.pxToRem(15),
        flexBasis: '50%',
        flexShrink: 0,
    },
    secondaryHeading: {
        fontSize: theme.typography.pxToRem(15),
        color: theme.palette.text.secondary,
    },
}));

function content() {
    const componentRef = useRef();
    const handlePrint = useReactToPrint({
```

```
        content: () => componentRef.current,
    });

    const formatter = new Intl.NumberFormat('fil', {
        style: 'currency',
        currency: 'PHP',
        minimumFractionDigits: 2
    })

    _isMounted = true;

    const classes = useStyles();

    const api = '/api/v1/ote';
    const { enqueueSnackbar } = useSnackbar();

    var localTerminal = JSON.parse(localStorage.getItem("ote") ||
"[]");
    const [state, setState] = useState({
        data: localTerminal,
        afterComm: null,
        afterProdCost: null,
        afterOpex: null,
        afterOtex: null,
        afterHon: null,
        from_date: null,
        to_date: null,
        searchChange: null,
        timeout: 0,
        revenue_data:[],
        prodCost_data:[],
        opEx_data:[],
        otEx_data:[],
        honorarium_data:[]
    });

    useEffect(() => {
        Http.get(api)
            .then((response) => {
                localStorage.setItem('ote',
JSON.stringify(response.data.ote))
                localTerminal = JSON.parse(localStorage.getItem("ote")
|| "[]");
                if (_isMounted) {
                    setState(prevState => ({ ...prevState, data:
localTerminal }))
                }
            })
            .catch((error) => {
                var variant = 'error';
                console.log(error);
                enqueueSnackbar('Failed to fetch items', { variant })
            });
    }, []);

    function setPrintingData(data){
        if (_isMounted) {
            setState(prevState => ({
                ...prevState, [data.field]: data.value,
            }))
        }
    }

    function getTotalSale(data) {
        if (state.timeout) clearTimeout(state.timeout);
        state.timeout = setTimeout(() => {
            if (_isMounted) {
                setState(prevState => ({
                    ...prevState, afterComm: parseFloat(data),
afterProdCost: null,
                    afterOtex: null,
                    afterHon: null
                }))
            }
        }, 300);
    }

    function onChange(event) {//Sets the value of states
        const { name, value } = event.target;
        if (_isMounted) {
            setState(prevState => ({ ...prevState, [name]: value }));
        }
    }

    function calcAfterProd(data) {
        if (state.timeout) clearTimeout(state.timeout);
        state.timeout = setTimeout(() => {
            if (_isMounted) {
                setState(prevState => ({ ...prevState, afterProdCost:
parseFloat(state.afterComm) - parseFloat(data), afterOpex: null }))
            }
        }, 300);
    }

    function calcAfterOpex(data) {
        if (state.timeout) clearTimeout(state.timeout);
        state.timeout = setTimeout(() => {
            if (_isMounted) {
                setState(prevState => ({ ...prevState, afterOpex:
parseFloat(state.afterProdCost) - parseFloat(data), afterOtex: null
}))
            }
        }, 300);
    }

    function calcAfterOtex(data) {
        if (state.timeout) clearTimeout(state.timeout);
        state.timeout = setTimeout(() => {
            if (_isMounted) {
                setState(prevState => ({ ...prevState, afterOtex:
parseFloat(state.afterOpex) - parseFloat(data), afterHon: null }))
            }
        }, 300);
    }

    function calcAfterHon(data) {
        if (state.timeout) clearTimeout(state.timeout);
        state.timeout = setTimeout(() => {
            if (_isMounted) {
                setState(prevState => ({ ...prevState, afterHon:
parseFloat(state.afterOtex) - parseFloat(data) }))
            }
        }, 300);
    }

    function getDatedReportTrans(e) {
        e.preventDefault();
        if (_isMounted) {
            setState(prevState => ({
                ...prevState,
                afterComm: null,
                afterProdCost: null,
                afterOpex: null,
                afterOtex: null,
                afterHon: null,
            }))

            setState(prevState => ({ ...prevState, searchChange:
state.from_date + state.to_date }))
        }
    }

    function printTableHeader(head, after, amount){
        return(
            <>
            <div style={{fontSize: 23, color: 'black'}}>
            <span>{head}</span>
            <span style={{float:'right'}}>Total After {after}:
{formatter.format(amount)}</span>
            </div>
            </>
        )
    }

    function printContent() {

        return (<>
        {printOrders()}
        {printProdCost()}
        {printopex()}
        {printOtex()}
        {printHon()}

        </>)
    }

    function printProdCost(){
        const columnsv = [{
            dataField: 'date_verified',
            text: 'Date',
            headerStyle: (colum, colIndex) => {
                return {textAlign: 'center' };
            },
        }, {
```

```
            dataField: 'amount',
            text: 'Amount Due',
            headerStyle: (colum, colIndex) => {
                return { textAlign: 'center' };
            },
        }];

        return (<>
            <div>
            <BootstrapTable    keyField='id'
                        data={state.prodCost_data}
                        columns={columnsv}
                        caption={printTableHeader("Production          Cost",
"Production Cost", state.afterProdCost)}
                        />
            </div>

            </>)
    }


    function printHon(){
        const columnsv = [{
            dataField: 'date_paid',
            text: 'Date',
            headerStyle: (colum, colIndex) => {
                return {textAlign: 'center' };
            },
        }, {
            dataField: 'staff',
            text: 'Staff',
            headerStyle: (colum, colIndex) => {
                return { textAlign: 'center' };
            },
        }, {
            dataField: 'amount',
            text: 'Amount Due',
            headerStyle: (colum, colIndex) => {
                return { textAlign: 'center' };
            },
        }];

        return (<>
            <div>
            <BootstrapTable    keyField='id'
                        data={state.honorarium_data}
                        columns={columnsv}

                        caption={printTableHeader("Honorarium",
"Honorarium", state.afterHon)}
                        />
            </div>

            </>)
    }

    function printOtex(){
        const columnsv = [{
            dataField: 'expense_date',
            text: 'Date',
            headerStyle: (colum, colIndex) => {
                return {textAlign: 'center' };
            },
        }, {
            dataField: 'particular',
            text: 'Particular',
            headerStyle: (colum, colIndex) => {
                return { textAlign: 'center' };
            },
        }, {
            dataField: 'amount',
            text: 'Amount Due',
            headerStyle: (colum, colIndex) => {
                return { textAlign: 'center' };
            },
        }];

        return (<>
            <div>
            <BootstrapTable    keyField='id'
                        data={state.otEx_data}
                        columns={columnsv}

                        caption={printTableHeader("One  Time  Expenses",
"OTEx", state.afterOtex)}
                        />
            </div>

            </>)
    }
    function printOrders(){
        const columnsv = [{
            dataField: 'date_verified',
            text: 'Date',
            headerStyle: (colum, colIndex) => {
                return {textAlign: 'center' };
            },
        }, {
            dataField: 'amount_due',
            text: 'Amount Due',
            headerStyle: (colum, colIndex) => {
                return { textAlign: 'center' };
            },
        }, {
            dataField: 'commission',
            text: 'Commission',
            headerStyle: (colum, colIndex) => {
                return {  textAlign: 'center' };
            },
        }, {
            dataField: 'net',
            text: 'Net',
            headerStyle: (colum, colIndex) => {
                return {  textAlign: 'center' };
            },
        }];
        return (<>
            <div>
            <BootstrapTable    keyField='id'
                        data={state.revenue_data}
                        columns={columnsv}

                        caption={printTableHeader("Revenue",
"Commission", state.afterComm)}
                        />
            </div>

            </>)
    }

    function printopex(){
        const columnsv = [{
            dataField: 'date_transact',
            text: 'Date',
            headerStyle: (colum, colIndex) => {
                return {textAlign: 'center' };
            },
        }, {
            dataField: 'particular',
            text: 'Particular',
            headerStyle: (colum, colIndex) => {
                return { textAlign: 'center' };
            }
        },{
            dataField: 'amount',
            text: 'Amount Due',
            headerStyle: (colum, colIndex) => {
                return { textAlign: 'center' };
            }
        }];

        return (<>
            <div>
            <BootstrapTable    keyField='id'
                        data={state.opEx_data}
                        columns={columnsv}

                        caption={printTableHeader("Operationg    Expense",
"OpEx", state.afterOpex)}
                        />
            </div>

            </>)
    }


    const shstyleTab = { width: "100%" }

    var printdata = {
        title: 'Tracker',
        content: printContent()
    }

    return (
```

```
      <>
        <div style={{ paddingLeft: "15%", paddingRight: "15%" }}>
          <div style={{ display: "none" }}><ComponentToPrint
data={printdata} ref={componentRef} /></div>
            <form onSubmit={getDatedReportTrans}>
              <TextField name="from_date" onChange={onChange}
required InputLabelProps={{ shrink: true }} size="small" type="date"
id="outlined-basic" label="From Date" variant="outlined" />
                  
              <TextField name="to_date" onChange={onChange}
required InputLabelProps={{ shrink: true }} size="small" type="date"
id="outlined-basic" label="To Date" variant="outlined" />
                  
              <Button type="submit" variant="contained"
color="primary"> Search</Button>
              <Button style={{ float: 'right' }} variant="contained"
onClick={handlePrint} color="primary">Print</Button>
            </form>
            <br />
            <br />
            <Accordion>
              <AccordionSummary
                expandIcon={<ExpandMoreIcon />}
                aria-controls="panel1a-content"
                id="panel1a-header"
              >
                <Typography
className={classes.heading}>Revenue</Typography>
                <Typography
className={classes.secondaryHeading} >
                  <table style={shstyleTab} >
                    <tr>
                      <td width="200px" >Total After
Commission:</td>

<td>{formatter.format(state.afterComm)}</td>
                    </tr>
                  </table>

                </Typography>
              </AccordionSummary>
              <AccordionDetails>
                <Orders printing={setPrintingData}
searchChange={state.searchChange} to_date={state.to_date}
from_date={state.from_date} totalsale={getTotalSale} />
              </AccordionDetails>
            </Accordion>
            {
              state.afterComm ?
                <>
                  <Accordion>
                    <AccordionSummary
                      expandIcon={<ExpandMoreIcon />}
                      aria-controls="panel1a-content"
                      id="panel1a-header"
                    >
                      <Typography
className={classes.heading}>Production Costs</Typography>
                      <Typography
className={classes.secondaryHeading} >
                        <table style={shstyleTab} >
                          <tr>
                            <td width="200px" >Total After
Producion Cost:</td>
                            <td
>{formatter.format(state.afterProdCost)}</td>
                          </tr>
                        </table>
                      </Typography>
                    </AccordionSummary>
                    <AccordionDetails>
                      <ProdCost printing={setPrintingData}
searchChange={state.searchChange} to_date={state.to_date}
from_date={state.from_date} totalopex={calcAfterProd} />
                    </AccordionDetails>
                  </Accordion>

                  {
                    state.afterProdCost ?
                      <>
                        <Accordion>
                          <AccordionSummary
                            expandIcon={<ExpandMoreIcon />}
                            aria-controls="panel1a-content"
                            id="panel1a-header"
                          >
                            <Typography
className={classes.heading}>Operating
Expenses</Typography>
                            <Typography
className={classes.secondaryHeading} >
                              <table style={shstyleTab} >
                                <tr>
                                  <td width="200px" >Total
After OpEx:</td>
                                  <td
>{formatter.format(state.afterOpex)}</td>
                                </tr>
                              </table>
                            </Typography>
                          </AccordionSummary>
                          <AccordionDetails>
                            <Opex printing={setPrintingData}
searchChange={state.searchChange} to_date={state.to_date}
from_date={state.from_date} totalcompopex={calcAfterOpex} />
                          </AccordionDetails>
                        </Accordion>

                        {
                          state.afterOpex ?
                            <>
                              <Accordion>
                                <AccordionSummary

expandIcon={<ExpandMoreIcon />}
                                  aria-controls="panel1a-
content"
                                  id="panel1a-header"
                                >
                                  <Typography
className={classes.heading}>One Time Expenses</Typography>
                                  <Typography
className={classes.secondaryHeading} >
                                    <table style={shstyleTab}
>
                                      <tr>
                                        <td width="200px"
>Total After OTEx:</td>
                                        <td
>{formatter.format(state.afterOtex)}</td>
                                      </tr>
                                    </table>
                                  </Typography>
                                </AccordionSummary>
                                <AccordionDetails>
                                  <Otex
printing={setPrintingData} searchChange={state.searchChange}
to_date={state.to_date} from_date={state.from_date}
totalOtex={calcAfterOtex} />
                                </AccordionDetails>
                              </Accordion>
                              {
                                state.afterOtex ?
                                  <>
                                    <Accordion>
                                      <AccordionSummary

expandIcon={<ExpandMoreIcon />}
                                        aria-
controls="panel1a-content"
                                        id="panel1a-header"
                                      >
                                        <Typography
className={classes.heading}>Honorarium</Typography>
                                        <Typography
className={classes.secondaryHeading} >
                                          <table
style={shstyleTab} >
                                            <tr>
                                              <td
width="200px" >Total After Homorarium:</td>
                                              <td
>{formatter.format(state.afterHon)}</td>
                                            </tr>
                                          </table>
                                        </Typography>
                                      </AccordionSummary>
                                      <AccordionDetails>
                                        <Honorarium
printing={setPrintingData} searchChange={state.searchChange}
to_date={state.to_date} from_date={state.from_date}
totalHon={calcAfterHon} />
                                      </AccordionDetails>
```

```
                                        </Accordion>
                                    </>
                                    : <></>
                                }
                            </>
                            : <></>
                        }



                    </>
                    : <></>
                }
            </>
            : <></>
        }

    </div>



    </>
  )
}

function AllItems() {
  const breadcrumps = [
    {
      label: "Tracker",
      icon: <ListAltIcon fontSize='small' />,
      route: "/report/tracker"
    }
  ]

  return (
    <>

      <GenCont title="  " breadcrumps={breadcrumps}
content={content()} />
      <br /><br />
    </>
  );
}

const mapStateToProps = (state) => ({
  isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(AllItems);
```

13. Predictive Analytics

13.1 View

13.1.1 Components

A. Form

```
import React, { useState } from 'react';
import { connect } from 'react-redux';
import MenuItem from '@material-ui/core/MenuItem';
import TextField from '@material-ui/core/TextField';
import Button from '@material-ui/core/Button';
var _isMounted = false;

function PredForm(props) {
  _isMounted = true;
  const [state, setState] = useState({ date: '', period: 'month' });

  function onChange(event) {//Sets the value of states
    const { name, value } = event.target;
    if (_isMounted) {
      setState(prevState => ({ ...prevState, [name]: value }));

    }
  }

  function submit(e) {
    e.preventDefault()
    const subs = {
      date: state.date,
      period: state.period
    }
    props.predinpt(subs)
  }

  function handlePrint(){
    if(props.print){
      props.print('')
    }
  }

  return (
    <>
      <br />
      <form
        onSubmit={submit}
      >
        <TextField name="date"
          onChange={onChange}
          value={state.date}
          required InputLabelProps={{ shrink: true }} size="small"
type="date" id="date" label="Prediction Date" variant="outlined" />
           
        <TextField
          required
          value={state.period}
          size="small"
          id="period"
          variant="outlined"
          style={{ width: "20%" }}
          select
          label="Period"
          onChange={onChange}
          name="period"
          disabled

        >

          <MenuItem key="monthly" value="month">
            Monthly
            </MenuItem>
          <MenuItem key="weekly" value="week">
            Weekly
            </MenuItem>

        </TextField>

           
        <Button type="submit" variant="contained"
color="primary"> Predict</Button>
        <Button style={{ float: 'right' }} variant="contained"
onClick={handlePrint} color="primary">Print</Button>
      </form>
      <br />
    </>
  );
}

const mapStateToProps = (state) => ({
  isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(PredForm);
```

B. Existing Sale Data Table

```
import React from 'react';
import { connect } from 'react-redux';
import { makeStyles } from '@material-ui/core/styles';
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableContainer from '@material-ui/core/TableContainer';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import Paper from '@material-ui/core/Paper';

const useStyles = makeStyles({
  table: {
    minWidth: 650,
  },
});

function OPStats(props) {
```

```
    const classes = useStyles();

    const data = props.onpopc? props.onpopc : [];

    return (
        <>
            <br /><br />
            <div style={{ width: "100%" }}>
                <TableContainer component={Paper}>
                    <Table  className={classes.table}  aria-label="simple
table">
                        <TableHead>
                            <TableRow>
                                <TableCell>Package</TableCell>
                                <TableCell        align="center">Number        of
Orders</TableCell>
                                <TableCell
align="center">Percentage</TableCell>
                            </TableRow>
                        </TableHead>
                        <TableBody>
                            {data.map((row) => (
                                <TableRow key={row.name}>
                                    <TableCell component="th" scope="row">
                                        {row.name}
                                    </TableCell>
                                    <TableCell
align="center">{row.no_ord}</TableCell>
                                    <TableCell
align="center">{parseInt(row.percent * 100) + '%'}</TableCell>
                                </TableRow>
                            ))}
                        </TableBody>
                    </Table>
                </TableContainer>
            </div>
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(OPStats);
```

C. Existing Sales Statistics

```
import React, { useState, useEffect } from 'react';
import { connect } from 'react-redux';
import OutlineCont from '../../../../components/outline_container';
import OdPackStats from './od_package_stats';
import SingleDetCont from './sigle_details_container';
var _isMounted = false;

function OrgDet(props) {
    _isMounted = true;
    const [state, setState] = useState({ dets: {} });

    useEffect(() => {
        if (props.orgdets) {
            if (_isMounted) {
                setState(prevState => ({ ...prevState, dets: props.orgdets
}));
            }
        }
    }, [props.orgdets]);


    return (
        <>
            <br />
            <OutlineCont label="Current Sale Details">
                <SingleDetCont        lable1="Average        Orders/Month"
content1={state.dets.oao  ?  state.dets.oao  :  "N/A"}  lable2="Total
Orders" content2={state.dets.oto ? state.dets.oto : "N/A"} />
                <OdPackStats onpopc={state.dets.onpopc} />
            </OutlineCont>
            <br />
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(OrgDet);
```

D. Predictive Sales Data Tables

```
import React from 'react';
import { connect } from 'react-redux';
import { makeStyles } from '@material-ui/core/styles';
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableContainer from '@material-ui/core/TableContainer';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import Paper from '@material-ui/core/Paper';

const useStyles = makeStyles({
    table: {
        minWidth: 650,
    },
});

const formatter = new Intl.NumberFormat('fil', {
    style: 'currency',
    currency: 'PHP',
    minimumFractionDigits: 2
})

function PredStat(props) {
    const classes = useStyles();

    const data = props.pacpo ? props.pacpo : [];

    var revenue = 0;

    var papStyleLabel = {
        padding: "5px",

        backgroundColor: "#c8f7c8",

    }
    var papStyleCont = {
        padding: "5px",
    }

    var pred_qty = 0;
    var pred_cost = 0;

    var revenue = 0;
    data.map((row) => {
        revenue          +=          Math.round(row.pred_cost)          *
Math.round(row.pred_qty)
    })

    return (
        <>
            <br /> <br />
            <div style={{ width: "100%" }}>
                <TableContainer component={Paper}>
                    <Table  className={classes.table}  aria-label="simple
table">
                        <TableHead>
                            <TableRow>
                                <TableCell>Package</TableCell>
                                <TableCell        align="center">Number        of
Orders</TableCell>
                                <TableCell align="center">Cost</TableCell>
                                <TableCell align="center">Total</TableCell>
                            </TableRow>
                        </TableHead>
                        <TableBody>
                            {data.map((row) => (

                                <TableRow key={row.name}>
                                    <TableCell component="th" scope="row">

                                        <span hidden>
                                            {pred_qty = 0}
                                            {pred_cost = 0}
                                            {pred_cost                                =
Math.round(row.pred_cost)}
                                            {pred_qty = Math.round(row.pred_qty)}
                                        </span>
```

```
                                {row.name}
                            </TableCell>
                            <TableCell
align="center">{pred_qty}</TableCell>
                            <TableCell
align="right">{formatter.format(pred_cost)}</TableCell>
                            <TableCell align="right">{
                                <>
                                {formatter.format(pred_qty                *
pred_cost)}
                                </>
                            }</TableCell>
                        </TableRow>
                    ))}
                </TableBody>
            </Table>
        </TableContainer>
        <br /><br />
        <div style={{ width: '50%', display: "inline-block" }}>
            <div style={{ display: "inline-block", width: "45%" }}>
                <Paper style={papStyleLabel} elevation={3} >
                    Predicted Revenue
                </Paper>
            </div>
               
            <div style={{ display: "inline-block", width: "45%" }}>
                <Paper style={papStyleCont} elevation={3} >
                    {formatter.format(revenue)}
                </Paper>
            </div>
        </div>
    </div>
  </>
 );
}

const mapStateToProps = (state) => ({
  isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(PredStat);
```

E. Predictive Statistics Sale

```
import React from 'react';
import { connect } from 'react-redux';
import OutlineCont from '../../../../components/outline_container';
import PredPackStats from './pred_package_stat';
import SingleDetCont from './sigle_details_container';

var _isMounted = false;

function PredDet(props) {
  _isMounted = true;
  const data = props.preds.pacpo ? props.preds.pacpo : [];
  var totalPredOrder = 0;
  data.map((row) => {
    totalPredOrder += Math.round(row.pred_qty)
  })

  return (
    <>
      <br />
      <OutlineCont label="Predictive Sale Details">
        <SingleDetCont  lable1={"No.  of  "  +  props.period}
content1={props.preds.pnt  ?  Math.round(props.preds.pnt)  :  "N/A"}
lable2="Total  Predicted  Orders"  content2={props.preds.po  ?
totalPredOrder : "N/A"} />
        <PredPackStats pacpo={props.preds.pacpo} />
      </OutlineCont>
      <br />
    </>
  );
}

const mapStateToProps = (state) => ({
  isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(PredDet);
```

F. Single Details Container

```
import React from 'react';
import { connect } from 'react-redux';
import Paper from '@material-ui/core/Paper';

function SingleDetCont(props) {
  var papStyleLabel = {
    padding: "5px",

    backgroundColor: "#c8f7c8",

  }
  var papStyleCont = {
    padding: "5px",

  }

  return (
    <>
      <div style={{ width: "100%" }}>
        <div style={{ width: '50%', display: "inline-block" }}>
          <div style={{ display: "inline-block", width: "45%" }}>
            <Paper style={papStyleLabel} elevation={3} >
              {props.lable1}
            </Paper>
          </div>
             
          <div style={{ display: "inline-block", width: "45%" }}>
            <Paper style={papStyleCont} elevation={3} >
            {props.content1}
            </Paper>
          </div>
        </div>

        <div style={{ width: '50%', display: "inline-block" }}>
          <div style={{ display: "inline-block", width: "45%" }}>
            <Paper style={papStyleLabel} elevation={3} >
              {props.lable2}
            </Paper>
          </div>
             
          <div style={{ display: "inline-block", width: "45%" }}>
            <Paper style={papStyleCont} elevation={3} >
              {props.content2}
            </Paper>
          </div>
        </div>
      </div>
    </>
  );
}

const mapStateToProps = (state) => ({
  isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(SingleDetCont);
```

G. Predictive Analytics Main View

```
import React, { useState, useEffect, useRef } from 'react';
import { connect } from 'react-redux';
import GenCont from '../../../../components/gen_container';
import Http from '../../../../Http';
import PredForm from './form'
import OrgDetails from './original_details'
import PredDetails from './predictive_details';
import          'react-bootstrap-table2-paginator/dist/react-bootstrap-
table2-paginator.min.css';
import { Search, CSVExport } from 'react-bootstrap-table2-toolkit';
import      'react-bootstrap-table2-toolkit/dist/react-bootstrap-table2-
toolkit.min.css';
import { useReactToPrint } from 'react-to-print';
import { ComponentToPrint } from '../../components/printTemplate';
import { useSnackbar } from 'notistack';
import ListAltIcon from '@material-ui/icons/ListAlt';
import BootstrapTable from 'react-bootstrap-table-next';

var _isMounted = false;
var variant;

const formatter = new Intl.NumberFormat('fil', {
  style: 'currency',
  currency: 'PHP',
  minimumFractionDigits: 2
```

```
})

function content() {
    _isMounted = true;
    const api = '/api/v1/pred-analytics/date';
    const { enqueueSnackbar } = useSnackbar();


    const [state, setState] = useState({
        data: [], orgDet: [],
        period: "month",
        preds: [], date: null, predord: 0, pnt: 0
    });

    function getPrediction(data) {

        const period = data.period;
        const ddate = data.date;

        Http.post(api, data)
            .then(({ data }) => {

                const orgs = {
                    oao: data.oao,
                    oto: data.oto,
                    onpo: data.onpo,
                    onpopc: data.onpopc
                }

                const preds = {
                    pnt: data.pnt,
                    po: data.po,
                    pacep: data.pacep,
                    pnpo: data.pnpo,
                    pacpo: data.pacpo
                }

                if (_isMounted) {
                    setState(prevState => ({
                        ...prevState, orgDet: orgs, period: period + "s",
preds: preds, date: ddate, predord: data.po,
                        pnt: data.pnt
                    }));

                }
                variant = 'success';
                enqueueSnackbar('Prediction fetched', { variant })
            })
            .catch(() => {
                variant = 'error';
                console.log(error);
                enqueueSnackbar('Prediction Failed', { variant })
            });
    }



    const componentRef = useRef();
    const handlePrint = useReactToPrint({
        content: () => componentRef.current,
    });

    function printContent() {

        return (<>
            {predPrintBasic()}
            {predTable()}

        </>)
    }
    function predTable() {
        const data = state.preds.pacpo ? state.preds.pacpo : [];

        function actionFormatQty(cell, row) { return
Math.round(row.pred_qty) }
        function actionFormatCost(cell, row) { return
formatter.format(Math.round(row.pred_cost)) }
        function actionFormatTotal(cell, row) { return( formatter.format(
Math.round(row.pred_cost) * Math.round(row.pred_qty)) )}

        const columns = [{
            dataField: 'id',
            text: ' ID',
            hidden: true
        }, {
            dataField: 'name',
            text: 'Package'
        }, {
            dataField: 'pred_qty',
            formatter: actionFormatQty,
            text: 'Predictive Orders'
        }, {
            dataField: 'pred_cost',
            formatter: actionFormatCost,
            text: 'Cost'
        }, {
            dataField: 'id',
            formatter: actionFormatTotal,
            text: 'Total'
        }];
        return (
            <>
                <br />
                <BootstrapTable keyField="id" data={data}
columns={columns} />
            </>
        )
    }
    function predPrintBasic() {
        const data = state.preds.pacpo ? state.preds.pacpo : [];
        var revenue = 0;
        var totalPredOrder = 0;
        data.map((row) => {
            revenue += Math.round(row.pred_cost) *
Math.round(row.pred_qty);
            totalPredOrder += Math.round(row.pred_qty)
        })
        return (<>
            <table style={{ width: '100%' }}>
                <tr>
                    <td>
                        <tr>
                            <td>Date of Prediction</td>
                            <td>:</td>
                            <td>{state.date}</td>
                        </tr>
                        <tr>
                            <td>Period Used</td>
                            <td>:</td>
                            <td>{state.period.toUpperCase()}</td>
                        </tr>
                    </td>
                    <td>
                        <tr>
                            <td>No. of months/weeks</td>
                            <td>:</td>
                            <td>{Math.round(state.pnt)}</td>
                        </tr>
                        <tr>
                            <td>Total Predicted Orders</td>
                            <td>:</td>
                            <td>{Math.round(totalPredOrder)}</td>
                        </tr>
                    </td>
                    <td style={{ verticalAlign: 'text-top' }}>
                        <tr>
                            <td>Predicted Revenue</td>
                            <td>:</td>
                            <td>{formatter.format(revenue)}</td>
                        </tr>
                    </td>
                </tr>

            </table>
        </>)
    }

    var printdata = {
        title: 'Predictive Analytics',
        content: printContent()
    }

    return (
        <>
            <div style={{ display: "none" }}><ComponentToPrint
data={printdata} ref={componentRef} /></div>
            <hr />
            <PredForm predinpt={getPrediction} print={handlePrint} />
            <OrgDetails orgdets={state.orgDet} />
            <hr />
            <PredDetails period={state.period} preds={state.preds} />

        </>
    )
```

```
}

function Preddate() {
    const breadcrumps = [
        {
            label: "Predictive Analytics",
            icon: <ListAltIcon fontSize='small' />,
            route: "/all/items"
        },
        {
            label: "Date",
            icon: <ListAltIcon fontSize='small' />,
        }
    ]

    return (
        <>
            <GenCont                title="Predictive                Analytics"
breadcrumps={breadcrumps} content={content()} />
            <br /><br />
        </>
    );
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(Preddate);
```

13.2 Controller

```
<?php

namespace App\Http\Controllers;

use App\Client;
use App\Item;
use App\Order;
use App\OrderItem;
use App\Package;
use DateTime;
use Illuminate\Http\Request;
use DB;

class predTotalCostperPckOrd
{
    public $id;
    public $name;
    public $pred_qty;
    public $pred_cost;
    public $pred_total_cost;
}
class predPackOrdPercnt
{
    public $id;
    public $name;
    public $ptotal_ord;
    public $perprcnt_noOrd;
}
class orgOrdPackPerc
{
    public $id;
    public $name;
    public $percent;
    public $no_ord;
}
class predAmtCostEPack
{
    public $pid;
    public $name;
    public $sumOrgP;
    public $sumSumMk;
    public $sumAvgMk;
    public $overSum;
    public $count;
}

class ReportController extends Controller
{
    public function predictiveAnalyticsDate(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }
        try {
            $period_type = request("period");
            $oao = $this->avgOrgVOrders($period_type);
            $pnt          =          $this->predictiveNoTimes($period_type,
request("date"));
            $po = $oao * $pnt;
            $oto = $this->totalOrgNoOrder();
            $onpo = $this->noOrdEPackage();
            $onpopc = $this->ordEPackPercentage($onpo, $oto);
            $pacep = $this->predAmountCostEPackage();
            $pnpo = $this->predNoPackOrders($onpopc, $po);
            $pacpo = $this->predTotalCostAmtPckOrd($pnpo, $pacep);
            $revenue = $this->calculateRevenue($pacpo);
        } catch (Exception $e) {
        }


        return response()->json([
            'status' => 200,
            'oao' => $oao,
            'pnt' => $pnt,
            'po' => $po,
            'oto' => $oto,
            'onpo' => $onpo,
            'onpopc' => $onpopc,
            'pacep' => $pacep,
            'pnpo' => $pnpo,
            'pacpo' => $pacpo,
            'revenue' => $revenue,
        ], 200);
    }
    function avgOrgVOrders($period_type)
    {
        $orders = [];
        if ($period_type == "month") {
            $orders = Order::select(
                'orders.date_verified',
                DB::raw('(COUNT(orders.id))as no_days '),
                DB::raw('YEAR(created_at)                                year,
MONTH(orders.date_verified) month'),
                DB::raw("DATE_FORMAT(orders.date_verified,      '%m')
new_date")
            )
                ->where('orders.transaction_status', 'verified')
                ->groupby('year', 'month')
                ->orderBy('orders.date_verified', 'ASC')
                ->get();

            $first_date = $orders[0]->date_verified;
            $last_date = $orders[count($orders) - 1]->date_verified;

            $d1 = strtotime($first_date);
            $d2 = strtotime($last_date);
            $no_period = abs((date('Y', $d2) - date('Y', $d1)) * 12 +
(date('m', $d2) - date('m', $d1)));
            $no_period += 1;
        } else {

            $orders = Order::select(
                'orders.date_verified',
                DB::raw('(COUNT(orders.id))as no_days '),
                DB::raw('YEAR(created_at)                                year,
MONTH(orders.date_verified)  month,  WEEK(orders.date_verified)
week'),
                DB::raw("DATE_FORMAT(orders.date_verified,      '%m')
new_date")
            )
                ->where('orders.transaction_status', 'verified')
                ->groupby('year', 'month', 'week')
                ->orderBy('orders.date_verified', 'ASC')
                ->get();

            $first_date = $orders[0]->date_verified;
            $last_date = $orders[count($orders)-1]->date_verified;

            $date1 = new DateTime($first_date);
            $date2 = new DateTime($last_date);

            $no_period = $date1->diff($date2)->days / 7;
        }


        $total = 0;

        foreach ($orders as $o) {
            $total += $o['no_days'];
        }
```

```
        $average = $total / $no_period;

        return round($average);
    }

    function predictiveNoTimes($period_type, $expcDate)
    {

        $currentDate = date("Y-m-d");

        $ptn = $period_type == "month" ? 30 : 7;

        $date1 = new DateTime($currentDate);
        $date2 = new DateTime($expcDate);
        $pnt = $date1->diff($date2)->days / $ptn;
        return $pnt;
    }

    function totalOrgNoOrder()
    {
        $ord = Order::where('transaction_status', 'Verified')
            ->select(DB::raw('COUNT(id) as org_orders'))
            ->get();

        return $ord[0]->org_orders;
    }

    function noOrdEPackage()
    {
        $ord = Order::join('packages', 'orders.package_id', '=',
'packages.id')
            ->where('transaction_status', 'Verified')
            ->select('packages.id', 'packages.name',
DB::raw('COUNT(orders.id) as no_orders'))
            ->groupBy('orders.package_id')
            ->get();

        return $ord;
    }

    function ordEPackPercentage($orders, $total)
    {
        $percArr = [];
        foreach ($orders as $o) {
            $pa = new orgOrdPackPerc();
            $pa->id = $o['id'];
            $pa->name = $o['name'];
            $pa->percent = round(($o['no_orders'] / $total), 2);
            $pa->no_ord =  $o['no_orders'];
            $percArr[] = $pa;
        }

        return $percArr;
    }

    function predAmountCostEPackage()
    {
        $packages = Package::select('packages.*')->get();

        $percArr = [];
        foreach ($packages as $p) {
            $ordServ = OrderItem::join('orders', 'order_items.order_id',
'=', 'orders.id')
                ->join('packages', 'orders.package_id', '=', 'packages.id')
                ->where('orders.package_id', $p['id'])
                ->where('orders.transaction_status', 'Verified')
                ->select(
                    'orders.package_id as pid',
                    'packages.name as name',
                    DB::raw('SUM(order_items.original_price) as
sum_orgprice'),
                    DB::raw('SUM(order_items.markup) as sum_markup'),
                    DB::raw('COUNT(order_items.id) as count')

                )
                ->groupby('order_items.order_id')
                ->get();
            $sumAvgMk = $ordServ[0]->sum_markup / $ordServ[0]-
>count;
            $pa = new predAmtCostEPack();
            $pa->pid = $ordServ[0]->pid;
            $pa->name = $ordServ[0]->name;
            $pa->sumOrgP = $ordServ[0]->sum_orgprice;
            $pa->sumAvgMk = $sumAvgMk;
            $pa->sumSumMk = $ordServ[0]->sum_markup;
            $pa->count = $ordServ[0]->count;
            $pa->overSum = $ordServ[0]->sum_orgprice +
$sumAvgMk;
            $percArr[] = $pa;
        }

        return $percArr;
    }

    function predNoPackOrders($onpopc, $po)
    {

        $percArr = [];
        foreach ($onpopc as $onp) {
            $pa = new predPackOrdPercnt();
            $pa->id = $onp->id;
            $pa->name = $onp->name;
            $pa->ptotal_ord = $po;
            $pa->perprcnt_noOrd = $po * $onp->percent;
            $percArr[] = $pa;
        }

        return  $percArr;
    }

    function predTotalCostAmtPckOrd($pnpo, $pacep)
    {
        $percArr = [];

        foreach ($pnpo as $pn) {
            foreach ($pacep as $pac) {
                if ($pn->id === $pac->pid) {
                    $pa = new predTotalCostperPckOrd();
                    $pa->id = $pac->pid;
                    $pa->name = $pac->name;
                    $pa->pred_qty = $pn->perprcnt_noOrd;
                    $pa->pred_cost = $pac->overSum;
                    $pa->pred_total_cost = $pn->perprcnt_noOrd * $pac-
>overSum;
                    $percArr[] = $pa;
                }
            }
        }

        return $percArr;
    }

    function calculateRevenue($pacpo)
    {

        $revenue = 0;

        foreach ($pacpo as $p) {
            $revenue += $p->pred_total_cost;
        }

        return $revenue;
    }

    public function verifiedOrders(Request $request)
    {
        // Get user from $request token.
        if (!$user = auth()->setRequest($request)->user()) {
            return $this->responseUnauthorized();
        }

        try {
            $verford = Order::select(DB::raw('COUNT(id) as count'))-
>where('transaction_status', 'Verified')->first();
            $clients = Client::select(DB::raw('COUNT(id) as count'))-
>first();
            $items = Item::select(DB::raw('COUNT(id) as count'))-
>first();

            return response()->json([
                'verford' => $verford->count,
                'clients' => $clients->count,
                'items' => $items->count
            ], 200);
        } catch (Exception $e) {
        }
    }
}
```

14. Data Mining

14.1 View

```
import React, { useState, useEffect } from 'react';
import { connect } from 'react-redux';
import Http from '../../Http';
import ReactTable from '../../pages/tables/components/reactTable';
import { Button, Dialog, DialogActions, DialogContent, DialogContentText, DialogTitle } from '@material-ui/core';
import GenCont from '../../components/gen_container';
import ListAltIcon from '@material-ui/icons/ListAlt';
var _isMounted = false;
function Scraping(props) {
    _isMounted = true;
    var localTerminal = JSON.parse(localStorage.getItem("mining") || "[]");

    const [state, setState] = useState({ data: localTerminal, author: null, authorLink: null, authorDesc: null, loadingDesc: null });
    const [open, setOpen] = React.useState(false);

    const handleClose = () => {
        setOpen(false);
    };

    function setauthor(value) {
        setOpen(true);
        if (_isMounted) {
            getAUthDesc(value.link)
            setState(prevState => ({ ...prevState, author: value.author, authorLink: value.link }));
        }
    };

    function getAUthDesc(link) {
        setState(prevState => ({ ...prevState, loadingDesc: true }));
        Http.post('/api/v1/specscraped', { link: link })
            .then(({ data }) => {
                setState(prevState => ({ ...prevState, authorDesc: data.cont[0], loadingDesc: null }));
            })
            .catch((error) => {
                setState(prevState => ({ ...prevState, loadingDesc: null }));
            });
    }

    useEffect(() => {
        Http.post('/api/v1/scraped')
            .then(({ data }) => {

                var links = data.links;
                var reslinks = [];


                links.map((l, index) => {
                    l.id = index;
                    if (l.author[0] == ' ') {
                        l.author = l.author.substring(1);
                    }
                    var lsplit = l.link.split("/");

                    if (lsplit[1][0] == l.author[0]) {


                        if (reslinks.length < 1) {
                            reslinks.push(l)
                        } else {
                            var result = reslinks.filter(function (v) {
                                return (v.author == l.author ? true : false);
                            })

                            if (result == false) {
                                reslinks.push(l)
                            }
                        }
                    }

                })

                localStorage.setItem('mining', JSON.stringify(reslinks))
                localTerminal = JSON.parse(localStorage.getItem("mining") || "[]");
                if (_isMounted) {
                    setState({ data: localTerminal })
                }
            })
            .catch((error) => {

            });
    }, []);


    function actionFormat(cell, row) {
        return (
            <><Button variant="contained" color="primary" onClick={(e) => setauthor(row)} >Details</Button> </>
        )
    }

    function modal() {
        return (
            <>

                <Dialog
                    open={open}
                    onClose={handleClose}
                    aria-labelledby="alert-dialog-title"
                    aria-describedby="alert-dialog-description"
                >
                    <DialogTitle id="alert-dialog-title">{"Author Information"}</DialogTitle>
                    <DialogContent>
                        <DialogContentText id="alert-dialog-description">
                            <b>  Author :</b>    {state.author} <br />
                            <b> Author Description: </b> <br />
                            {state.loadingDesc ? "Fetching description ..." : <>{state.authorDesc ? state.authorDesc : "No Available Description"}</>}
                            <br /><br /><br />
                            <a href={'https://www.amazon.com/' + state.authorLink} target='_blank' >Amazon profile</a>
                            <br />
                            <iframe src={"https://www.amazon.com/" + state.authorLink} name="myFrame" style={{ width: 500, height: 500 }}></iframe>
                        </DialogContentText>
                    </DialogContent>
                    <DialogActions>
                    </DialogActions>
                </Dialog>
            </>
        )
    }


    const columns = [
        {
            dataField: 'author',
            text: ' Author',
        }, {
            dataField: 'link',
            text: ' link',
            hidden: true
        }, {
            text: 'Actions',
            dataField: 'id',
            formatter: actionFormat,
            headerStyle: (colum, colIndex) => {
                return { width: '120px', textAlign: 'center' };
            }
        }];


    function content() {
        return (<>
            {modal()}
            <ReactTable columns={columns} data={state.data} />

        </>)
    }

    const breadcrumps = [
        {
            label: "Data Mining",
            icon: <ListAltIcon fontSize='small' />,
            route: "/datamining"
        }
    ]

    return (
```

```
      <>
        <>          <GenCont          title="Data          Mining"
breadcrumps={breadcrumps} content={content()} /> </>
      </>
   );
}

const mapStateToProps = (state) => ({
   isAuthenticated: state.Auth.isAuthenticated,
});

export default connect(mapStateToProps)(Scraping);
```

14.2 Controller

```
<?php

namespace App\Http\Controllers;

use DOMDocument;
use Illuminate\Http\Request;
use stdClass;

class DataMining extends Controller
{

   public function getSpecDetails(Request $request){
      $dom = new DOMDocument();
      $html                                                    =
file_get_contents('https://www.amazon.com/'.request('link'));
      @$dom->loadHTML($html);
      $ols = $dom->getElementsByTagName('span');

      $cont = array();
      foreach ($ols as $ol) {
         $classes = $ol->getAttribute('id');

         if (strpos($classes, 'author_biography') !== false) {
            array_push($cont, trim($ol->nodeValue));
         }
      }

      return response()->json([
         'status' => 200,
         'cont' => $cont,
      ], 200);

   }

   public function getScrapedv2(Request $request)
   {
      $dom = new DOMDocument();
      $html                                                    =
file_get_contents('https://www.amazon.com/s?rh=n%3A21102321
011&fs=true&ref=lp_21102321011_sar');
      @$dom->loadHTML($html);

      $ols = $dom->getElementsByTagName('div');
      $key = 0;

      $cont = array();
      foreach ($ols as $ol) {
         $classes = $ol->getAttribute('class');

         if (strpos($classes, 's-asin') !== false) {
            $nodes = $ol->getElementsByTagName('div');

            foreach ($nodes as $node) {
               array_push($cont, trim($node->nodeValue));

            }
            $key +=  1;
         }
      }

      $name = array();
      foreach ($cont as $c) {
         $check = substr($c, 0, 2);

         if ($check == 'by') {
            $sCC = substr($c, 3);
            $checkin =  explode("|", $sCC);

            $checkcom = explode(",", $checkin[0]);
            foreach ($checkcom as $cc) {
               $checkand = explode("and", $cc);

               foreach ($checkand as $ca) {

                  array_push($name, $ca);
               }
            }
         }
      }

      $toDistinct =  array_unique($name);

      $un = array();
      $key = 0;
      foreach ($toDistinct as $td) {
         $key += 1;
         $authdet = new stdClass();
         $authdet->id = $key;
         $authdet->author = $td;
         $un[] = $authdet;
      }

      return response()->json([
         'status' => 200,
         'html' => $html,
         'imgclass' => $cont,
         'result' => $un,
         'links' => $this->getLinks($dom),
      ], 200);
   }

   function getLinks($dom)
   {
      $ols = $dom->getElementsByTagName('div');
      $un = array();
      foreach ($ols as $ol) {
         $classes = $ol->getAttribute('class');

         if (strpos($classes, 's-asin') !== false) {

            $nodes = $ol->getElementsByTagName('div');

            foreach ($nodes as $key=>$node) {
               $as =  $node->getElementsByTagName('a');

               $check = substr($node->nodeValue, 0, 2);

               if ($check == 'by') {

                  $sCC = substr(trim($node->nodeValue), 3);
                  $checkin =  explode("|", $sCC);

                  $checkcom = explode(",", $checkin[0]);
                  foreach ($checkcom as $cc) {

                     $checkand = explode("and", $cc);

                     foreach ($checkand as $keyand=>$ca) {
                        foreach ($as as $keya=>$aa) {
                           $authdet = new stdClass();
                           $authdet->author = $checkand[$keya];
                           $authdet->link = $aa->getAttribute('href');
                           $un[] = $authdet;
                        }
                     }
                  }
               }
            }

         }
      }

      return $un;
   }
}
```

II. Technical Reference

1. Final System Specification

The hardware and software requirements for this web application are listed in this section.

A. Hardware

Component	Minimum	Recommended
Processor	1.9 gigahertz (GHz) x86- or x64-bit dual core processor with SSE2 instruction set	3.3 gigahertz (GHz) or faster 64-bit dual core processor with SSE2 instruction set
Memory	2-GB RAM	4-GB RAM or more
Display	Super VGA with a resolution of 1024 x 768	Super VGA with a resolution of 1024 x 768

Running this web application on a computer that has less than the recommended requirements may result in inadequate performance. Additionally, satisfactory performance may be experienced running this web system that use a different hardware configuration than those published here—for example, a system with a modern quad-core processor, lower clock speed, and more RAM.

This web application is designed to work best over networks that have the following elements:

- Bandwidth greater than 1 MBps
- Latency under 60 ms

B. Operating Systems

The web application is compatible to any operating systems that has web browser installed, browser preferences are Google Chrome and Mozilla Firefox. Below is the Operating System used during development and testing.

Edition: Windows 10 Home Single Language
Version: 20H2
OS Build: 19042.1348
Experience: Windows Feature Experience Pack 120.2212.3920.0

C. Programming Language

The web application has two (2) different development environment types, the Back-end environment and front-end environment. These two environments are communicating thru API (Application Programming Interface). API is a software intermediary that allows two applications to talk to each other.

The Back-end is the code that runs on the server, that receives requests from the clients, and contains the logic to send the appropriate data back to the client. The back-end also includes the database, which will persistently store all of the data for the application.

The programming language used on back-end development is PHP with Laravel framework.
PHP (Hypertext Preprocessor) is known as a general-purpose scripting language that can be used to develop dynamic and interactive websites. The results are in faster site loading speeds. It is also flexible for database connectivity.

Laravel is an open-source PHP framework, which is robust and easy to understand. It follows a model-view-controller design pattern. Laravel reuses the existing components of different frameworks which helps in creating a web application. The web application thus designed is more structured and pragmatic.

Front-end web development is the development of the graphical user interface of a website, through the use of HTML, CSS, and JavaScript, so that users can view and interact with that website.

The programming language used on Front-end environment is JavaScript with REACT JS framework.

JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. Where HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user.

React.js is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. It's used for handling the view layer for web and mobile apps. React also allows the developers to create reusable UI components.

D. Server Application Used

This web application uses LiteSpeed web Server (LSWS). LSWS is a proprietary web server software. IT is the 4th most popular web server, estimated used by 10% of websites as of July 2021.
The software uses the same configuration format as Apache HTTP Server and is compatible with most Apache features.

LSWS is compatible with commonly-used Apache features, including mod_rewrite, .htaccess, and mod_security. LSWS can load Apache configuration files directly and works as a drop-in replacement for Apache while fully integrating with popular control panels. LSWS replaces all Apache functions, but uses an event driven approach to handle requests.

2. Maintenance Plan for the Software System

The web application can be tested offline thru local apache server. Clone the repository provided below and read the README.md file for installation instructions.

https://github.com/MBSFMS/mbsfms.git

A. System live Installation

MBSFMS is hosted on Hostinger. Hostinger is a hosting provider that uses LSWS server, a server that is similar to Apache Server. MySQL Database is also being used in this hosting provider.

For testing purposes, the developer will be using the Hostinger as sample hosting provider and the hosting will be used for the following system live installation instructions.

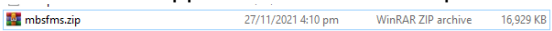
1. Assuming that the following are already purchased

- Hosting provider (Hostinger)
- Domain (mb-sfms.com)
  - Hostinger offers 1 free domain
- SSL
  - SSL to make the web application secured

2. Clone the repository.
https://github.com/MBSFMS/mbsfms.git

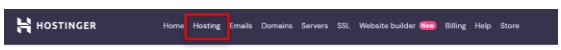
3. On local project folder, Zip files on root directory except node_modules if installed.

4. Name the zipped file as `mbsfms.zip`

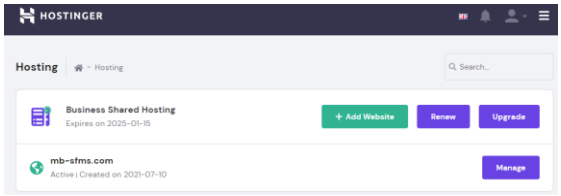


5. Go to hosting provider (Hostinger)

6. Go to Hosting tab



7. Navigate the purchased domain. In this case navigate mbsfms.com



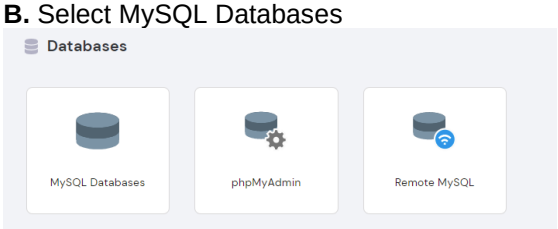
8. Press Manage

9. Managing database

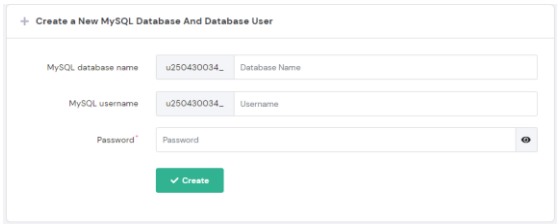
9.1 Set up database

A. Inside mb-sfms.com hosting, scroll down and navigate for Database section.

B. Select MySQL Databases



C. Create new MySQL Database

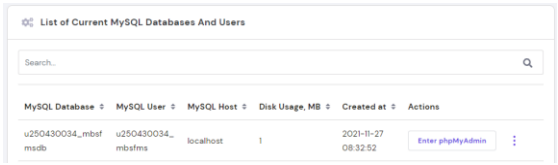


Fill the Database name, username, and password and press create.

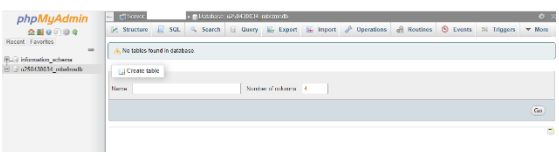
In this case,

MySQL database name: u250430034_mbsfmsdb
MySQL username: u250430034_mbsfms
Password: 5Hm^RWm^^

The newly created database will be listed on all MySQL Databases

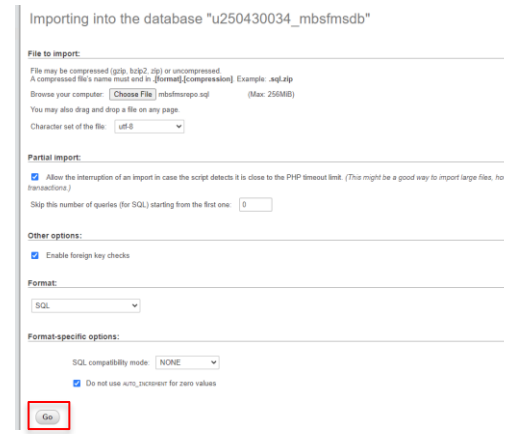


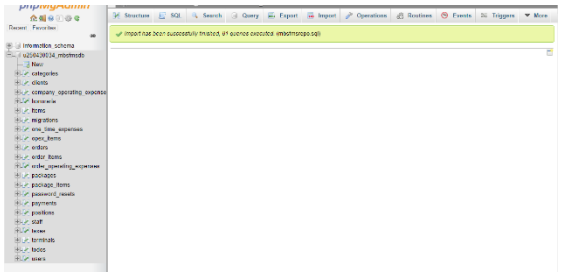
D. Press phpMyAdmin on the newly created database



E. Import Database

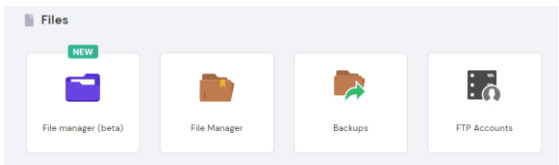
A. Press Import and choose file
B. Go to the project root directory `/sql` and select mbsfmsrepo.sql
C. Press Go



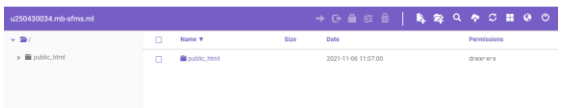


10. Manage files on hosting

Inside mb-sfms.com, navigate on Files section and select File Manager (brown folder)

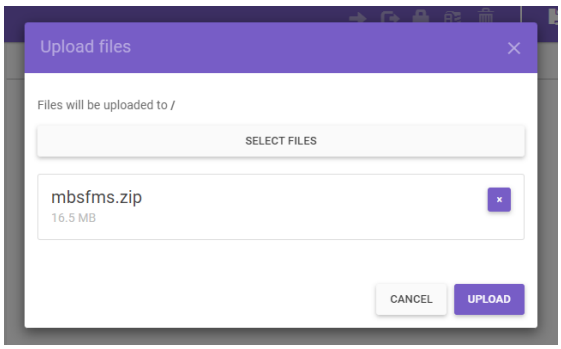


The page will redirect to file manager.

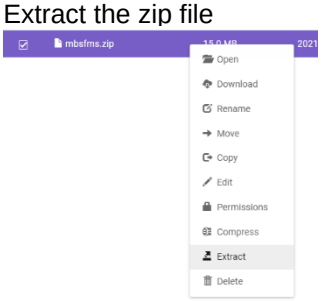


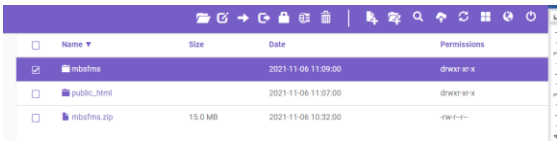
*note: do not remove the public_html folder

A. Upload the mbfms.zip file

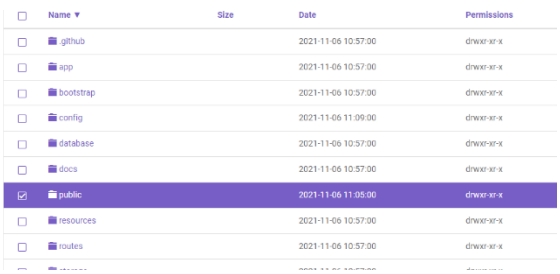


B. Extract the zip file

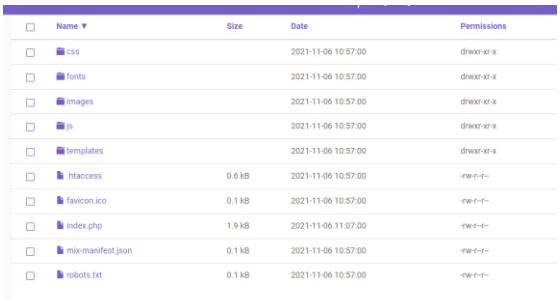




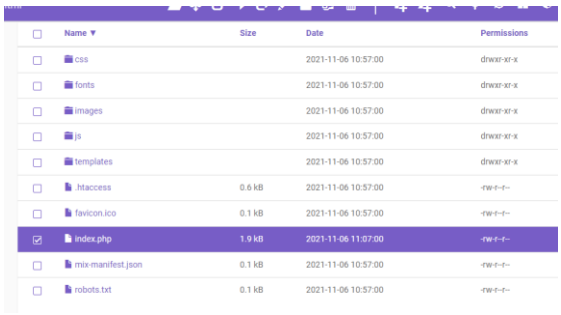
C. Go to mbsfms folder and navigate public folder



D. Move all content of public folder to public_html



E. Navigate and open index file



F. Modify the following

require __DIR__.'/../vendor/autoload.php';

to

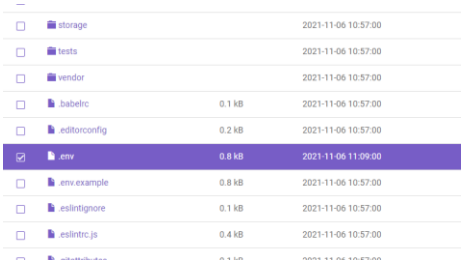
require __DIR__.'/../mbsfms/vendor/autoload.php';

$app = require_once __DIR__.'/../bootstrap/app.php';

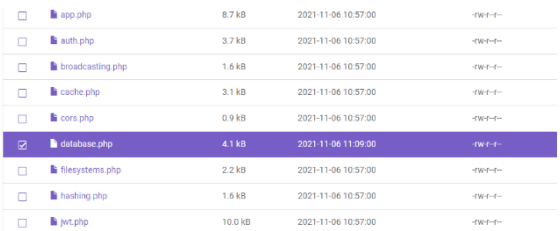
to

$app = require_once __DIR__.'/../mbsfms/bootstrap/app.php';

G. Navigate the .env file located on the mbsfms root folder



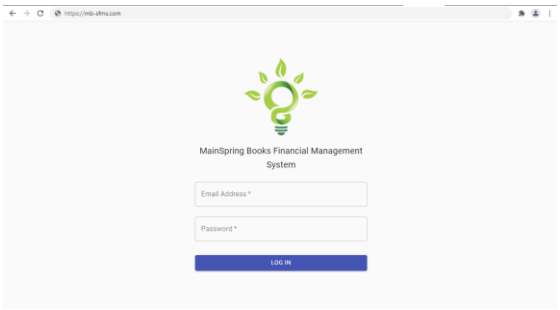
H. Change database settings on config/database.php



I. Modify the MySQL settings based on the newly created database

```
'mysql' => [
    'driver' => 'mysql',
    'host' => env('DB_HOST', '127.0.0.1'),
    'port' => env('DB_PORT', '3306'),
    'database' => env('DB_DATABASE', 'u250430034_mbsfmsdb'),
    'username' => env('DB_USERNAME', 'u250430034_mbsfms'),
    'password' => env('DB_PASSWORD', '5Hm^RWm^^'),
    'unix_socket' => env('DB_SOCKET', ''),
    'charset' => 'utf8mb4',
    'collation' => 'utf8mb4_unicode_ci',
    'prefix' => '',
    'strict' => false,//prev. true (3-27-2021)
    'engine' => null,
    'timezone'  => '+00:00'
],
```

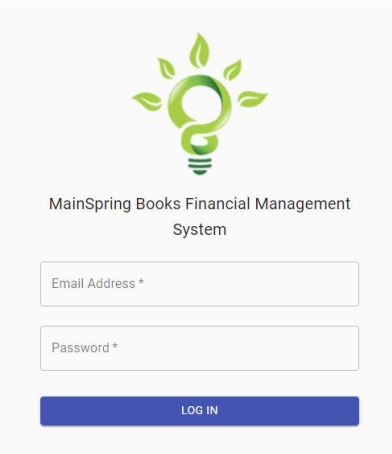
J. Test the web application



III. User Manual

1. System Login

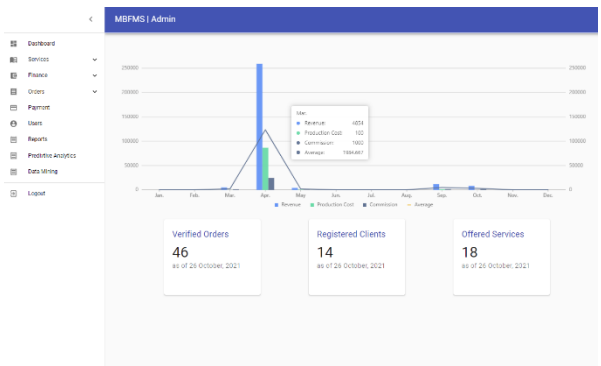
To start with the system, the user must login into Mainspring Books Financial Management System web application using the given email address and password. By pressing the login button, the system will validate the user's credentials.



2. System Dashboard

The dashboard shows a graph of verified monthly orders of the current year. It also has counters for verified orders, registered orders, offered services as of the current date.

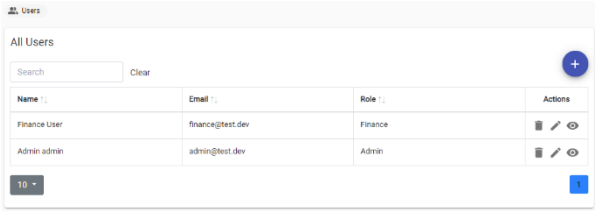
On the left side pane, it shows the main menu of the system. This menu can be toggled hide/show to easily navigate the features of the program.



3. Manage User

3.1. View All Users

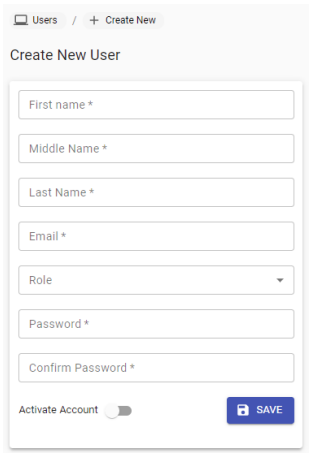
To view all users, user must click on *User* menu located at the left side bar menu.



This module contains list of users, Add new user, Remove, Edit, and View user buttons.

3.2 Add New User

The user must click on circular plus button located at *view all users module* to open the add new user form.
The user must fill out all fields in the form and select specific role to assign the user.
To activate the account of the newly created user, the Activate Account toggle should be turned On.
If all fields are filled, click save button to save all created data in to the database.



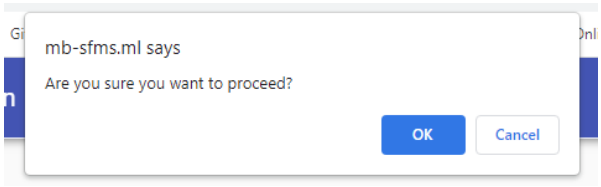
3.3 Edit User

On *view all users module*, select a specific user to modify and press pencil icon. The user will be redirected to edit user form page

To edit details of the user, modify desired fields or the activation status and press save button.

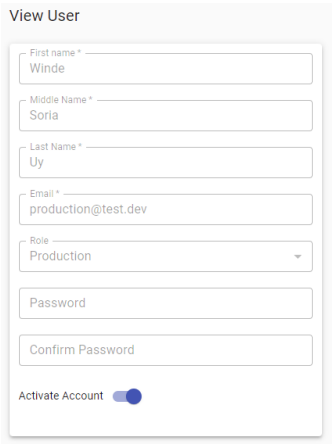
3.4 Remove User

On *view all users module*, select a specific user to remove and press trash button. The user will be asked a confirmation message to proceed, press OK to proceed, cancel to discard the process.



3.5 View Specific User

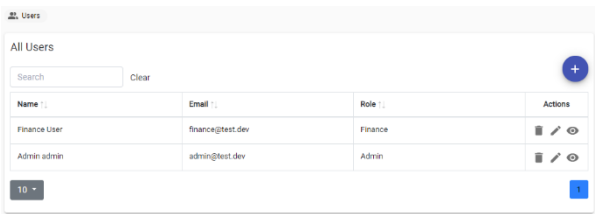
On *view all users module* , select a specific user to view and press eye button to open details of the user.

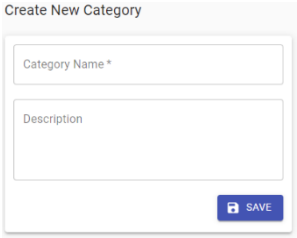


4. Manage Service Categories

4.1. View All Service Category

To view all service categories, user must click on *Services>Service Categories* menu located at the left side bar menu.





This module contains list of service categories, Add new, Remove, Edit, and View service category buttons.

4.2. Add New Service Category

The user must click on circular plus button located at *view all service categories module* to open the add new form.
The user must fill out all necessary fields in the form and press save button.

4.3. Edit Service Category

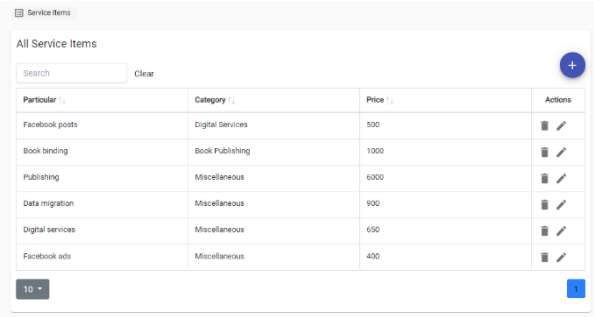
On *view all service categories module*, select a specific category to modify and press pencil icon. The user will be redirected to edit category form page
To edit details of the category, modify desired fields and press save button.

4.4 Remove Service Category

On *view all service categories module*, select a specific category to remove and press trash button. The user will be asked a confirmation message to proceed, press OK to proceed, cancel to discard the process

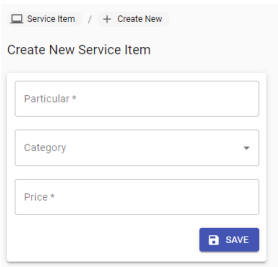
5. Manage Service Items

5.1. View All Service Category



To view all service Items, user must click on *Services>Service Items menu* located at the left side bar menu.
This module contains list of service items, Add new, Remove, Edit, and View service item buttons.

5.2. Add New Service Category



The user must click on circular plus button located at *view all service items module* to open the add new form.
The user must fill out all necessary fields and select service categories in the form and press save button.

5.3. Edit Service Item

On *view all service items module*, select a specific item to modify and press pencil icon. The user will be redirected to edit category form page

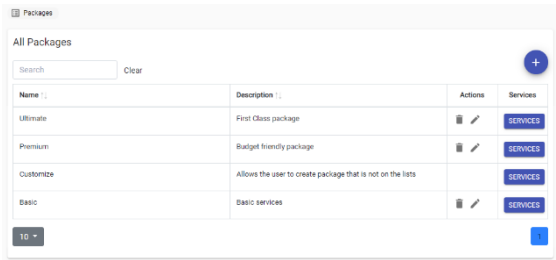
To edit details of the item, modify desired fields and press save button

5.4 Remove Service Item

On *view all service items module*, select a specific item to remove and press trash button. The user will be asked a confirmation message to proceed, press OK to proceed, cancel to discard the process

6. Manage Packages

In managing packages, the system allows the user to add, edit, and remove a package. Assigning services to a certain package is also part of this module.

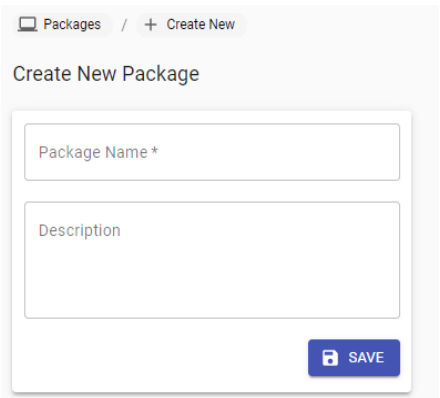


6.1. View All Packages

To view all packages, user must click on *Services>Packages* menu located at the left side bar menu.
This module contains list of packages, Add new, Remove, Edit, View package buttons, and services assignment

6.2. Add New Package

The user must click on circular plus button located at *view all package module* to open the add new form.
The user must fill out all necessary fields and press save button.



6.3. Edit Package

On *view all packages module*, select a specific package to modify and press pencil icon. The user will be redirected to edit category form page

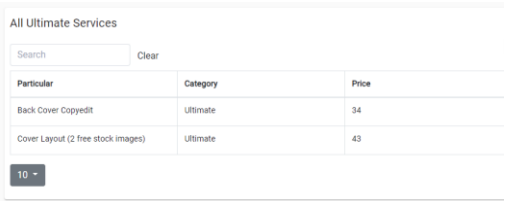
To edit details of the package, modify desired fields and press save button

6.4 Remove Service Item

On *view all packages module*, select a specific package to remove and press trash button. The user will be asked a confirmation message to proceed, press OK to proceed, cancel to discard the process

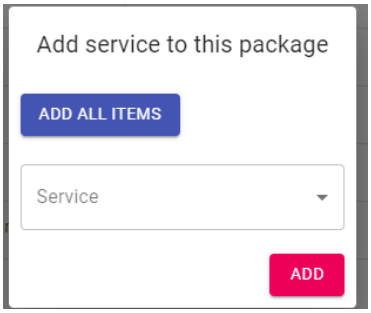
6.5 Assign Services to a Package

On *view all packages module*, select a specific package to assign services and press Services button. The user will be redirected to all services assigned to the selected package.



6.5.1 Add Services to package

To add services to a package, press the circular plus button and service selection will be prompted. To add all available services, press *Add All Items.* To add single service, select service on dropdown and press save button.



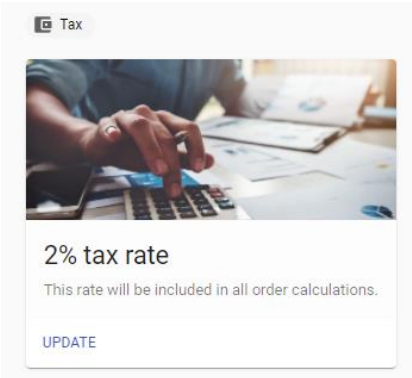
6.5.2 Remove Service to package

On list of assigned services of specific package, select service to remove and press trash icon.

7. Tax Rate Modification

This module allows the user to modify the base tax rate of company's sales. This will serve as default tax value in all orders. Tax rate can still be modified during costing of a specific order.

Modifying the tax rate doesn't affect the calculations of existing orders.



To update tax rate, go to *Finance* > *Tax* menu located at the left side bar menu. Press update button, fill in the field with percentage tax and press save button.

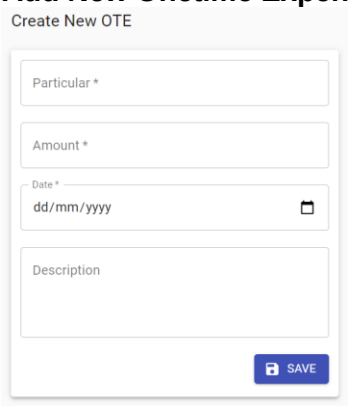
8. Manage One Time Expenses

To view all onetime expenses, user must click on *Finance>Onetime Expenses* menu, located at the left side bar menu.

The system allows the user to navigate existing one-time expenses using date range and search filters in this module. It also includes adding, removing and updating a certain onetime expense.

Onetime expenses can be printed upon clicking the print button. Print preview will be prompted before printing.

8.1 Add New Onetime Expense



The user must click on circular plus button located at *view all onetime expenses module* to open the add new form.

The user must fill out all necessary fields in the form and press save button.

8.2 Remove Onetime Expenses

On *view all onetime expenses module*, select a specific expense to remove and press trash button. The user will be asked a confirmation message to proceed, press OK to proceed, cancel to discard the process

8.3 Edit Onetime Expenses

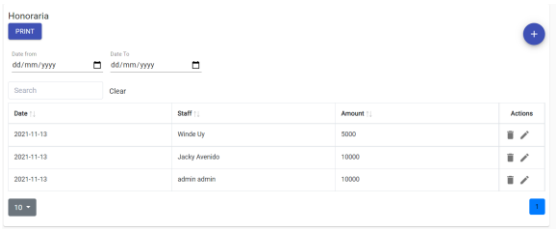
On *view all onetime expenses module*, select an expense to modify and press pencil icon. The user will be redirected to edit onetime expense form page

To edit details of the expense, modify desired fields and press save button

9. Manage Honorarium

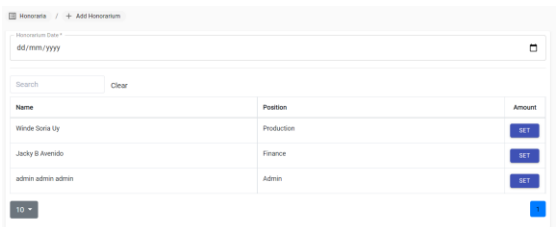
This module allows the Admin and Finance staff to navigate existing honorarium using date range and search filters. It also includes adding, removing and updating of a certain staff honorarium.

To view all honorarium transactions, user must click on *Finance>Honoraria* menu, located at the left side bar menu.



9.1 Add New Honorarium Transaction

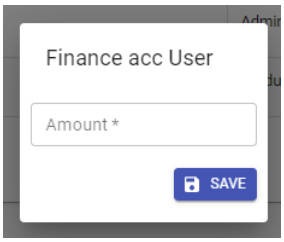
The user must click on circular plus button located at *view all honorarium transactions module* to open the setup module.



To add new honorarium, user must provide date of the honorarium given to the staff.

By selecting a staff, user must click on SET button and amount field will be prompted. The amount to be inputted on the field is the amount

of honorarium given to that staff on the specified date.

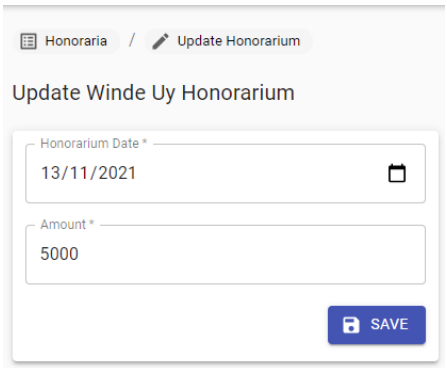


After setting honorarium on a single staff, the transaction will now be posted on the list of honoraria.

9.2. Edit Specific Honorarium

On *view all honorarium transactions module*, select a specific honorarium to modify and press pencil icon. The user will be redirected to edit honorarium form page

To edit details of the honorarium, modify desired fields and press save button.



9.3 Remove Honorarium

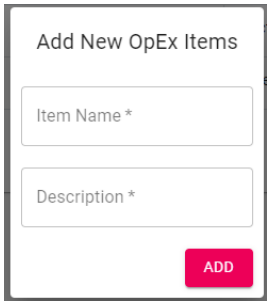
On *view all honorarium transactions module*, select a specific honorarium to remove and press trash button. The user will be asked a confirmation message to proceed, press OK to proceed, cancel to discard the process.

10. Manage Operating Expenses Items

To view all operating expenses items, user must click on *Finance>OpEx Items* menu, located at the left side bar menu.

This module allows the user to add, update, and remove operating expense particulars that the company frequently used.

10.1 Add New Operating Expenses Item



The user must click on circular plus button located at *view all operating expenses items module* to open the add new form modal.

The user must fill out all necessary fields in the form and press Add button.

10.2 Edit Operating Expenses Item

On *view all operating expenses module*, select a specific particular to modify and press pencil icon. Modal will be prompted with the particular details.

To edit details of the particular, modify desired fields and press update button.

10.3 Remove Operating Expenses Item

On *view all operating expenses module*, select a specific particular to remove and press trash button. The user will be asked a confirmation message to proceed,

press OK to proceed, cancel to discard the process.

11. Manage Company Operating Expenses

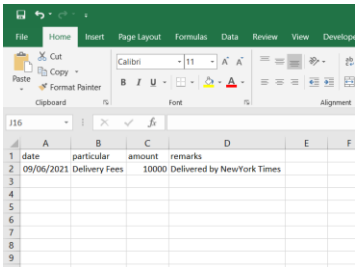
To view all operating expenses items, user must click on *Finance>Company Operating Exp.* menu, located at the left side bar menu.

This module allows the user to navigate all company operating expenses thru date range and search filter. Generate the report thru printing and excel file reports. Manage operating expenses thru adding, removing and updating.

11.1 Import Company Operating Expenses

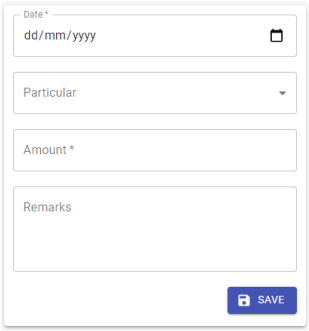
If the company wants to migrate their existing expenses, the system provides file import thru CSV.

A. Click Import button, a modal will be prompted.
B. Click *"Download template here"* to download template
Note: Do not modify the headers of the template, the import will not be successful.
C. Fill the CSV file with previous company operating expenses.



D. Go back to import modal, upload the filled CSV file.
E. Wait for the upload to be successful

11.2 Add New Company Operating Expense



The user must click on circular plus button located at *view all company operating expenses module* to open the add new form.

The user must fill out all necessary fields and select particular on provided dropdown in the form and press Save button.

11.2 Edit Company Operating Expense

On *view all company operating expenses module*, select a specific particular to modify and press pencil icon. The user will be redirected to edit form.

To edit details of the expense, modify desired fields and press save button.

11.3 Remove Company Operating Expense

On *view all company operating expenses module*, select a specific expense to remove and press trash button. The user will be asked a confirmation message to proceed, press OK to proceed, cancel to discard the process.

11.4 Download Company Expenses as Excel File

On *view all company operating expenses module*, press *Download as Excel* button. The data will be based on what is displayed on the

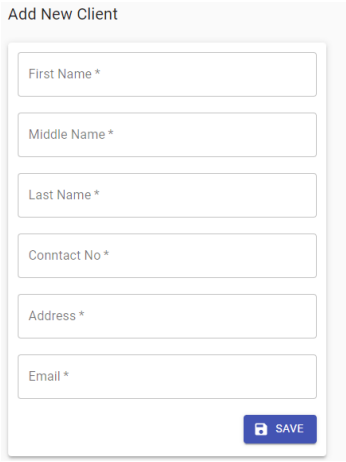
module date filtered or non-date filtered.

12. Manage Clients

To view all registered clients, user must click on *Orders>Clients.* menu, located at the left side bar menu.

In managing clients, the system allows the user to add, remove, update details of clients. It can also monitor client's payable balances.

12.1 Add New Client



The user must click on circular plus button located at *view all clients module* to open the add new form.

The user must fill out all necessary fields in the form and press save button.

12.2 Edit Client

On *view all clients module*, select client to modify and press pencil icon. The user will be redirected to edit client form page

To edit details of the client, modify desired fields and press save button

9.3 Remove Client

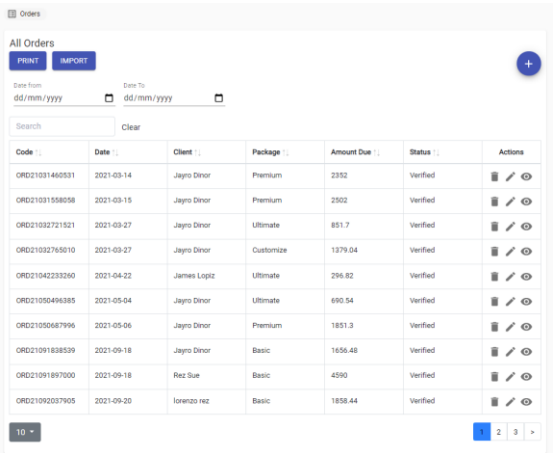
On *view all client module*, select a specific client to remove and press trash button. The user will be asked a confirmation message to proceed, press OK to proceed, cancel to discard the process.

Note: Client can only be deleted if it has 0 balance and has no transactions on the system yet.

13. Manage Orders

To view all orders, user must click on *Orders>Orders.* menu, located at the left side bar menu.

In this module, the system allows the user to add orders, update order details, monitor order status, perform order costing, and view specific order.



12.1 Import Order

To import previous order transactions, the system requires 3 CSV files in order for the import to be successful. The templates are provided in every files. The imports are separated by Basic Order Details, Order Services, and Order Expenses.

The files should be connected thru the Basic Order Details ID that the user must provide. These IDs are just reference of each order on each file, these IDs will not be used as IDs on the orders when imported on the system.

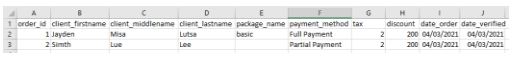
A. Click Import button, a modal will be prompted with 3 upload fields.

B. Click the 3 labels *"Download template here"* to download template of Basic Order Data, Order Services, and Order Expenses.

Note: Do not modify the headers of the template, the import will not be successful.

C. Fill the CSV file with verified.

Basic order details sample template



	A	B	C	D	E	F	G	H	I	J
1	order_id	client_firstname	client_middlename	client_lastname	package_name	payment_method	tax	discount	date_order	date_verified
2	1	Jayden	Misa	Lutsa	basic	Full Payment	2	200	04/03/2021	04/03/2021
3	2	Simth	Lue	Lee		Partial Payment	2	200	04/03/2021	04/03/2021

The order_id 1and 2 will be used later on the other 2 CSV files as reference to these orders.

The client details (first name, last name, and middle) and package name will be checked first into the database if existed else the data will be created.

Order Services sample template

	A	B	C	D
1	order_id	service_name	service_org_price	service_markup
2	1	Dbook Publishing	5000	2000
3	2	Digital services	6000	600
4	2	Facebook ads	3000	200

The order_id shown above are the order_id from the Basic Order Details CSV file. In order for the import to be successful, the order id must properly correspond to the basic order details.

Order id may occur multiple times in Order services.

Service name will be check into the database if existed to prevent duplications of data.

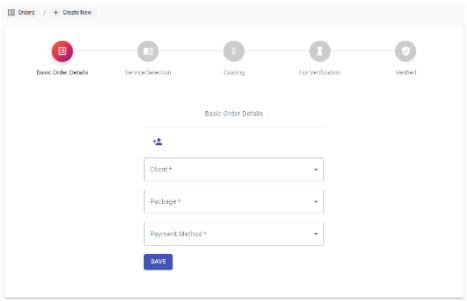
Order Expense sample template

	A	B	C
1	order_id	particular	cost
2	1	Commission	6000
3	1	Food Exoenses	3000
4	2	Transportatin Fee	5000

Order id must also correspond properly to the order detail id.
Order id may also occur multiple times in Order services.

D. Go back to import modal, upload the filled CSV files.
E. Wait for the upload to be successful

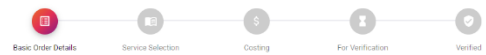
12.2 Add New Order



The user must click on circular plus button located at *view all orders module* to open the add new form.

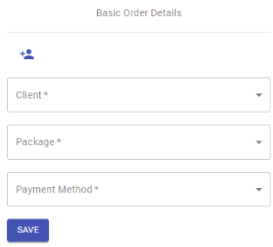
The order form comes with different steps.

- The **Basic Order Details**, where the user inputs the basic details of the order.
- The **Service Selection**, services will be selected based on the clients package preferences.
- The **Costing**, where the costing of services and expenses of the order are performed.
- The **For Verification**, where the summary of the order is displayed for checking before verifying the order.
- The **Verified** step, specifies that the order is already verified.



Each step will turn red if the user completed the required field of the step.

12.2.1 Basic Order Details

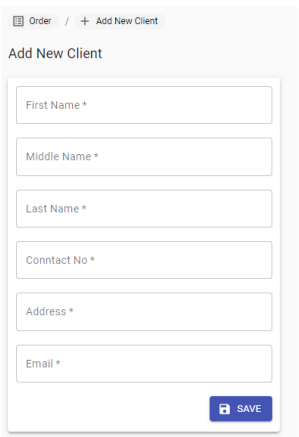


When entering the basic details of an order, the user must specify the client, package, and payment method.

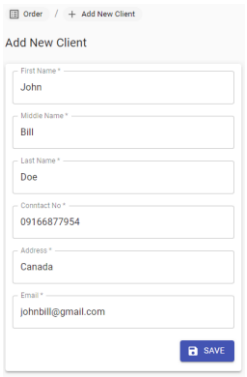
Existing clients are selected thru client dropdown.

If the client has no record yet in the system, user must fill in the add new client form upon clicking the person with plus icon.

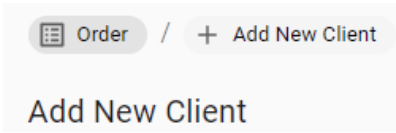
The user will be redirected to add new client form.



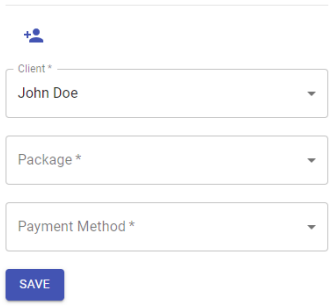
After filling up all required fields, click save.



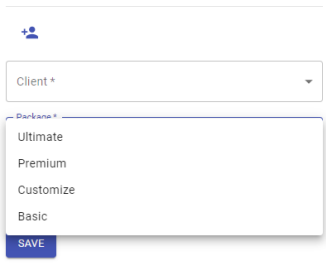
To go back to Basic Order details form, click the Order breadcrumbs on upper left.



The currently created client will be automatically fill the client dropdown.



Package list are generated from package masterdata.

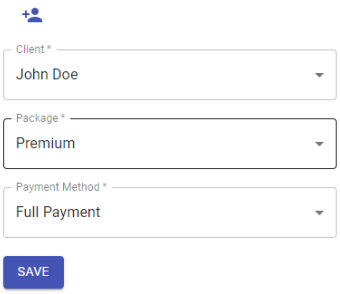


Services will be different on each package.

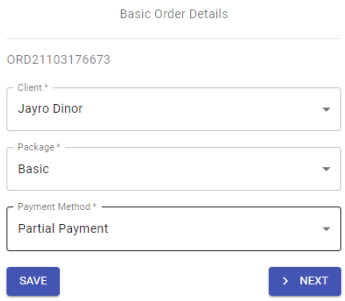
Payment method selections are Full Payment and Partial Payment. If Full Payment is

selected, the client must pay the charges in full amount before verifying the order status. Orders that are in partial payment can be verified if the client only pays the partial amount.

If all required fields are completed, press Save button.



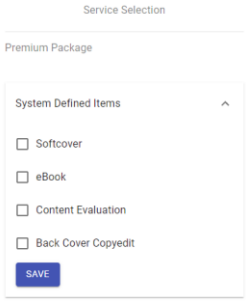
The system creates an order code, and the user is ready to go on to the next phase of the order process.



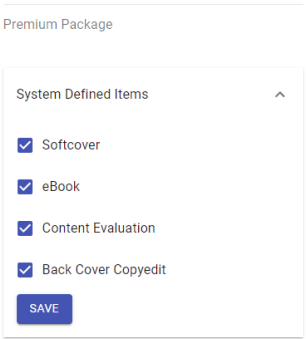
Press Next to proceed to the next stage of adding new order.

12.2.2 Services Selection

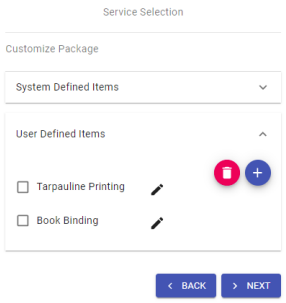
Service selection step, the system allows the user to select services based on the selected package.



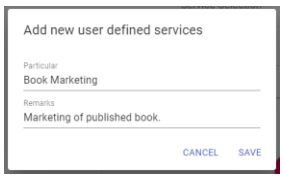
The system shows services assigned to the package if the client chooses conventional packages. Select all services that the client wants to include and press save button.



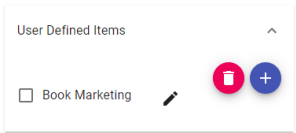
If the customer opts for customization, the system allows the user to construct services based on the client's preferences. This is only applicable if the user selected customized package.



To add customize service, press circular plus button. Add new user defined service form will be prompted.

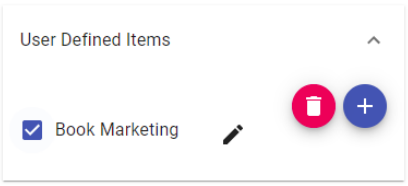


Press save to add new user defined service. The added service will be listed on user defined items.



To edit a user defined service, select specific service and press pencil icon. Update form with selected data will be prompted. Modify needed fields and press save button.

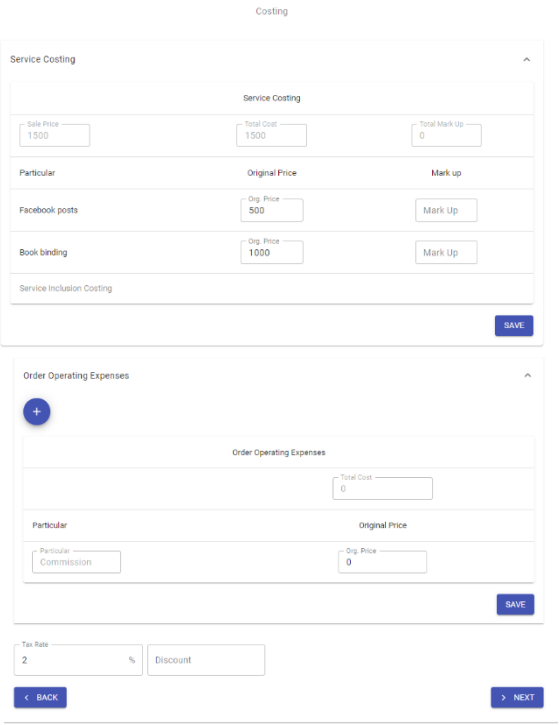
To remove user defined service, select services or mark as check the services to remove and press trash button.



If all clients preferred services are selected and saved, the user is now ready for the next phase which is Costing. Press Next button to proceed.

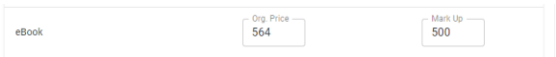
12.2.3 Order Costing

Costing step, allows the user to configure the costing of services and expenses covered on the project. The original prices and markups of selected services can be modified. The default cost of the original price will be based on the masterdata of services.



12.2.3.1 Setting service costing

On service costing section, select service to set original price and markup. Fill the original price and markup fields on the selected row.

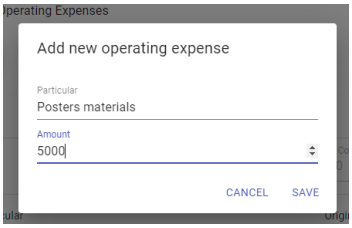


Press Save button to save the changes.

12.2.3.2 Setting order operating expenses cost

Commission expense is a default operating expense since it is common to all orders.

To add new order operating expense, press the circular plus button located at Order Operating Expenses section. Operating expense modal will be prompted with Particular and amount fields. Fill the necessary fields and press save button.

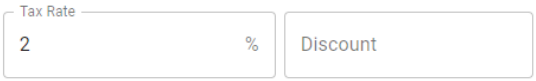


To modify the operating expenses, select a particular and modify on particular or original price field and press Save button.

To remove an expense, select a particular and press the trash button on a specific row. A confirmation message will be prompted, press OK to proceed.

12.2.3.3 Setting other calculation

To modify tax percentage and discount, enter the percentage amount of tax or amount of discount. After setting the amounts, a confirmation message will be prompted, press OK to proceed, the data will be saved automatically.



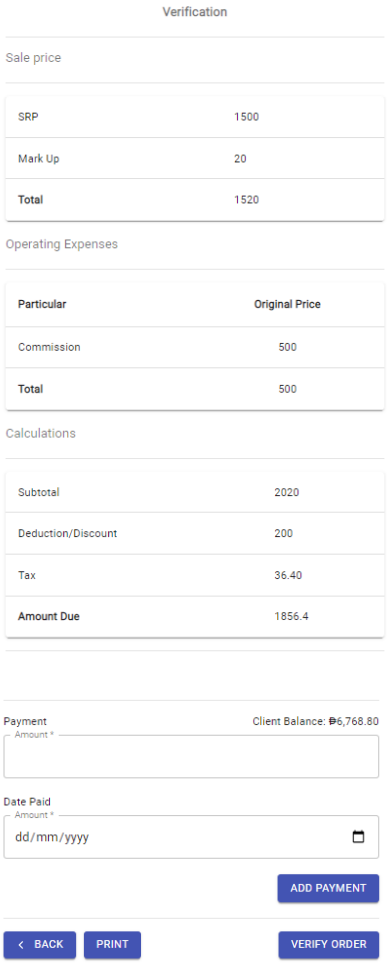
To proceed to the next stage of the order which is the verification, press Next button.

12.2.4 Order summary/ Order verification

The order will be for verification if the user finished the order's costing.

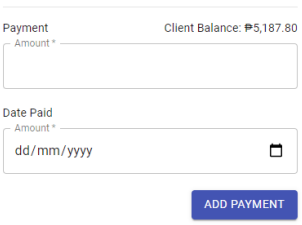
The order's computation summary is displayed in the for-verification stage. It also allows the user to enter payment amounts from the customer, either in full or in part.

The printing of invoice is also generated in this step.



12.2.4.1 Adding Payment from order Module

On For-verification stage of order module, specifically payment section, the client charge balance is displayed on the upper right. To add payment of client, enter amount paid and date of payment then press Add payment. The amount entered will be debited to client's account.



12.2.4.2 Verifying Order

After reviewing and completing all details of the order, the order is now ready for verification. To verify order press, ***Verify Order*** button on the bottom right of the module.

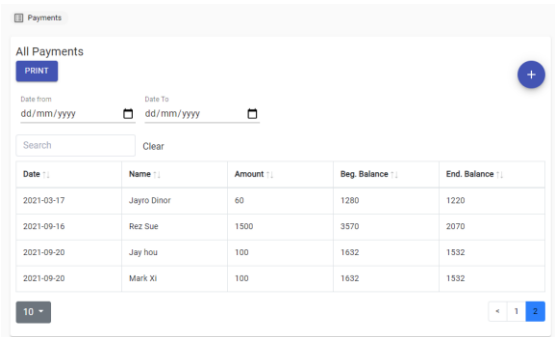
If the order is successfully verified, the button will change its appearance and the order is now complete.

14. Manage Payments

To view all client's payments, user must click on *Payments* menu, located at the left side bar menu.

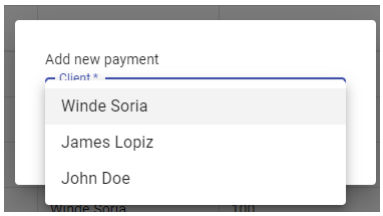
This module allows the user to navigate all payment transactions of the clients thru date range and search filter. This can also generate the report thru printing.

The listing of payments can also track the remaining balances of the client.

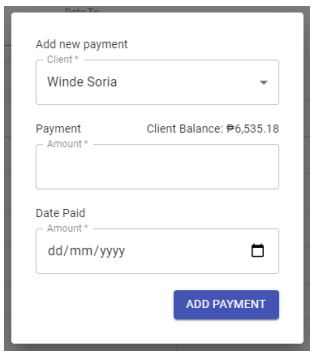


14.1 Adding new payment

To add new payment, user must click the circular plus button located at *all payments modules,* a modal will be prompted with client listings.



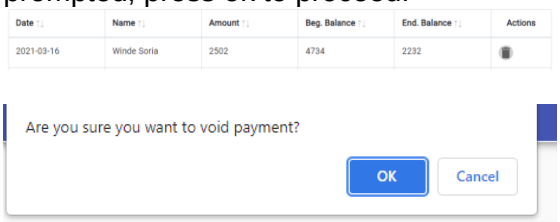
After selection of client, payment form will be displayed with current client charge balance.



Enter the amount and date of payment and press Add Payment button.

14.2 Void payment

To void payment, user must select payment to void, press the trash button and a confirmation message will be prompted, press ok to proceed.



The amount of voided payment will be added back to the client's credit balance.

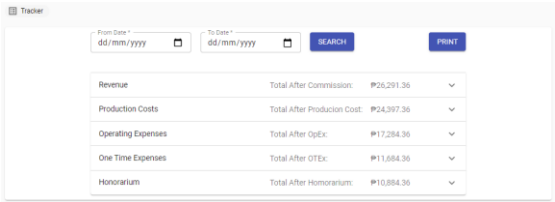
15. Tracker Report

To view reports, user must click on *Reports* menu, located at the left side bar menu.

Tracker Report module tracks the revenue of the company. This will generate report of Revenue, Production cost of verified orders,

Company operating expenses, One-time expenses, and Honorarium.

The report can be generated from date range filter.

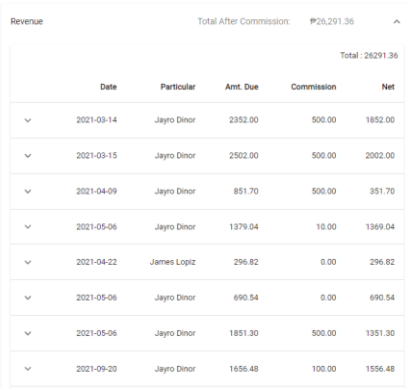


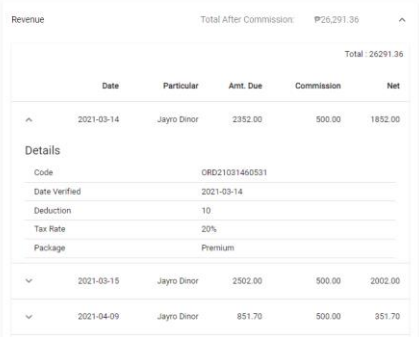
The tracker report calculates the net income of the company after all total deductions of all expenses.

- **Total After Commission** (Revenue – Commission).
- **Total After Production cost** (Total After Commission - Total Production Cost
- **Total After Operating Expenses** (Total After Production cost - Total of Operating Expenses)
- **Total After One Time Expense** (Total After Operating Expenses – Total of One Time Expense)
- **Total After Honorarium** (Total After One Time Expense - Total of Honorarium)

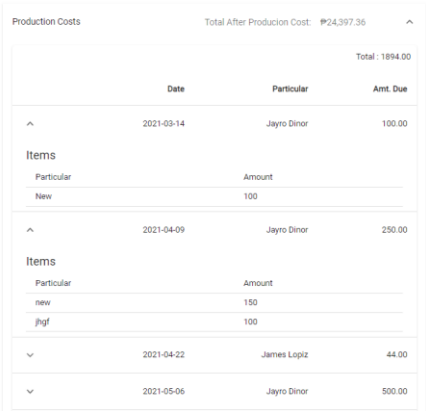
To view the contents of each report, expand the account title, and rows of transactions will appear, which may be extended by the report's individual details.

A. Revenue reports contains list of verified orders.

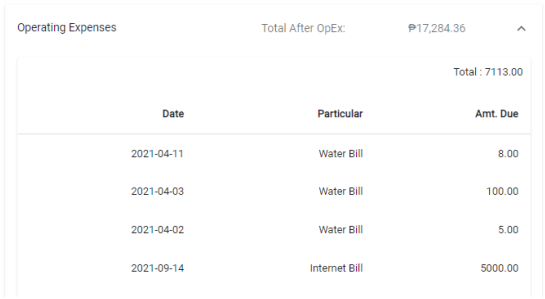




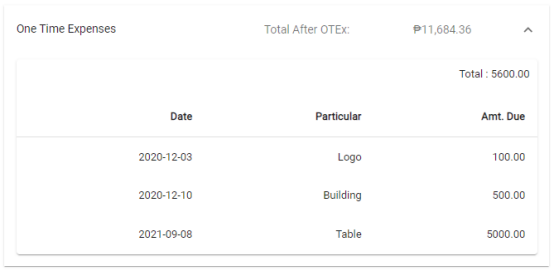
B. Production Cost report provides cost of production particulars of each verified orders.



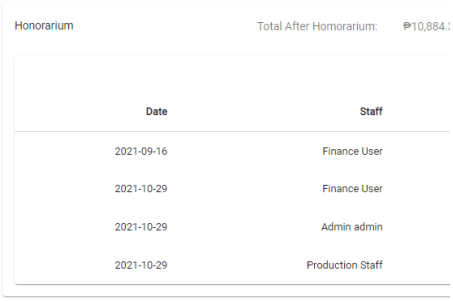
C. Operating Expenses provides list of regular expenses made by the company.



D. One Time Expenses provides reports of expenses that arise from non-operating activities



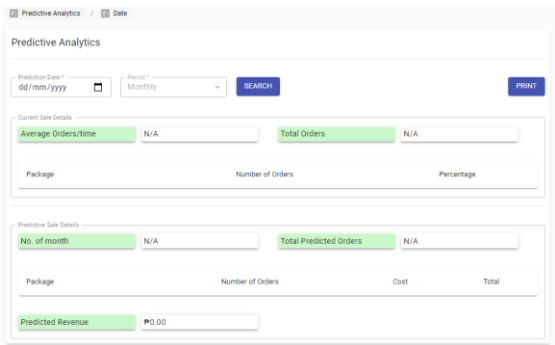
E. Honorarium Report provides expense report of honorarium given to company staffs.



16. Predictive Analytics

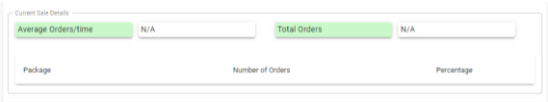
To view predictive analytics, user must click on *Predictive Analytics* menu, located at the left side bar menu.

Predictive analytics feature predicts company's revenue on their preferred prediction date. The prediction is based on the company's average revenue since the first verified order on the system. It will calculate the average sales per month and these average sales will be used to predict the revenue.

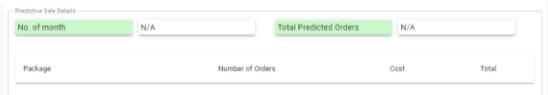


Predictive Analytic module contains preferred predictive time form, The predictive date and monthly average period.

Current sale details section provides data from existing sales. The data are based on the first sale up to the last sale. This will be the basis of data for prediction.

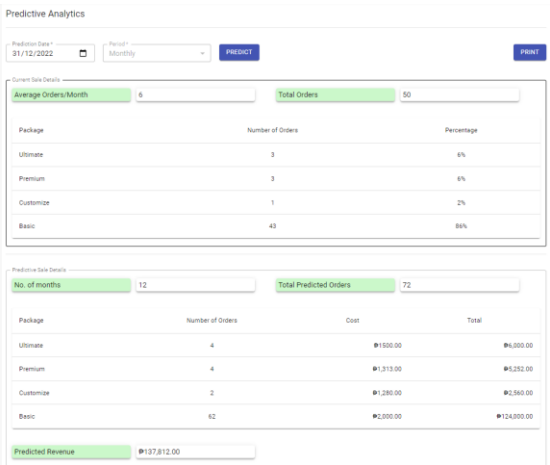


Predictive Sale Details section provides prediction data from the data provided on current sale. This prediction range covers the current date up to the prediction date provided by the user.



The **predicted revenue** is calculated from the predicted data.

To generate prediction, input prediction date and press predict button.

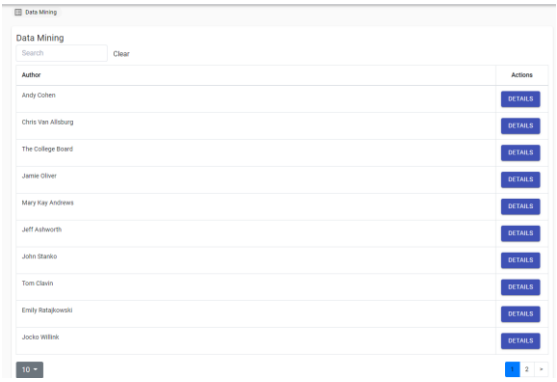


17. Data Mining

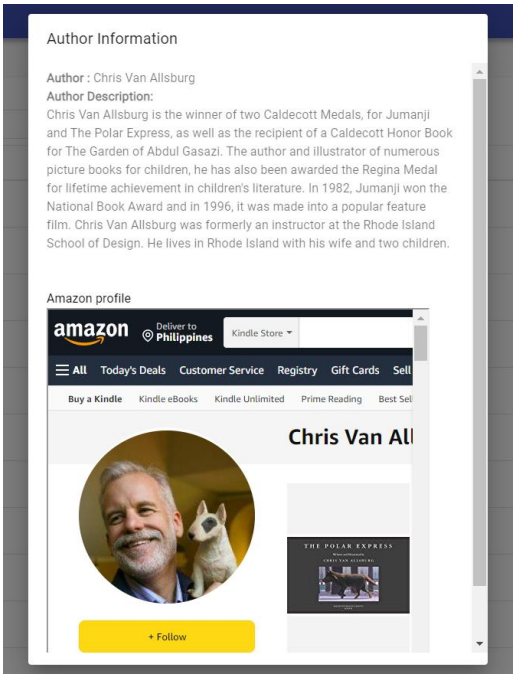
To view Data mining, user must click on *Data Mining* menu, located at the left side bar menu.

Data mining is a process used by companies to turn raw data into useful information.

In this application, the system fetches list of authors available on Amazon Books website.



To view details of an author, user must click Details button. Author detail will be prompted on the screen.



The system retrieves the author name, author description which contains the background and some personal details of the author, and author profile on amazon, which can be navigated through the system.