UNIVERSITY OF THE PHILIPPINES
OPEN UNIVERSITY

Master of Information Systems

MICHAEL ANTHONY V. YGNACIO

CHEMICAL INVENTORY SYSTEM FOR CHEMISTRY DEPARTMENT OF LIBERAL ARTS COLLEGE

Thesis/Dissertation Adviser:
Mari Anjeli L. Crisanto, MIT (UKM)
Faculty of Information and Communication Studies

Date of Submission
1 June 2022

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

Available to the general public	Yes
Available only after consultation with author/thesis/dissertation adviser	Yes
Available only to those bound by a confidentiality agreement	Yes

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

Disclaimer

This project is submitted to the Faculty of Information and Communication Studies in partial fulfillment of the requirements for the degree Master of Information Systems at the University of the Philippines – Open University, Los Baños, Laguna. It is the product of my own work except where indicated in the text. The project report or any portion thereof including the source code or any section may be freely copied and distributed provided that the source is acknowledged.

Acceptance Sheet

This project proposal entitled “Chemical Inventory System for Chemistry Department of Liberal Arts College” submitted to the Faculty of Information and Communication Studies, University of the Philippines – Open University, Los Baños, Laguna in partial fulfillment of the requirements for the degree Master of Information Systems is hereby accepted.

_____
Adviser

_____
Date

_____
Program Chair

_____
Date

_____
Dean
Faculty of Information and
Communication Studies

_____
Date

Acknowledgement

I would like to acknowledge and thank my adviser, Mari Anjeli L. Crisanto, for her guidance in completing this special project. Special thanks as well to my professors in prior subjects that have helped build the foundation for the components in this project proposal and system development.

I also would like to sincerely thank the client for accepting this project proposal and working in close collaboration during system implementation. I also would like to thank the test users for participating in the system evaluation. And lastly, I would like to extend my gratitude to my family for their continued support throughout this time.

Abstract

Within the Chemistry department of a given liberal arts college in the US, chemical inventory has been managed through a traditionally manual process using spreadsheets and documents that quickly became unstructured, inefficient, and disorganized. As such, this project aimed for the development of a custom-built, open-source, web-based Chemical inventory system. This project has been intended to help streamline the client's chemical management process and migrate physical or paper-based data into a more efficient and structured digital environment. This project also intended to serve as a pioneer in IT modernization initiatives through increased adoption of software-based solutions within the department.

The proposed system has been developed using a distributed logical architecture consisting of separate front-end (web), back-end (application), and database (data) layers. The front-end has been built as a Single Page Application (SPA) using Angular/Typescript. The back-end has been built as a Java Spring Boot web services application with various RESTful API endpoints for serving and accepting structured JSON data. The relational database has been built using PostgreSQL. System components have been deployed and hosted on the Amazon Web Services (AWS) cloud infrastructure.

The client is a Chemistry Professor with a bachelor's degree and Ph.D. in Chemistry who is currently working at the Presbyterian College, a liberal arts college in

South Carolina, USA. Also, given the broad scope of a liberal arts college compared to a research university, the Chemistry department has a fairly limited budget and lower priority being given to the consideration of software-based solutions.

This project has been developed using the Kanban framework that focuses on flexible, iterative, visualized, and transparent methods and practices. There have also been consultations, and reviews of online references for the design of website templates, UI/UX interfaces, data modeling, database schema, and API endpoints. Also, the Kanban framework has allowed for continuous collaboration with the client for design, demos, testing, and feedback throughout the project duration. These helped assess the development progress and software output in terms of matching the client's needs and expectations.

Table of Contents

I. Problem Domain

A. Statement of the Problem

Currently, within the Chemistry department of a given liberal arts college, there has been no dedicated software system for managing chemical inventories. Chemistry professors, laboratory assistants, and students have been following manual workflows using Excel spreadsheets and other documents for recording and tracking chemicals that are ordered, delivered, stored, consumed, and disposed of for academic research and instructional purposes, and activities. Also given the broad scope of a liberal arts college compared to a research university, the Chemistry department has a fairly limited budget and lower priority being given to the consideration of software-based solutions. This has led to the following issues and concerns:

- Use of Excel spreadsheets, Word documents, and notes to record and track chemical containers that are ordered, delivered, stored, consumed, and disposed of.
- Duplication of the same information as well as disconnection of related information across different users.
- Loss of information when skipping or missing manual updates.
- Multiple versions of documents are being created and passed around.

- Difficulties in analysis and reporting due to data inconsistencies from the lack of a standard, well-defined and structured format.
- Manual processes ultimately become inconsistent, inefficient, and disorganized over time.

Also, having previously used commercial products for chemical inventory on a prior doctorate study in a different institution, the client has felt the need for a similar capacity in the current situation but also addresses past concerns such as the following:

- Outdated user interface that feels clunky and cumbersome to use.
- Difficulty in navigation due to the inclusion of other product features outside of chemical inventory.
- External dependency for barcode generation from another separate team/process/system.
- Lack of support for customization due to being a packaged product.
- Failure of users to comply with laboratory processes due to the lack of interest in working with the system.

B. Background and Objectives of the Project

This project proposal has been initiated by a combination of various factors such as the following:

- Institutional gaps in the form of limited budget and priority for the adoption of commercial software products.
- Proposed enhancements on the shortcomings of previously used systems from a prior different institution.
- Client initiative for the introduction of an inventory system to address both current gaps and previous pain points.

Given the above-mentioned concerns, this proposal aimed for the following:

- Introduce a chemical inventory system to help streamline the chemical management process.
- Provide a single access point for all users to view and manage inventory data.
- Convert physical or paper-based data into a more efficient and structured digital storage environment.
- Eliminate extra work effort due to overhead from duplication, redundancy, and clutter from using scattered and disorganized documents.
- Provide a structured collection of data that can be utilized for analysis and reporting needs.

- Provide capabilities to easily link and associate containers, locations, reports, and orders as part of the general workflow.

C. Significance and Scope of the Project

This proposal for a software development initiative has been of importance to the client given that the project:

- Offers a unique opportunity to transform workflows resulting in increased work efficiency and productivity.
- Serves as a pioneer system as part of IT modernization initiatives through the development and adoption of a software-based solution within the Chemistry department.
- Opens up possibilities for exploring technical solutions for other academic problems with the prospects for higher levels of support, schedule, and budget.

This proposal shall also be confined to the following scope:

- Limited to the core functions of managing chemical inventory (e.g. containers, locations, barcodes) in an academic laboratory environment and excludes the broader scopes of Environmental, Health, and Safety (EHS) software packages as in industrial cases.

- Private, internal use within the academic institution and with initial use/adoption for a single professor's (client) areas of concern.
- Development and infrastructure scope shall be limited based on the cost restrictions considering this project will mostly be complimentary.

D. Documentation of Existence and Seriousness of the Problem

As previously mentioned, there are no current systems used for inventory management leading to various problems mentioned in the sections above. One other professor who has been handling inventory for the longest time in the institution has been following manual processes for over 30 plus years. Throughout this decades-long period where other professors and lab personnel have also become involved, there have been countless documents, copies, versions, and file formats/extensions that were utilized. These include for example - MS Excel (.xls, .xslx), MS Word (.doc, .docx, .rtf), Text documents (.txt, .csv, no extension), PDF, and also physical/paper-based notes to name the most common ones. Other professors, such as the client, have had difficulties enacting such processes given the unstructured and disorganized methods and documents. Needless to say, it should be noted that these documents are considered confidential and thus not allowed to be referenced or disclosed outside of the institution.

II. Review of Existing Alternatives

To cope with the difficulties of managing chemical inventory manually, the client previously maintained a separate personal copy of the inventory data in Excel, apart from any other documents maintained by others in the lab. As previously mentioned, multiple copies led to inefficiency and disorganization but this did offer some form of control to the owner given they owned and managed their respective documents. The client and other professors mostly relied on their own individual documents and would seldom have had to consult with others or check the containers and/or locations directly to verify.

From a solutions standpoint, there were also other possible alternative solutions for an inventory system besides custom application development. The first alternative was the procurement of a commercial off-the-shelf (COTS) product. This refers to a software package with pre-defined, ready-made standard features that align with most use cases for different clients across various companies and institutions. From the broad domain of Environmental, Health, and Safety (EHS) solutions were the following examples of various commercial products: One, Chematix which does not have a free version and does not offer a free trial. Its user interface also already looks and feels outdated. Two, ChemInventory which has a good feature set, has better UI/UX, and has free and paid versions but with corresponding limits. It also appears to be powered by PHP through a traditional single web application. Three, Chemical

Safety EMS also has a good feature set, does not have a free version but has a free trial and with higher pricing. Considering the client's position, using commercial software required additional budget allocation and higher prioritization of IT initiatives versus other departmental concerns. Also, this needed broader support and participation of multiple professors and laboratories to justify software investment.

The second alternative was through the customization of shared spreadsheet documents such as via Google Spreadsheets or Microsoft Excel 365. However, this required certain degrees of configuration to make spreadsheets effectively function as a software application for tracking inventory. Shared workbooks are still limited to a grid-based (rows and columns) user interface for viewing and manipulating data. Also, it doesn't natively support other features such as searches/lookups, navigation, directory, and notifications, among others.

Given the proposed solution for a custom-built application, the project has implemented the following industry best practices:

- Adopted an iterative development methodology to better manage schedule, scope, quality, risk, and delivery.

- Utilized a distributed systems architecture Proof-of-Concept (POC) for reliability and scalability. Transitioned to a single server deployment to stay within cost limitations.
- Leveraged virtual infrastructure using cloud resources for resource usage efficiency and cost-effectiveness, such as starting small on server sizing and paying only for resources used.
- Implemented software coding best practices such as but not limited to:
    - REST (Representational State Transfer) principles for API services
    - KISS (Keep It Simple Silly) principle for code simplicity
    - DRY (Don't Repeat Yourself) principle for code reusability
    - Unit testing for code quality
    - Documentation for improved readability and knowledge sharing
    - Separation of concerns for modularity and ease of maintenance
    - Combination of Objected-Oriented (OO) design and Functional programming (FP) paradigms
    - Security through the principle of least privilege, temporary access credentials, and AWS cloud security recommendations.

III. Project Details

A. Special Features

On top of the core, standard system features for container management and location management, the completed inventory system has included the following special features that were either not supported or limited in prior systems and/or software alternatives. As such, the features below have been evaluated against other products such as Chematix and ChemInventory.

First, web notifications, for critical events such as low inventory reminders, scheduled weekly inventory reports, and container audit/lifecycle updates. These notifications shall help the client to continually keep track of the state of their inventory. None of the 2 products support this feature.

Second, barcode printing and scanning capabilities have been integrated within the inventory system instead of being an external dependency. Chematix does not support this, which meant the client previously made separate requests for barcode stickers from a different team using a separate barcode system.

Third, an inventory dashboard that provided summaries, overviews, numerical figures, and audit trails for key inventory metrics and data through applicable graphs, tables, or visualizations. This has helped provide a quick,

general depiction of the state of the inventory in a single view. None of the 2 products support this feature.

Fourth, site modernization through the utilization of UI/UX templates with a modern look and feel such as via Material design and Porto Admin web template. This also involved the adoption of more modern web technologies and frameworks such as Angular/Typescript for the front-end and RESTful web service APIs for the back-end. Chematix is still based on an old, outdated user interface while ChemInventory is still based on a traditional PHP web application style. This proposal aimed to develop a simple but intuitive UI/UX experience that will keep users engaged and also conform to inventory process compliance.

Fifth and lastly is the inclusion of CAS registry integration for looking up and displaying supplementary chemical information. The system communicates with official CAS sources (https://commonchemistry.cas.org/) after having made an official request submission and having been granted access to its public API endpoints.

B. Business Process Flow

The diagrams below illustrate the high-level business process for managing container inventory throughout the course of laboratory activities. This shows the connections between orders, barcodes, containers, locations, and reports within the process.

Figure 3.0 - Initial Business Process Flow

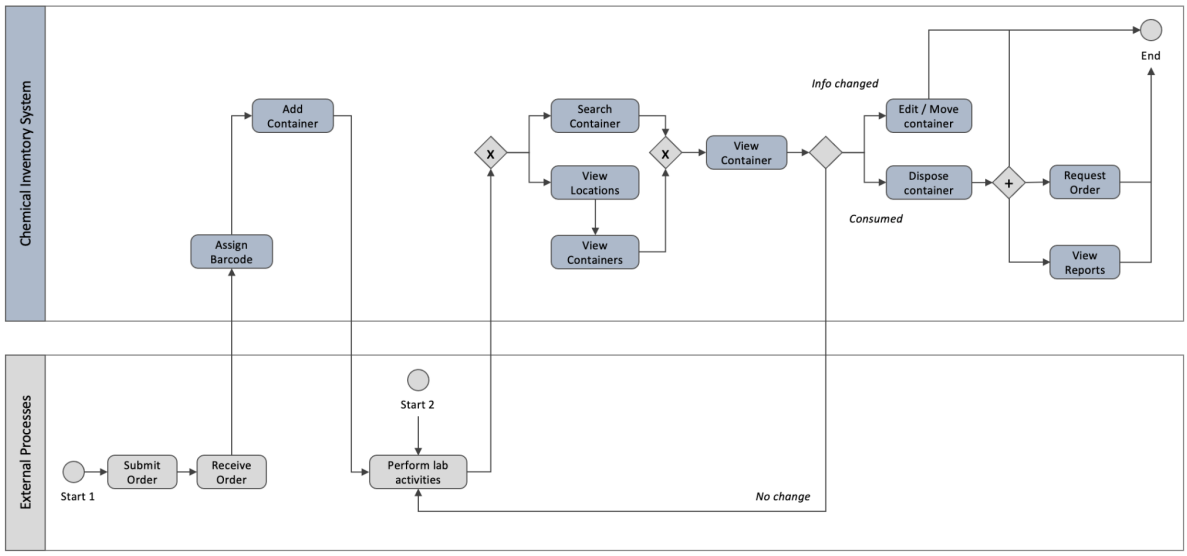
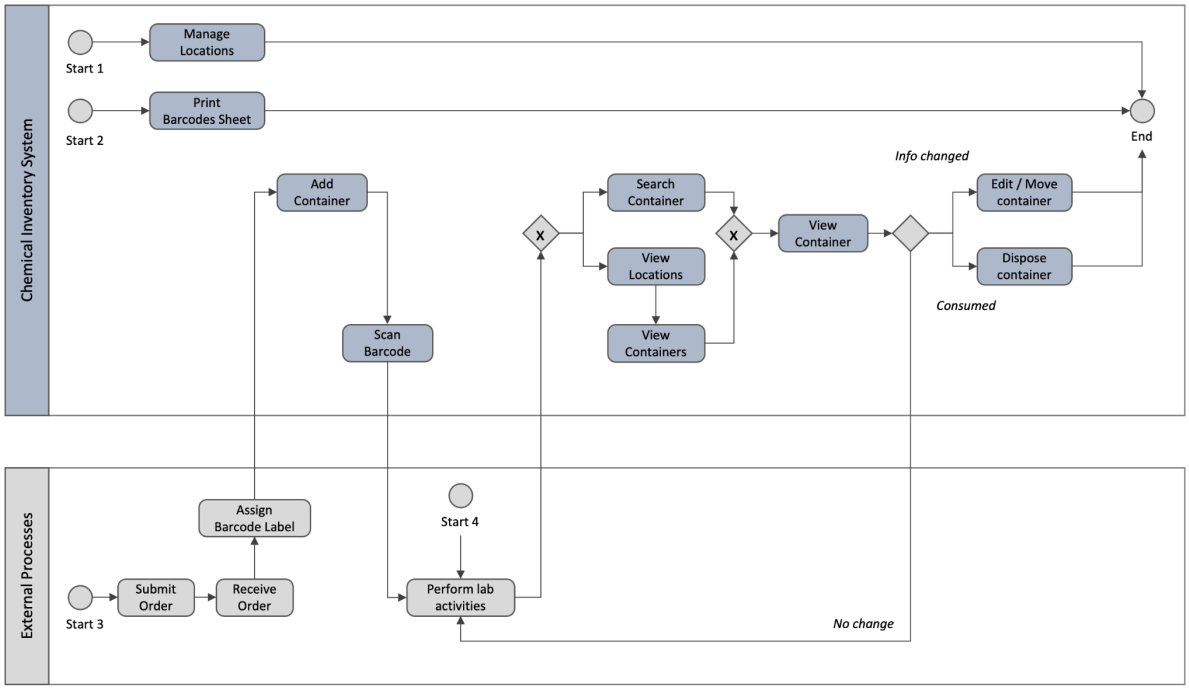


Figure 3.1 - Final Business Process Flow



C. System Architecture

Logical Architecture

The diagrams below illustrate the conceptual connections between the primary users and major system components. The inventory system has been built as a standard web-based application on a distributed architecture with different tiers for presentation, business, and data respectively. This promoted separation of concerns, flexibility, modularity, and potential scalability.

Figure 3.2 - Proposed Logical Architecture

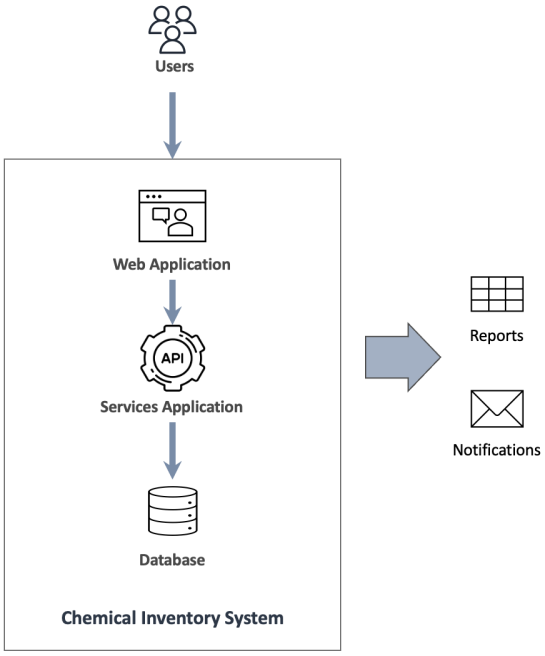
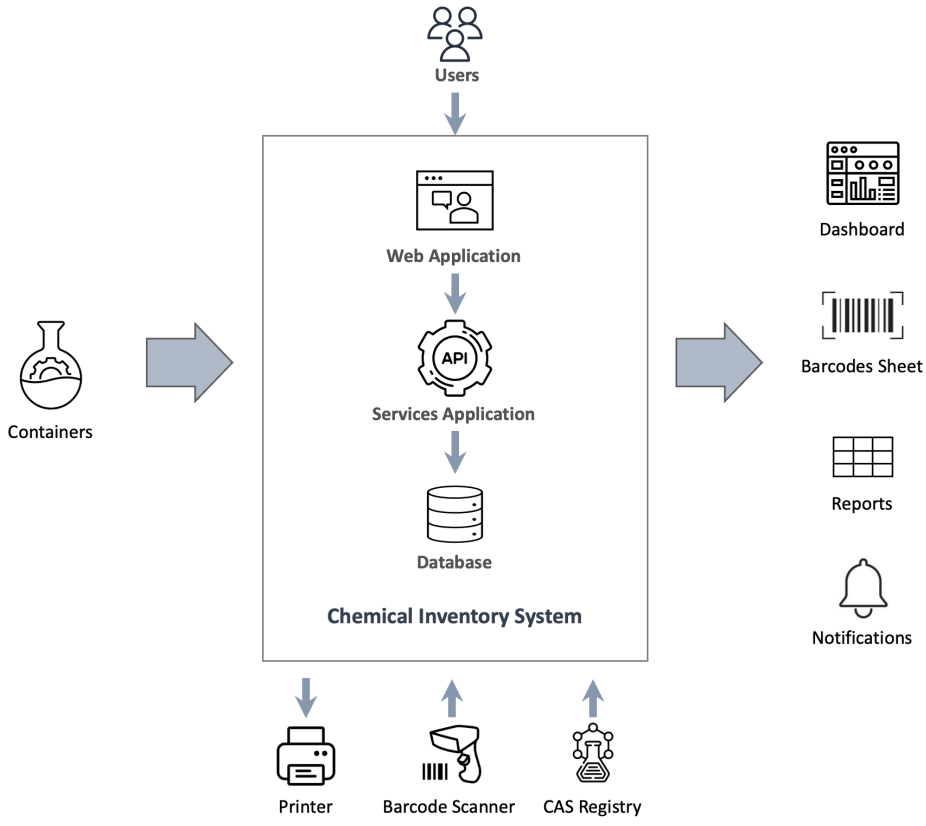
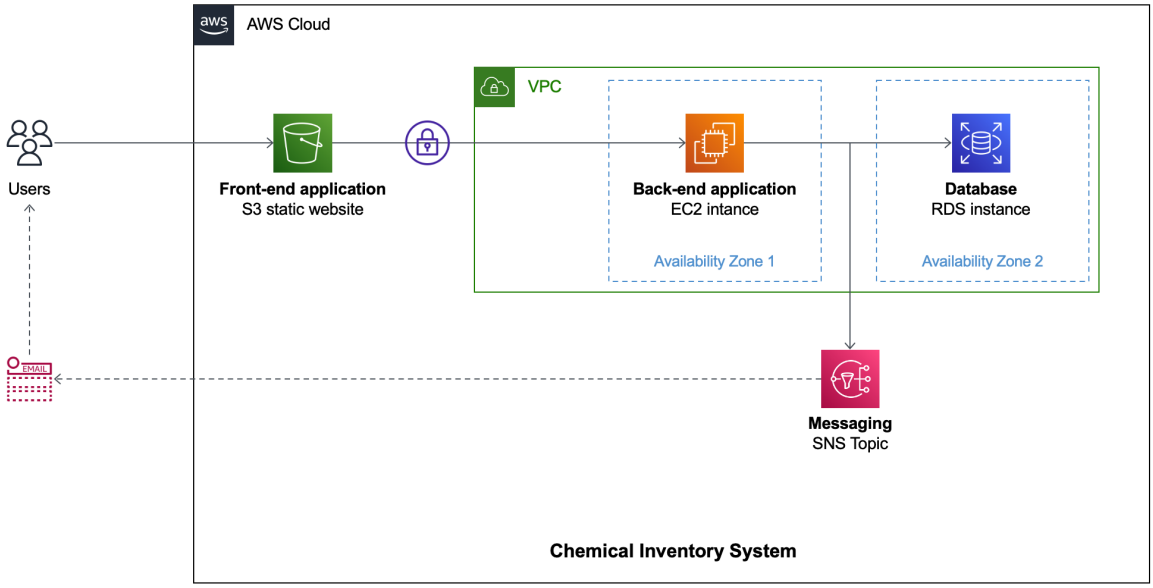


Figure 3.3 - Final Logical Architecture



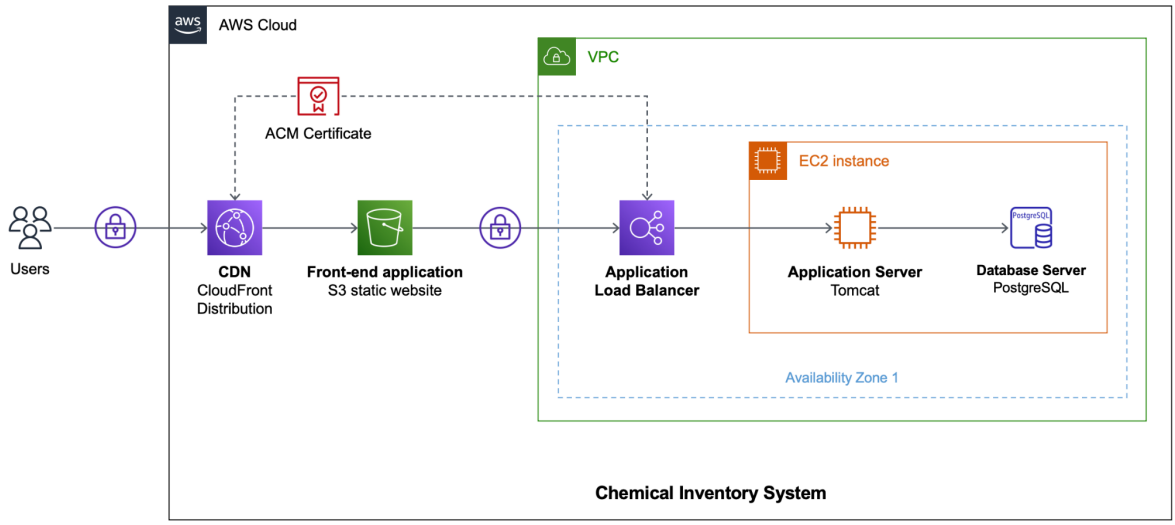
Network / Deployment Architecture

Figure 3.4 - Proposed Deployment Architecture



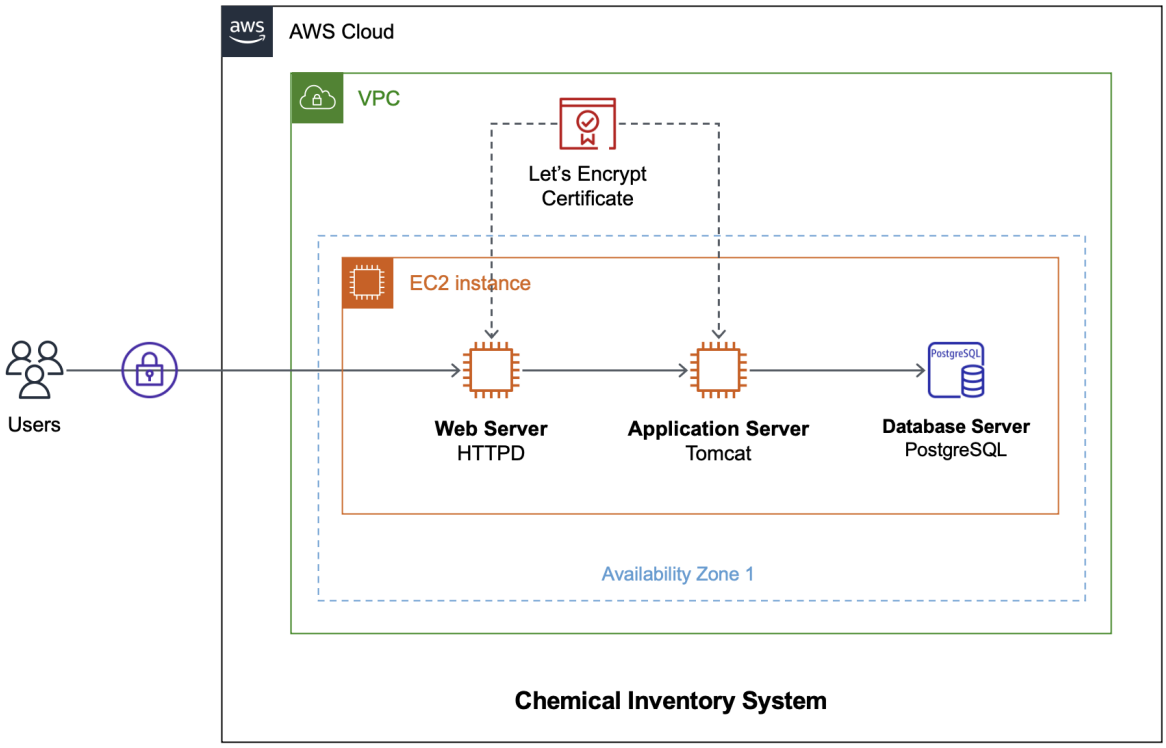
The diagram above illustrates the originally proposed network deployment of the system components on AWS cloud infrastructure using VPC, S3, EC2, RDS, and SNS. In this setup, a web server is no longer required for the front-end application since it can be hosted in S3 instead. The back-end services shall be hosted in an application server on an EC2 server instance while the database is also hosted independently on an RDS instance. Servers will be secured using proper configurations for a private and secured network on the cloud. This is the most ideal setup that promotes reliability and scalability since each respective server can be provisioned independently and in accordance with the resource demands of each layer component.

Figure 3.5 - POC Deployment Architecture



The diagram above illustrates the initial system network deployment during the early stages of development as a proof of concept for utilizing AWS service components. Certain architectural changes have been made to account for higher cost projections relative to the actual system size, resource needs, and necessity. For example, RDS has been replaced with an internal EC2 database while SNS has been removed in favor of standard web notifications instead. In addition, AWS ACM, Cloudfront, and ELB were also utilized to support scalability, performance, and security (HTTPS).

Figure 3.6 - Final Deployment Architecture



The diagram above illustrates the final network deployment towards the later stages of development to account for minimal cost projections over the long run. This utilized a single EC2 server hosting the web server, application server, and database server, which can be scaled down during off-hours. This also required the use of Let's Encrypt for SSL/TLS encryption to secure web traffic into the static web files as well as into dynamic API endpoints.

D. Technology Overview

The project has been developed by leveraging various frameworks and open-source technologies for application development. These frameworks are widely adopted in the industry, have a thriving developer community, have extensive resource materials, and generally simplify and speed up coding and development.

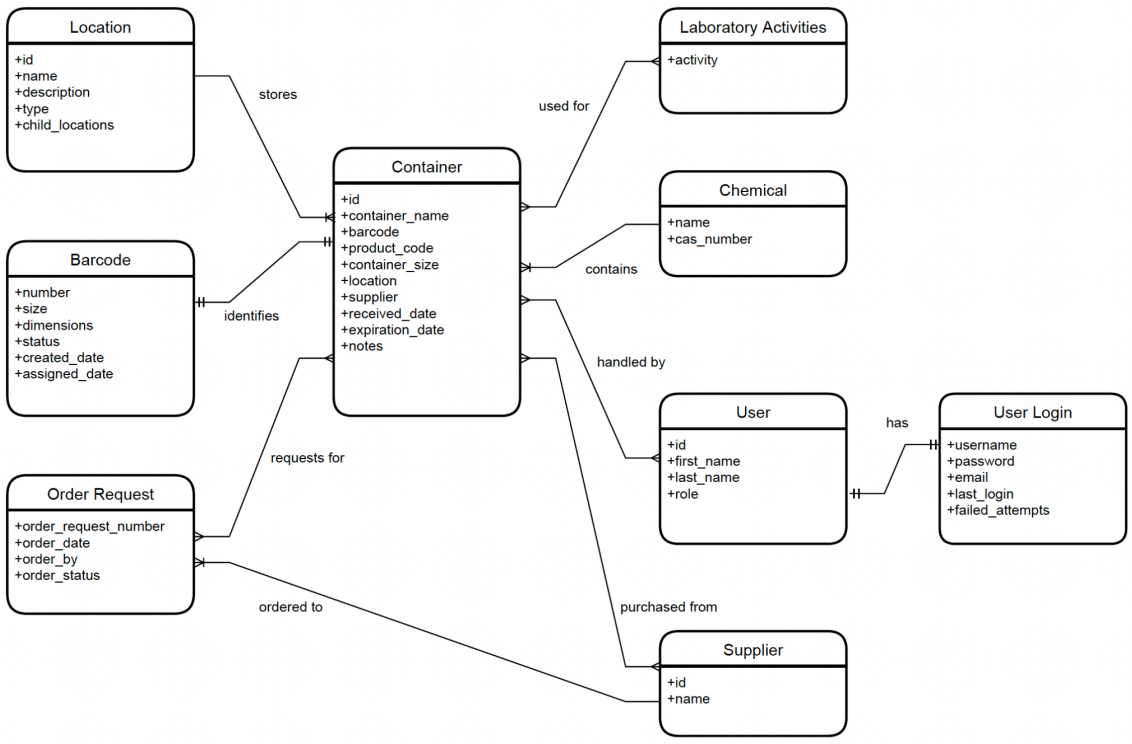
The front-end utilized a web application framework such as Angular for developing a Single Page Application (SPA) using Typescript/Javascript, alongside a Bootstrap web template. The back-end utilized Java and Spring Boot to create RESTful web service API endpoints serving JSON content to the front-end. It also utilized an application server, specifically Tomcat. The database utilized a relational database, specifically PostgreSQL, as well as Flyway for data schema setup/migration. The infrastructure has been built on the AWS cloud for hosting the web, application, and database components. These included AWS services such as VPC, Route 53, and EC2 among others. 3rd party libraries have also been utilized for technical functions such as barcodes, reports, notifications, API security, etc. Physical devices such as printer and barcode scanner have also been used when handling barcode sheets and labels. See Appendix for detailed technical listings.

E. Data model

Entity Relationship Diagram

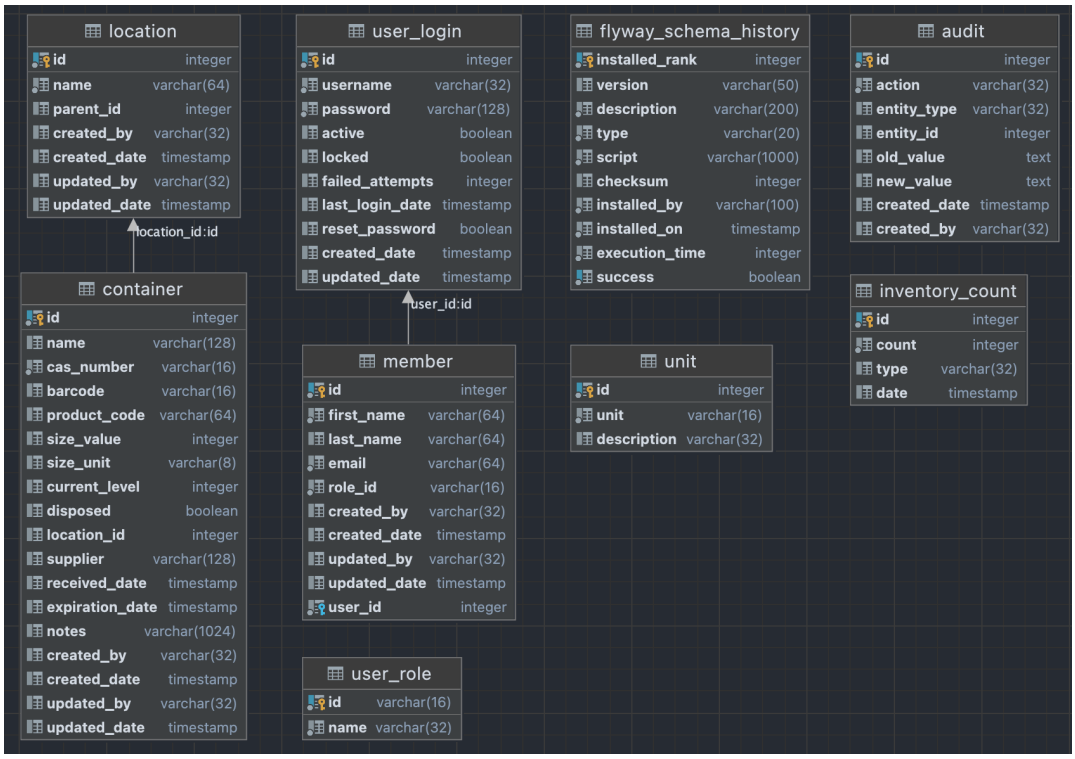
The diagrams below illustrate the proposed logical model and the implemented physical schema for the Chemical inventory system.

Figure 3.7 - Proposed ERD



Database Schema

Figure 3.8 Final database schema



The physical database schema has been simplified into flat tables for direct and straightforward data access and storage. Flyway has also been utilized as a database utility tool for easily managing initial schema setup and migration.

F. Security

The following security mechanisms have been implemented:

- SSL/TLS server configuration using Let's Encrypt certificates for secured HTTPS access on entry points:

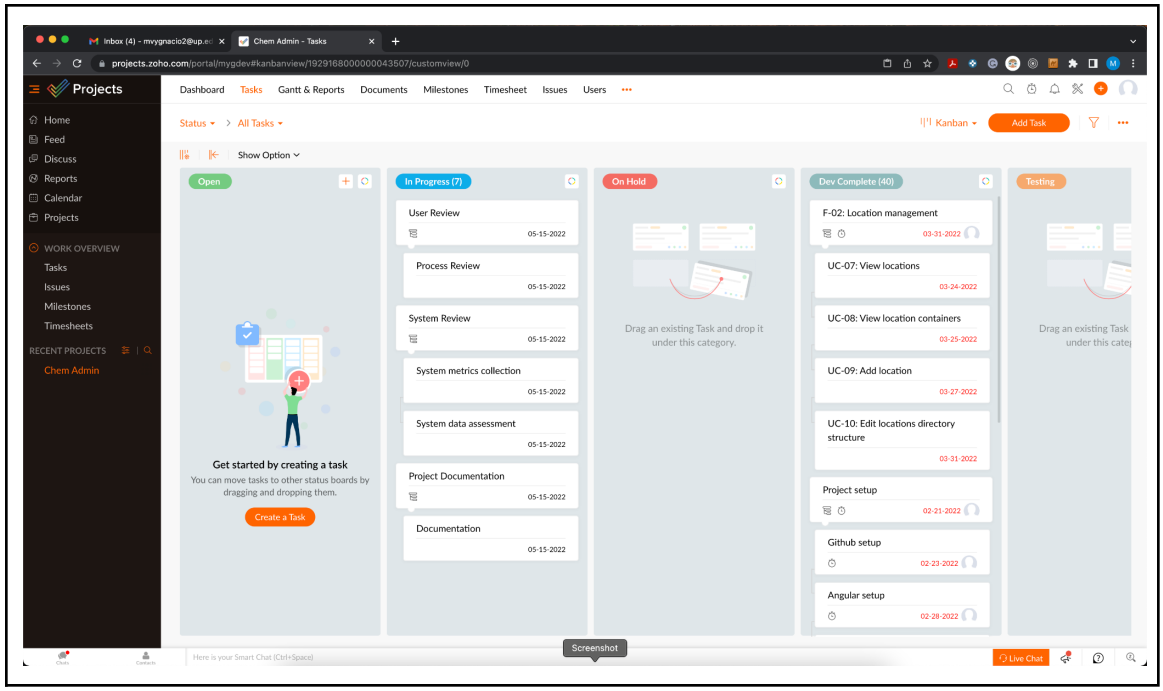
    - Port 443 on web server for static content
    - Port 8443 on application server for dynamic content
- User login
    - Secure password storage / encryption using BCrypt
    - Time-limited user login sessions using JWT
    - Account locks for maxed failed login attempts
    - Password reset
- API Security
    - Spring Security to configure authentication and authorization via user roles and permissions
    - JSON Web Tokens (JWT) to grant temporary access credentials with a pre-defined expiration of 1 hour
    - Cross-Origin Resource Sharing (CORS) configuration to only allow API access from pre-defined hostnames, e.g. domain name of front-end application
- Network
    - AWS security groups for controlled access to servers on various ports (443, 8443, 22) from known/allowed sources only

G. Project Framework

Considering the project resources and schedule, Kanban has been utilized as the project methodology given that it is a popular and widely used framework

for implementing Agile and DevOps development of software solutions. Using Zoho Projects as the Kanban software, this approach required real-time communication of capacity as well as full transparency of work between the developer/team and client. This has been achieved through the use of a Kanban board that visually represents work items and their corresponding status, such as ready for work, to do, in progress, code review, merge, in testing, and approved for deployment, to name a few. This allowed the proponent and stakeholders to see the state of every piece of work at any given time, as well as limit work-in-progress and maximize work efficiency through the project lifecycle.

Figure 3.9 - Zoho Projects Kanban Board (Towards the end of the project)



H. Coding Best Practices

The following system design principles and coding best practices have been utilized during coding and development:

- Utilization of version control tool, such as Github.com for storing applications source code, as well as doing continuous multiple commits.
- Object-Oriented Analysis and Design (OOAD) by modeling system objects' structure, behavior, and interactions after real-world logical concepts and references. Example objects include container, location, barcode, and audit, to name a few.
- Single Responsibility Principle (SRP) by designing each class to have a single purpose where it mostly does one thing and it does it well. Examples include controllers, services, repositories, data access objects, data transfer objects, etc.
- Dependency Inversion Principle (DIP) by prompting loose coupling by having higher-level classes depend on abstractions rather than concrete implementations of lower-level classes. Examples include injecting respositories into services via constructor parameters.
- Unit testing through Behavior-Driven Development (BDD) convention by using simple natural-language constructs that express the intended behavior and the expected outcomes of system components. Examples include writing test cases and using BDD testing frameworks such that

testing syntax closely resembles the semantics akin to reading code in plain English.

- Functional Programming (FP) by using a declarative programming paradigm of applying and composing functions to emulate system behavior. Examples include pure functions that depend solely only on input parameters and return a value without producing any side effects. This idempotent nature of functions leads to code simplicity and predictability. Functional programming has been used alongside OOP (Object-Oriented Programming) despite the 2 being different programming paradigms where objects normally represent structure while functions represent some reusable, specific, or focused behavior.
- Separation of concerns by segregating the system into distinct components or sections based on their intended roles or responsibilities. Examples include breaking classes into layers such as the web, business, and data access layers.

IV. Project Assessment

A. Kanban Framework

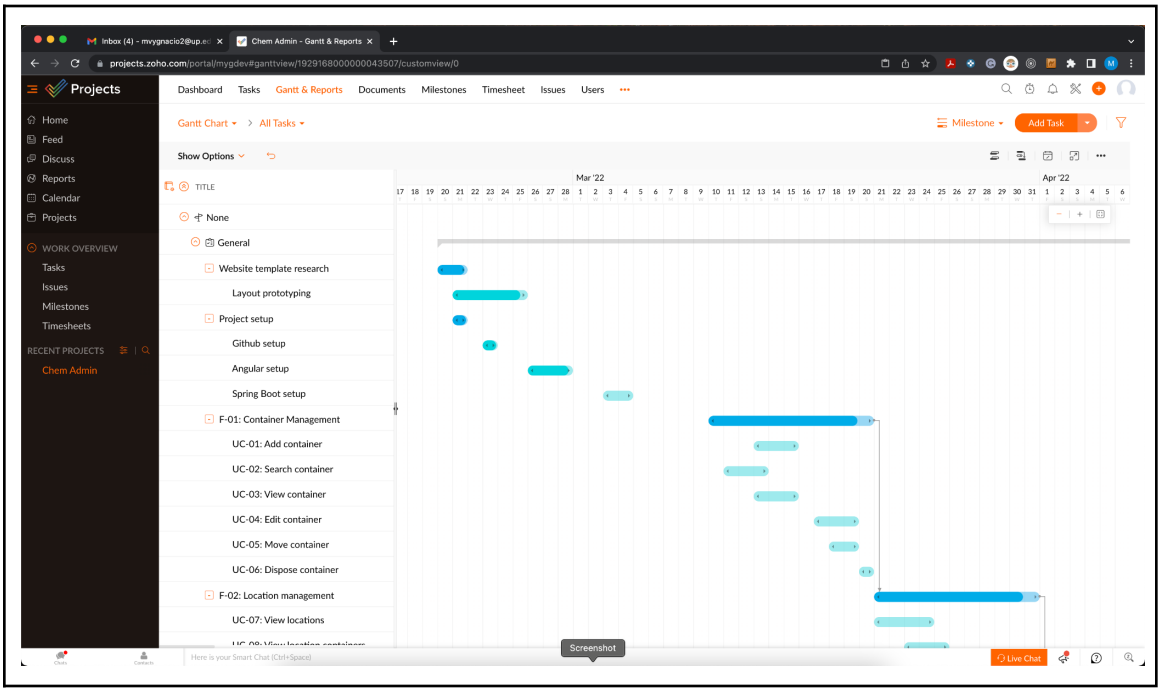
Compared to alternative frameworks such as Waterfall and Agile Scrum methodologies, Kanban, through Zoho Projects software, has been an ideal framework for this project for the following reasons:

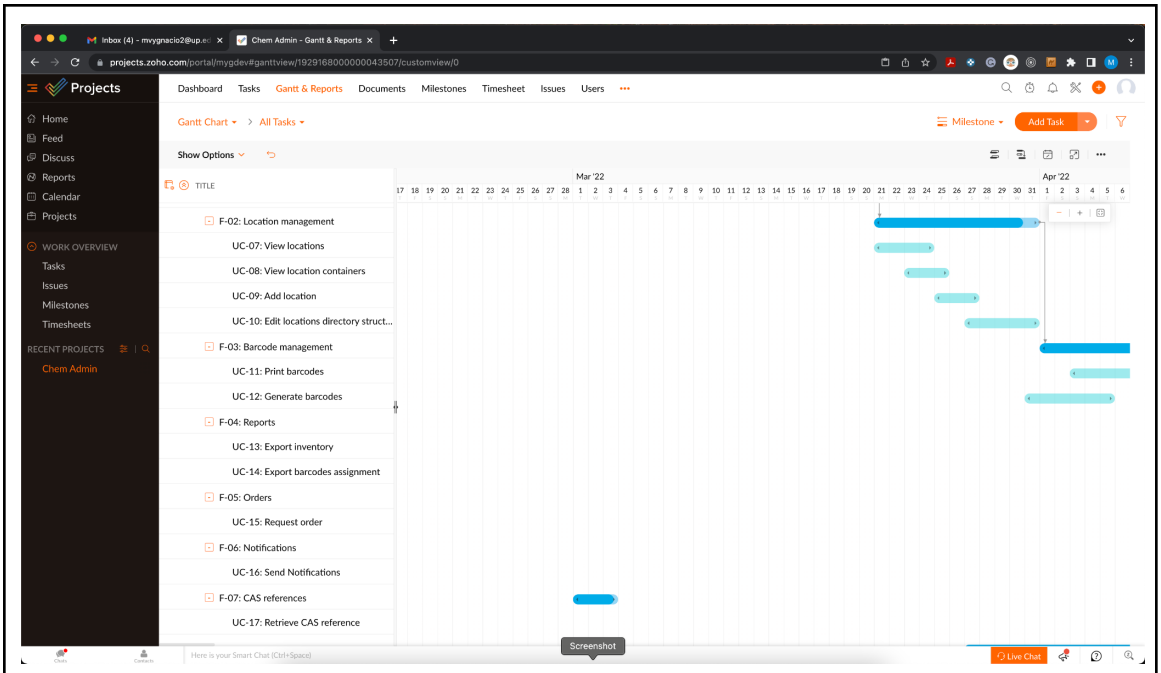
- It provided flexibility for development since it has no prescribed stage durations between bodies of work, features, and tasks.
- It provided visualization of development progress towards project completion through the use of the Kanban board and Gannt chart.
- It allowed for increased productivity and efficiency by allowing to focus on high-value and high-impact items, such as core and special features and infrastructure.
- It focused on continuous delivery where small working increments of the system were continually released to the client. For example, the important use cases and features, such as adding and viewing containers, have been worked on and deployed first followed by others in container management, location management, barcodes, etc.
- It allowed requirements and priorities to be continuously reevaluated based on the latest information and state. This allowed the system to be delivered exactly as the client wants and needs, concerning current

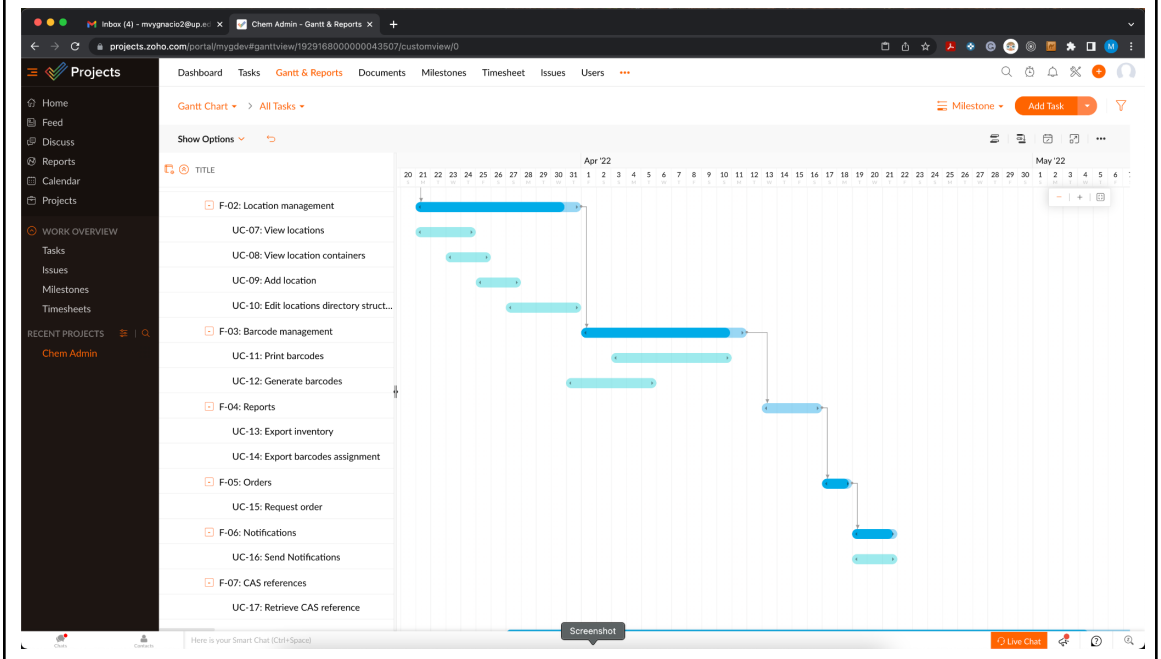
problems with manual inventory processes as well as pain points of prior inventory systems experience.

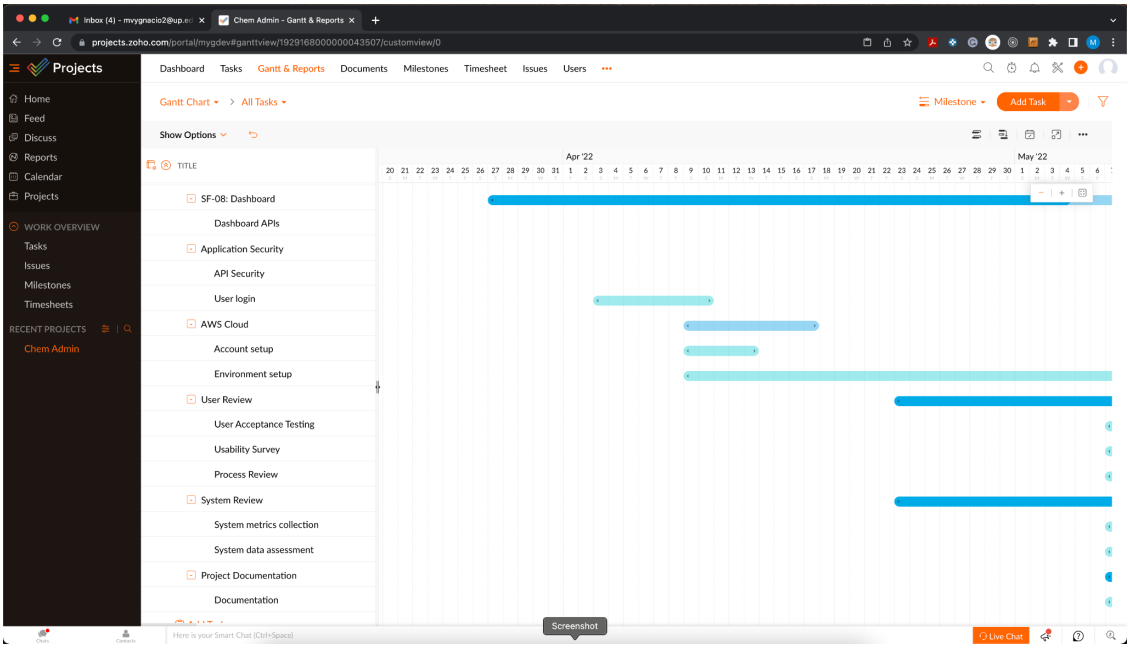
Having been able to realize these above-mentioned benefits has been an absolute necessity for a custom-built application that was constrained by a limited budget, a relatively short and tight schedule, and a single developer resource. Also, having had prior development experience as a professional Software Engineer has contributed tremendously to covering the significant coding effort required for the given system features in a relatively short timeframe.

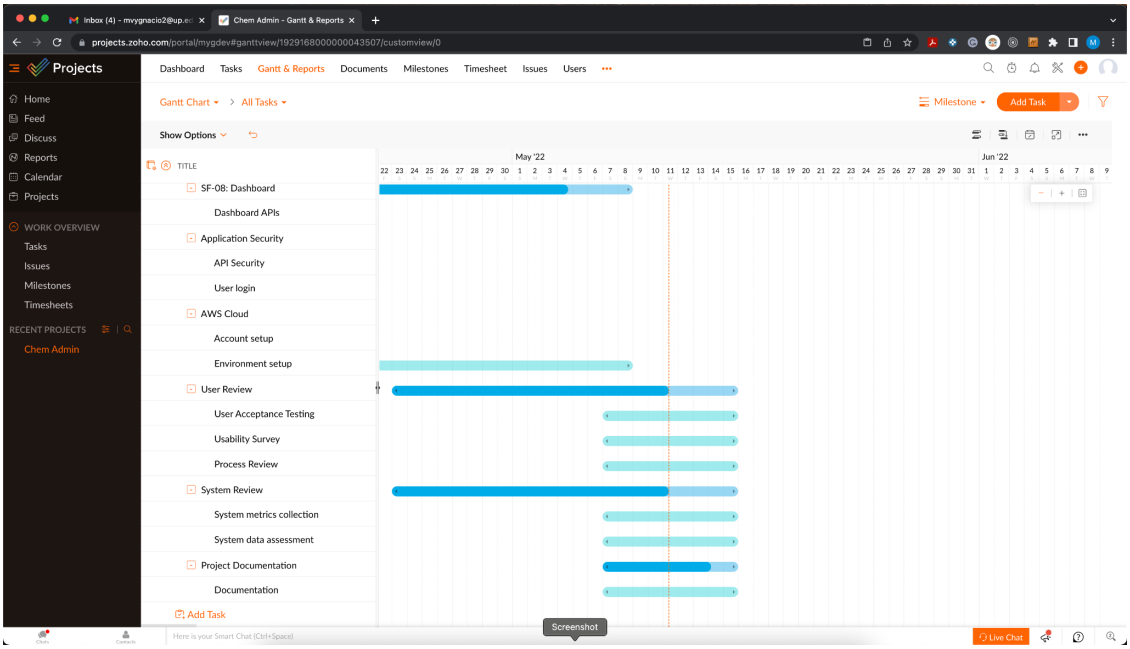
Figure 4.0 - Zoho Projects Gantt Chart




Projects
Dashboard
Tasks
Gantt & Reports
Documents
Milestones
Timesheet
Issues
Users
Gantt Chart
All Tasks
Milestone
Add Task
Show Options
TITLE
Mar '22
Apr '22
F-02: Location management
UC-07: View locations
UC-08: View location containers
UC-09: Add location
UC-10: Edit locations directory struct...
F-03: Barcode management
UC-11: Print barcodes
UC-12: Generate barcodes
F-04: Reports
UC-13: Export inventory
UC-14: Export barcodes assignment
F-05: Orders
UC-15: Request order
F-06: Notifications
UC-16: Send Notifications
F-07: CAS references
UC-17: Retrieve CAS reference
Home
Feed
Discuss
Reports
Calendar
Projects
WORK OVERVIEW
Tasks
Issues
Milestones
Timesheets
RECENT PROJECTS
Chem Admin
Screenshot
Live Chat


Projects
Dashboard
Tasks
Gantt & Reports
Documents
Milestones
Timesheet
Issues
Users
Gantt Chart
All Tasks
Milestone
Add Task
Show Options
TITLE
Apr '22
May '22
F-02: Location management
UC-07: View locations
UC-08: View location containers
UC-09: Add location
UC-10: Edit locations directory struct...
F-03: Barcode management
UC-11: Print barcodes
UC-12: Generate barcodes
F-04: Reports
UC-13: Export inventory
UC-14: Export barcodes assignment
F-05: Orders
UC-15: Request order
F-06: Notifications
UC-16: Send Notifications
F-07: CAS references
UC-17: Retrieve CAS reference
Home
Feed
Discuss
Reports
Calendar
Projects
WORK OVERVIEW
Tasks
Issues
Milestones
Timesheets
RECENT PROJECTS
Chem Admin
Screenshot
Live Chat


Inbox (4) - mvygnacio2@up.edu
Chem Admin - Gantt & Reports
projects.zoho.com/portal/mygdev#ganttview/192916800000043507/customview/0
Projects
Dashboard
Tasks
Gantt & Reports
Documents
Milestones
Timesheet
Issues
Users
Home
Feed
Discuss
Reports
Calendar
Projects
WORK OVERVIEW
Tasks
Issues
Milestones
Timesheets
RECENT PROJECTS
Chem Admin
Gantt Chart
All Tasks
Milestone
Add Task
Show Options
TITLE
Apr '22
May '22
SF-08: Dashboard
Dashboard APIs
Application Security
API Security
User login
AWS Cloud
Account setup
Environment setup
User Review
User Acceptance Testing
Usability Survey
Process Review
System Review
System metrics collection
System data assessment
Project Documentation
Documentation
Screenshot
Live Chat


Inbox (4) - mvygnacio2@up.edu
Chem Admin - Gantt & Reports
projects.zoho.com/portal/mygdev#ganttview/192916800000043507/customview/0
Projects
Dashboard
Tasks
Gantt & Reports
Documents
Milestones
Timesheet
Issues
Users
Home
Feed
Discuss
Reports
Calendar
Projects
WORK OVERVIEW
Tasks
Issues
Milestones
Timesheets
RECENT PROJECTS
Chem Admin
Gantt Chart
All Tasks
Milestone
Add Task
Show Options
TITLE
May '22
Jun '22
SF-08: Dashboard
Dashboard APIs
Application Security
API Security
User login
AWS Cloud
Account setup
Environment setup
User Review
User Acceptance Testing
Usability Survey
Process Review
System Review
System metrics collection
System data assessment
Project Documentation
Documentation
Add Task
Screenshot
Live Chat

B. Development Methods

The following table summary lists the proposed or planned development methods or activities and their corresponding implementations during the course of the project:

Planned	**Implemented**
Review of current manual documents	Yes. Client has provided details of manual documents format and data fields; these were used to define data model.
Review of alternative licensed systems	Yes. Reviewed technical system features of the following systems: ● Chematix ● ChemInventory ● Chemical Safety EMS
Review of available website design templates	Yes. Searched, reviewed, and selected applicable website bootstrap templates for an inventory system in Template Monster and Theme Forest sites.
Template customization for system UI/UX design	Yes. Explored, conceptualized, and customized available UI/UX components in the following frameworks: ● Angular Material components ● Porto admin template components ● JQuery plugins/extensions
Template integration with the web application framework	Yes. Integrated the following UI frameworks to work together:

	● Angular/Typescript for application structure/logic flow (components, services, etc) ● Available components from Angular material (cards, datepicker, snackbar ,etc) ● HTML, Bootstrap, CSS usage from Porto admin template
UI design consultations	Yes. Consulted with client on usage preferences, such as for: ● Amount percentage slider ● Dashboard widgets and links ● UI navigation flows ● Barcode scanning support ● Container form validations
Data modeling and database design	Yes. Utilized UML modeling tools (draw.io) and built-in IntelliJ database schema visualizer.
RESTful web service API endpoints design	Yes. Utilized Spring Boot to create secured REST controllers based on HTTP verbs and proper naming conventions.
Review of frameworks and libraries, e.g. API security, barcode generation	Yes. Utilized 3rd party frameworks for given technical usages (See Appendix).
Review of public datasets, e.g. CAS references	Yes. Requested and received access to official 3rd party CAS public API endpoints.
Local development environment setup	Yes. Utilized work development machine for this project as well.
Iterative development using Kanban	Yes. Reviewed available Kanban software alternatives, then selected and utilized Zoho Projects Software;

	used Kanban board and Gantt chart.
Prototype development	Yes. Developed proof of concepts for the following: ● Baseline application with separate front-end and back-end applications on AWS infrastructure ● Multiple network deployment setups to test/utilize various AWS security components
Feature demos with users	Yes. Performed feature demos and user testing with client upon reaching major development milestones.
Cloud environment setup	Yes. Utilized Amazon Web Services (AWS) cloud infrastructure.
Review of cloud setup costs and free tier limits	Yes. Created/reviewed cost calculations/projections; Reviewed actual monthly billing.
Preparation of final project documentation/report	Yes. Completed final project documentation/manuscript.

C. Testing Methods

The following table summary lists the proposed or planned testing or assessment methods or activities and their corresponding implementations during the course of the project:

Planned	Implemented
Provision of user access to the Test environment for the ability to view/test the system at any given time	Yes. User credentials created and provided to client and test users.
Iterative development with continuous deployments for frequent releases to users for testing and feedback	Yes. Continued deployment of UI and API apps to AWS environment from middle towards end of project duration to release features or fix any bugs/issues.
Utilization of usability surveys and feedback sessions	Yes. Conducted usability survey with 5 confirmed responses and 2 pending.
Process evaluation before, during, and after the system release to determine changes in process workflows	Yes. Client now utilizes the system as the centerpiece of inventory management workflow without the need for prior manual documents.
Assessment of previous manual documents to determine transition into the new system	Yes. Client has provided details of manual documents format and data fields.
User acceptance testing (UAT) with final client approval to deploy/promote for use in production	Yes. Client has completed User acceptance testing alongside Usability survey towards the end of the project. Client has also agreed and approved for actual use starting end of May using the summer schedule to populate existing inventory. This will be in preparation for the upcoming semester.
Collection of available system metrics to determine usage patterns and resource utilization via monitoring tools	Yes. Utilized linux commands and AWS monitoring to view/monitor resource utilization.
Introspection of system-generated data and evaluation of data patterns	Yes. Utilized IntelliJ database client to view/access preliminary system data.

via database queries	
Client discussions sessions to determine concerns on system perception, usability, and effectiveness	Yes. Received client feedback during deployments/demos and made corresponding improvements/fixes.
Evaluation of interest for adoption by other professors in the department	Yes. Another Chemistry Professor has expressed keen interest in using the system this upcoming semester, after participating in the demo/usability survey.

D. Usability Survey

Respondent Profile

There were 5 responses received from a total of 6 invited participants. These consist of working professionals in Chemistry and Information Technology industries. The respondents have varying levels of technical proficiency but are generally familiar with an inventory system. They also hold the following roles/positions:

- Chemistry Professors
- Chemistry PhD graduates
- Information Technology Manager
- Software Quality Assurance (QA) Engineer

SUS Scores per Respondent

Participant	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	SUS Score
p1	3	1	5	1	3	3	5	1	5	1	**85.0**
p2	4	1	5	1	5	1	4	1	5	1	**95.0**
p3	5	1	5	1	4	1	5	1	5	1	**97.5**
p4	4	4	3	2	4	2	4	2	4	2	**67.5**
p5	4	2	4	1	4	2	5	1	4	2	**82.5**

Mean SUS Score and Interpretation

The average SUS score from the above responses is **85.5**. According to *usability.gov*, this can be interpreted as being a usable system given that a SUS score above a 68 would be considered above average.

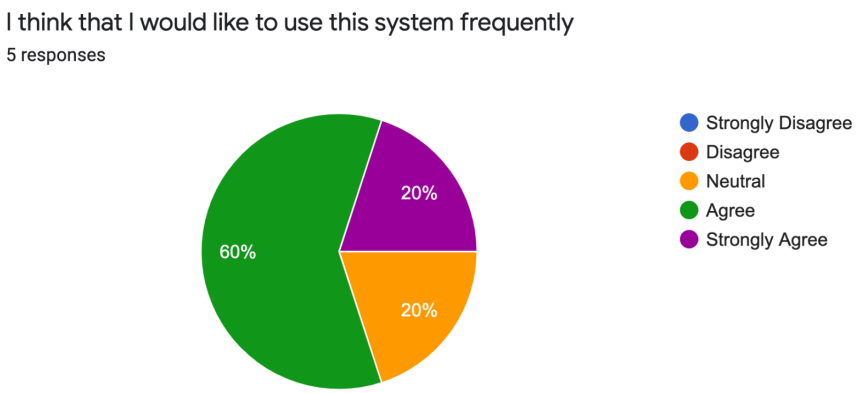
The respondents were also only given the website link and a short description of what it is in general. They were not provided with instructions as to how to specifically use the system which in itself serves as a test for the system's usability. The website UI/UX design (theme, layout, navigation, and overall look-and-feel) also served as the primary driver for the system's overall usage patterns or usability. As such, it is also noteworthy to mention that users with a Chemistry background have had no difficulties interacting with the system whereas non-Chemistry respondents (e.g. QA respondent) had more questions and concerns due to the lack of domain context. The client also ended up as

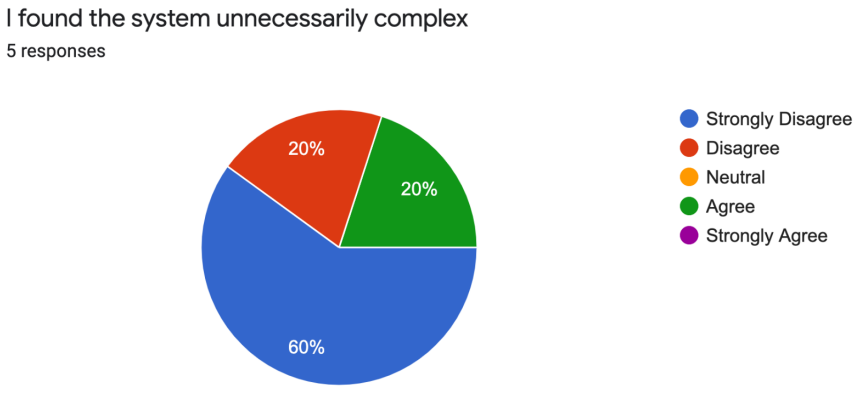
having the highest SUS score due to the system being tailored specifically for their requests and needs.

Individual Responses

Figure 4.1 - Usability responses

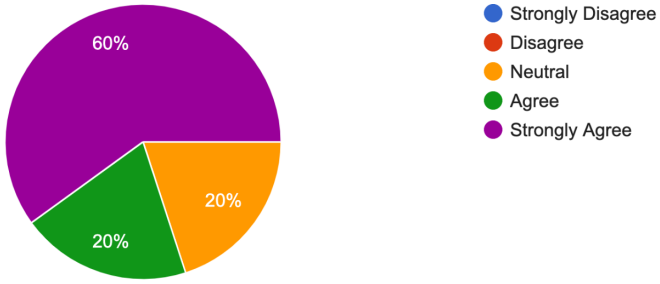
	A	B	C	D	E	F
1	Timestamp	2022/05/04 1:38:39 PM AST	2022/05/05 9:54:52 PM AST	2022/05/08 2:27:16 PM AST	2022/05/09 12:50:43 PM AST	2022/05/15 8:04:13 PM AST
2	I think that I would like to use this system frequently	Neutral	Agree	Strongly Agree	Agree	Agree
3	I found the system unnecessarily complex	Strongly Disagree	Strongly Disagree	Strongly Disagree	Agree	Disagree
4	I thought the system was easy to use	Strongly Agree	Strongly Agree	Strongly Agree	Neutral	Agree
5	I think that I would need the support of a technical person to be able to use this system	Strongly Disagree	Strongly Disagree	Strongly Disagree	Disagree	Strongly Disagree
6	I found the various functions in this system were well integrated	Neutral	Strongly Agree	Agree	Agree	Agree
7	I thought there was too much inconsistency in this system	Neutral	Strongly Disagree	Strongly Disagree	Disagree	Disagree
8	I would imagine that most people would learn to use this system very quickly	Strongly Agree	Agree	Strongly Agree	Agree	Strongly Agree
9	I found the system very cumbersome to use	Strongly Disagree	Strongly Disagree	Strongly Disagree	Disagree	Strongly Disagree
10	I felt very confident using the system	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Agree
11	I needed to learn a lot of things before I could get going with this system	Strongly Disagree	Strongly Disagree	Strongly Disagree	Disagree	Disagree





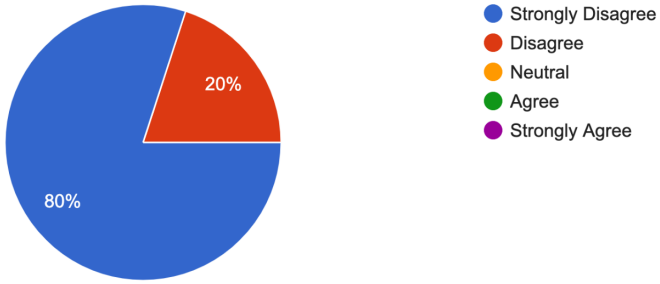
I thought the system was easy to use

5 responses



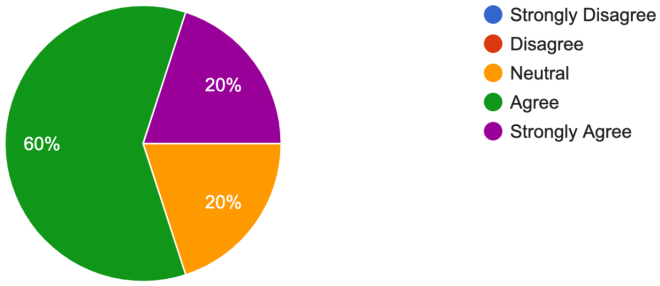
I think that I would need the support of a technical person to be able to use this system

5 responses



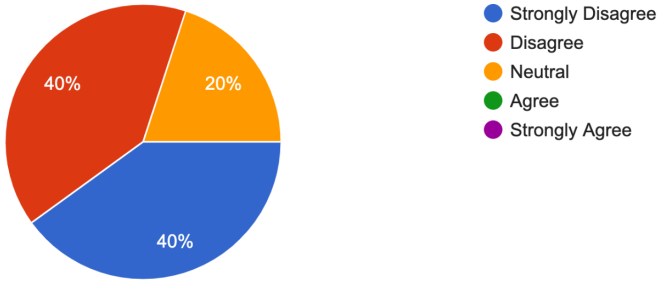
I found the various functions in this system were well integrated

5 responses



I thought there was too much inconsistency in this system

5 responses

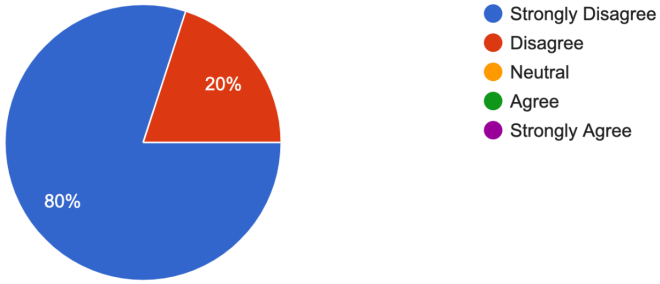


I would imagine that most people would learn to use this system very quickly

5 responses

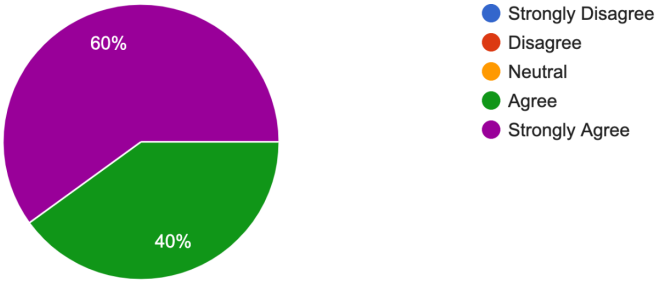
I found the system very cumbersome to use

5 responses



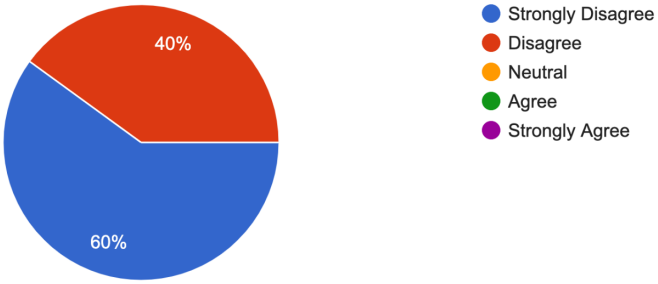
I felt very confident using the system

5 responses



I needed to learn a lot of things before I could get going with this system

5 responses



Comments (Optional)

2 responses

For users that are not used to dark mode maybe a light mode theme could be an option?

I found the system very easy to use with seamless user interface. I have used several systems similar to this, and I must say that this is by far better than the ones I used. I am excited to see this system evolve and offer more advanced functionalities such as automatic addition of containers based on photos. Conceivably, this system could also expand beyond chemical inventories and be integrated to online laboratory notebook management.

Job Title (Optional)

3 responses

IT Manager

Assistant Professor of Chemistry

Software test analyst

V. Discussions

Software Adoption

The use of software to manage chemical inventory in the client's institution has not been given much focus over many years. This is mainly due to the broad coverage of a liberal arts college in general and having limited budget and prioritization within the department towards software adoption. Second, is due to the less tech-savvy nature of more senior professors who have been content working with physical documents and manual processes. However, this perspective is limited to only within the Chemistry department as this project had no visibility of any current or planned software solutions in other departments. Given the project output, user testing, and usability results, the application has received positive feedback and perception from users. The client, alongside another Chemistry professor, has confirmed the adoption of this system starting with summer classes and then for the next semester. They have also expressed interest in making further improvements for succeeding releases. This supports the idea that software adoption increases when the system is highly functional by supporting core features/processes and has a fairly simple, intuitive, and user-friendly system interface which ultimately helps the users' tasks easier, faster, and more efficient.

Solution Alternatives

The development of a custom-built Chemical Inventory System over commercial alternatives has allowed for delivering a cost-effective solution that closely matches the client's wants and needs. The project has focused and placed more emphasis on supporting the core, high-value features and use cases, alongside having a modern and intuitive user interface. This has also allowed the flexibility to make appropriate design decisions and customizations that address specific pain points and any commercial limitations. This system included special features such as notifications, dashboards, barcode integration, and CAS registry lookups. However, it should be noted that this system, on its first release, will not be as mature and diverse overall as other software alternatives given that it is still in the early phase of its product lifecycle. Future development work for additional feature requests and technical improvements shall help improve the system into a more advanced state.

Project Methodology

The use of the Kanban framework during system development has made a significant contribution to completing the project on time given limited development resources and a relatively tight schedule. The visual, flexible, and iterative nature of Kanban has helped make steady and continuous progress

while reviewing feedback, reassessing priorities, and scheduling remaining work towards final completion. However, it should be noted that the Zoho Projects Kanban software required a minimal monthly subscription which turned out to be a reasonable expense given the benefits that were realized. Also, Zoho Projects has some implementation differences over other used Kanban software like Atlassian Jira but the general concepts and flows are mostly similar. This project, along with my professional work projects, has also shown why Kanban is a popular and widely used framework in the software industry given that it works on various size teams, software projects, and business industries.

Software and Technology Components

From an application perspective, various web and application frameworks allowed for rapid development of the front-end (Angular / Typescript) and back-end (Java 17 / Spring Boot) system components. These open-source 3rd party frameworks and libraries have provided baseline structures, technical capabilities, and utility functions upon which domain-specific code has been implemented. Specifically, these have provided valuable dependencies for supporting functional and non-functional aspects such as templates, dashboards, barcodes, reports, project structure, APIs, and security, among others. Without these frameworks, the project would have required more time, resources, and unnecessary coding.

From an infrastructure perspective, utilizing virtual servers on AWS cloud infrastructure has been the clear choice due to the lack of physical on-premise hardware. The cloud-based approach has also allowed for having quick resource access, adapting resource specifications, reducing cost, and leveraging flexibility for experimentations on server configurations and application deployments. For any small to medium organization, leveraging the cloud is highly beneficial due to the reduced barrier of entry, lower overall costs, and the broad availability of cloud-provider services for building and running applications using virtual resources. This project is however limited only to Amazon Web Services (AWS) but other cloud providers such as Microsoft Azure and Google Cloud Platform should be able to provide similar capabilities and benefits.

Cost Concerns

This project has also shown how smaller, non-commercial projects are more susceptible to budget and cost concerns compared to larger businesses and enterprises. This has led to further exploration of AWS free tier limits and resource pricing schemes and then finding the most cost-effective approach for the minimum viable setup with the lowest possible cost. In turn, this required making architectural decisions and tradeoffs to balance functional and non-functional requirements with the corresponding underlying cost implications. As such, this project culminated with a simple and minimal network configuration using a single EC2 server hosting the web, application, and database

components. It should be noted though that this setup is sufficient to handle the expected usage upon the first release. However, once a higher budget is feasible, it is highly recommended to transition into a more segregated approach that provides more scalability, reliability, and performance.

Chemical Data Integration

This project has also included the capability to lookup scientific CAS registry data by leveraging 3rd party sources. This has been done through API integration with the CAS Common Chemistry public datasets, thus also avoiding the need for separately maintaining our own CAS database which is highly likely to become inaccurate and incomplete. This feature has allowed for providing additional chemical context when managing containers. Apart from CAS data, this project also aimed to integrate chemical Safety Data Sheet (SDS) data to provide even more supplementary container information. Unfortunately, there have been no publicly available API sources since potential providers require a paid subscription. However, once a higher budget is available and as the system matures, it is also recommended to reassess options for SDS data integration.

Personal Learning

Despite having been in the IT industry for more than 14 years, this project has also provided an opportunity for gaining knowledge of technical and implementation details that I haven't personally worked on directly beforehand. These include usage of certain 3rd party libraries (barcodes), UI frameworks integration (Angular, JQuery plugins), usage of JWT without commercial providers (Okta), deploying builds without automation, applying SSL security in stand-alone servers, AWS free tier exploration and cost management, and public API integration for scientific data. This highlights the case that every project provides one or more avenues for learning and that continued growth is a must for IT professionals to thrive in the industry.

VI. Conclusions

In this project, the proponent has completed the development of a Chemical Inventory System for a Chemistry Professor in an academic institution to replace an old, manual, inefficient process with a software-based solution for managing chemical inventory. The system has covered the proposed key features for container and location management, alongside special features for dashboards, notifications, barcode integration, reports and CAS registry integration. The system has also been built and deployed to a working environment, has been positively reviewed/tested by users, and has been accepted for use by the client, with additional feature requests for future consideration.

The project has also shown that a combination of project management methodologies, technical frameworks, and prior work experience can support project completion given limited resources and schedule. These include the use of, first, the Kanban framework for managing development tasks and cadence/progress; second, web frameworks such as Angular/Typescript, web design templates, and Javascript plugins; third, application languages or frameworks such as Java 17, Spring Boot, Spring JPA, Flyway, Gradle, and RESTful API webservices; fourth, PostgreSQL relational database; fifth, virtual resources on Amazon Web Services (AWS) cloud infrastructure. However, this has also shown how cost can be a limiting factor in non-commercial projects and

how architectural changes and trade-offs may be necessary to minimize cost over the long run.

Lastly, the project has also shown that external, publicly available scientific datasets can be leveraged to provide supplementary contextual information within the application. However, this can only be supported up to a certain extent subject to the availability and pricing model of providers, as has been possible for CAS Registry information but not so for Safety Data Sheet (SDS) information.

Given a potential higher budget allocation and increased maturity in the future, it is recommended to make the corresponding architecture changes, technical design improvements, and new feature requests to match any change in functional and technical requirements.

VII. Future Work

A. Feature Requests

The following features have been requested by the client during the User Acceptance Testing period. This shall be considered for future development subject to the schedule and renewal of the client agreement.

- In the Locations module, when viewing a given parent location, then it should display all containers assigned to that location as well as all containers assigned to its child/descendant locations.
- In the Locations module, the user would like to sort locations by custom order instead of alphabetical by default.
- In the Reports module, the user would like to view an audit report that lists all container activity as well as location activity for a more detailed tracking history.
- In the Reports module, the user would like to view container usage patterns such as how fast containers are being consumed and thus how often they need to be disposed of and re-ordered.
- In the Reports module, the user would like to view expired containers for a given date range.
- In the Containers module, the user would like to be able to take a photo of the container label and then the system would auto-detect container

information such as name, CAS number, barcode, size, supplier, expiration date, etc.

- In the Dashboard module, the line graph should also include the current running total instead of just the daily scheduled inventory count process.
- The user would like to have a Management section to manage system users, roles, and password resets.
- The user would also like to view Safety Data Sheet (SDS) information associated with the given containers or chemicals.

B. Technical Recommendations

Given higher budget and resource capacity / allocations in the future, the following technical improvements, configurations, and development work are recommended to address both functional and non-functional concerns:

- Implement auto-scaling configuration to shutdown the EC2 instance during off-peak hours, such as outside regular school hours to minimize on-demand AWS costs.
- Provision of larger EC2 instance types (higher vCPU and memory capacity) to account for any potential increased resource requirements over time.
- Implement pagination to account for eventual larger datasets over time.

- Additional network protection layers, such as AWS WAF and VPC private subnets to further monitor, secure, and filter access to servers and APIs.
- Implement the distributed network architecture setup outlined in the POC to utilize separation of components, scalability, and potential performance improvements.
- Implement additional API permissions for given roles to refine capabilities between ADMIN and USER roles across the system features.
- Implement more unit tests for increased test coverage.

VIII. References

- Angular Framework. (n.d.). *Angular*. Retrieved from https://angular.io/docs
- Angular Material. (n.d.). *Angular*. Retrieved from https://material.angular.io/
- Porto Admin Template. (n.d.). *Themeforest*. Retrieved from https://themeforest.net/item/porto-admin-responsive-html5-template/8539472
- Spring Boot. (n.d.). *Spring*. Retrieved from https://spring.io/projects/spring-boot
- CAS Reference. (n.d.). *CAS American Chemical Society*. Retrieved from https://www.cas.org/cas-data/cas-references
- CAS Common Chemistry API. (n.d.). *CAS American Chemical Society*. Retrieved from https://commonchemistry.cas.org/
- Let's Encrypt. (n.d.). *Let's Encrypt.* Retrieved from https://letsencrypt.org/getting-started/
- AWS. (n.d.). *Amazon Web Services.* Retrieved from https://aws.amazon.com/
- Amazon Corretto 17. (n.d.). *Amazon Web Services.* Retrieved from https://docs.aws.amazon.com/corretto/latest/corretto-17-ug/amazon-linux-install.html
- Apache HTTPD. (n.d.). *The Apache Software Foundation*. Retrieved from https://httpd.apache.org/
- Apache Tomcat. (n.d.). *The Apache Software Foundation*. Retrieved from https://tomcat.apache.org/download-90.cgi
- PostgreSQL. (n.d.). *PostgreSQL*. Retrieved from https://www.postgresql.org/

IX. Appendices

A. Deliverables and Milestones

The following items below were the primary deliverables and milestones for this project. See also *Figure 4.0 - Zoho Projects Gantt Chart* for reference.

- Local environment
    - Local development environment setup
    - Initial bootstrapping of application components
- Chemical inventory web application
    - Front-end application
    - Back-end application
    - Database
- Cloud environment
    - Cloud account setup
    - Cloud environment setup
    - Cloud deployment

B. Budget

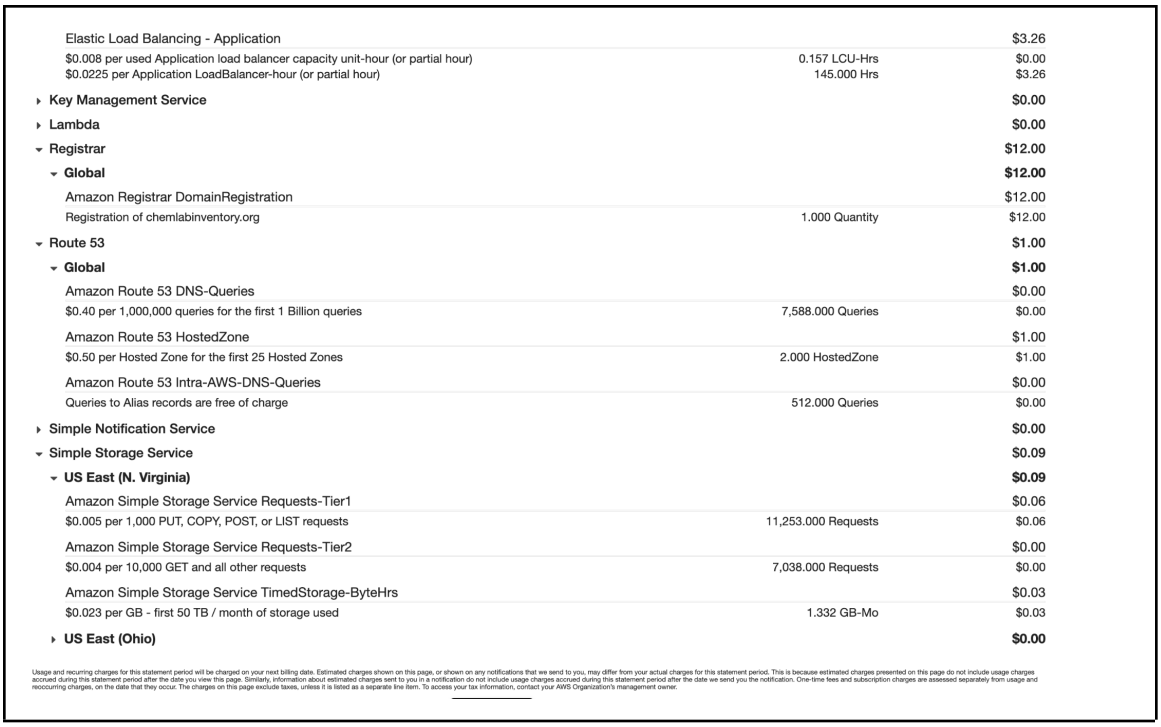
Given the limited client budget, this project has been conducted as a complimentary undertaking where typical costs associated with development time/effort and server resources shall be ceded for the most part. As such, the proposed system has been configured to stay within the AWS free tier limits or with minimal costs as much as possible. Any potential unforeseen costs shall be negotiated with the client. The table shows how a given AWS setup would translate to an estimated monthly cost.

Figure 9.0 - Sample AWS Billing (POC setup - April 2022)

Bills

Date: April 2022 — Download CSV — Print

Total		**$23.32**
▸ **Amazon Web Services, Inc. - Service Charges**		$23.32
Payment Summary		
▸ **Payment Summary**		
Details — + Expand All		
AWS Service Charges		**$23.32**
▸ **CloudFront**		**$0.00**
▸ **CloudWatch**		**$0.00**
▸ **Data Transfer**		**$0.00**
▾ **Elastic Compute Cloud**		**$6.97**
▾ **US East (N. Virginia)**		**$6.97**
Amazon Elastic Compute Cloud running Linux/UNIX		$5.62
$0.0116 per On Demand Linux t2.micro Instance Hour	484.590 Hrs	$5.62
EBS		$1.13
$0.05 per GB-Month of snapshot data stored - US East (Northern Virginia)	7.731 GB-Mo	$0.39
$0.10 per GB-month of General Purpose SSD (gp2) provisioned storage - US East (Northern Virginia)	7.444 GB-Mo	$0.74
Elastic IP Addresses		$0.22
$0.00 per Elastic IP address not attached to a running instance for the first hour	1.000 Hrs	$0.00
$0.005 per Elastic IP address not attached to a running instance per hour (prorated)	43.400 Hrs	$0.22



Elastic Load Balancing - Application		$3.26
$0.008 per used Application load balancer capacity unit-hour (or partial hour)	0.157 LCU-Hrs	$0.00
$0.0225 per Application LoadBalancer-hour (or partial hour)	145.000 Hrs	$3.26
Key Management Service		**$0.00**
Lambda		**$0.00**
Registrar		**$12.00**
Global		**$12.00**
Amazon Registrar DomainRegistration		$12.00
Registration of chemlabinventory.org	1.000 Quantity	$12.00
Route 53		**$1.00**
Global		**$1.00**
Amazon Route 53 DNS-Queries		$0.00
$0.40 per 1,000,000 queries for the first 1 Billion queries	7,588.000 Queries	$0.00
Amazon Route 53 HostedZone		$1.00
$0.50 per Hosted Zone for the first 25 Hosted Zones	2.000 HostedZone	$1.00
Amazon Route 53 Intra-AWS-DNS-Queries		$0.00
Queries to Alias records are free of charge	512.000 Queries	$0.00
Simple Notification Service		**$0.00**
Simple Storage Service		**$0.09**
US East (N. Virginia)		**$0.09**
Amazon Simple Storage Service Requests-Tier1		$0.06
$0.005 per 1,000 PUT, COPY, POST, or LIST requests	11,253.000 Requests	$0.06
Amazon Simple Storage Service Requests-Tier2		$0.00
$0.004 per 10,000 GET and all other requests	7,038.000 Requests	$0.00
Amazon Simple Storage Service TimedStorage-ByteHrs		$0.03
$0.023 per GB - first 50 TB / month of storage used	1.332 GB-Mo	$0.03
US East (Ohio)		**$0.00**

Usage and recurring charges for this statement period will be charged on your next billing date. Estimated charges shown on this page, or shown on any notifications that we send to you, may differ from your actual charges for this statement period. This is because estimated charges presented on this page do not include usage charges accrued during this statement period after the date you view this page. Similarly, information about estimated charges sent to you in a notification do not include usage charges accrued during this statement period after the date we send you the notification. One-time fees and subscription charges are assessed separately from usage and reoccurring charges, on the date that they occur. The charges on this page exclude taxes, unless it is listed as a separate line item. To access your tax information, contact your AWS Organization's management owner.

C. Qualifications

- IT professional with over 13 years of experience delivering enterprise solutions for various clients in the Philippines and abroad
- Full-stack developer with experience in Angular/Typescript, Java, Spring, MySQL, PostgreSQL, AWS
- Solutions Architect with experience working with clients in Enterprise Architecture teams, project consulting, and business development initiatives
- AWS Certified Solutions Architect Associate
- Learning agenda:

    - Recent Angular design and coding implementations
    - Latest UI/UX design conventions
    - API security without enterprise security products (Okta)
    - REST implementation in greater depth and to different domain context
    - Maximizing AWS free tier capabilities
    - Potential integration to public datasets if available
- Bachelor in Business Administration Major in Information Technology, Southville Foreign University, Las Piñas, 2008

D. Contributors / Collaborators

This project shall be conducted individually with no other contributors.

E. Resources

The following software, tools, and technologies were utilized for local software design and development.

Table 9.0 - Tools and frameworks

Development machine	Apple Macbook Pro
Dev tools - IDE	IntelliJ IDEA
Dev tools - JDK	JDK 17 (Amazon Corretto 17)

Dev tools - Framework	Spring Boot
Dev tools - Typescript	Angular 9 / Node 12 runtimes
Local web server	npm runtime
Local application server	Spring Boot Tomcat
Local database server	PostgreSQL 11
Source control	Github
API client	Postman
Cloud	Amazon Web Services (AWS)

F. Complete Program Listing

The application source codes have been stored in private personal Github repositories as part of the software development process, as shown below for preview purposes.

Source Code Previews

Figure 9.1 - Front-end source code preview

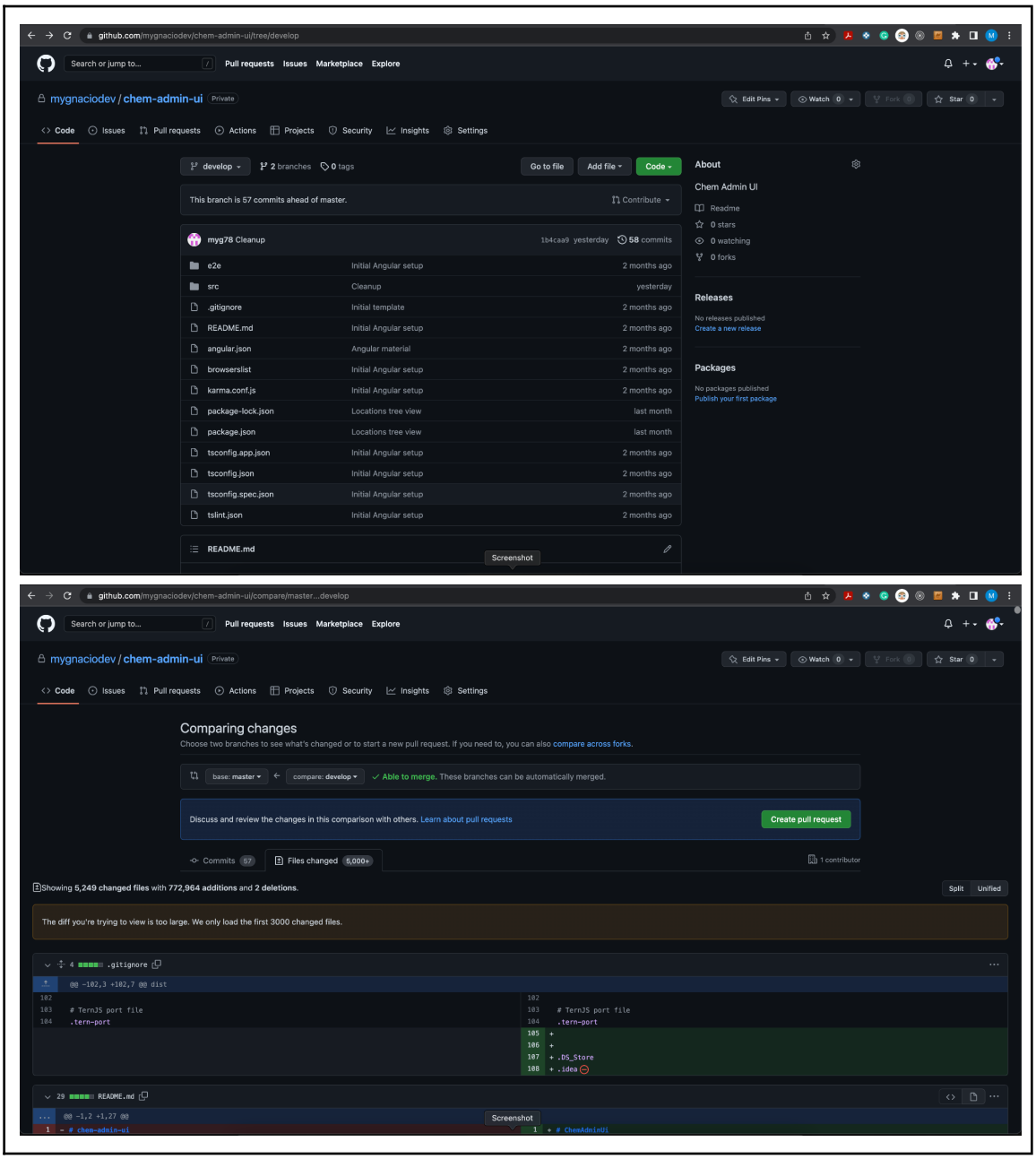
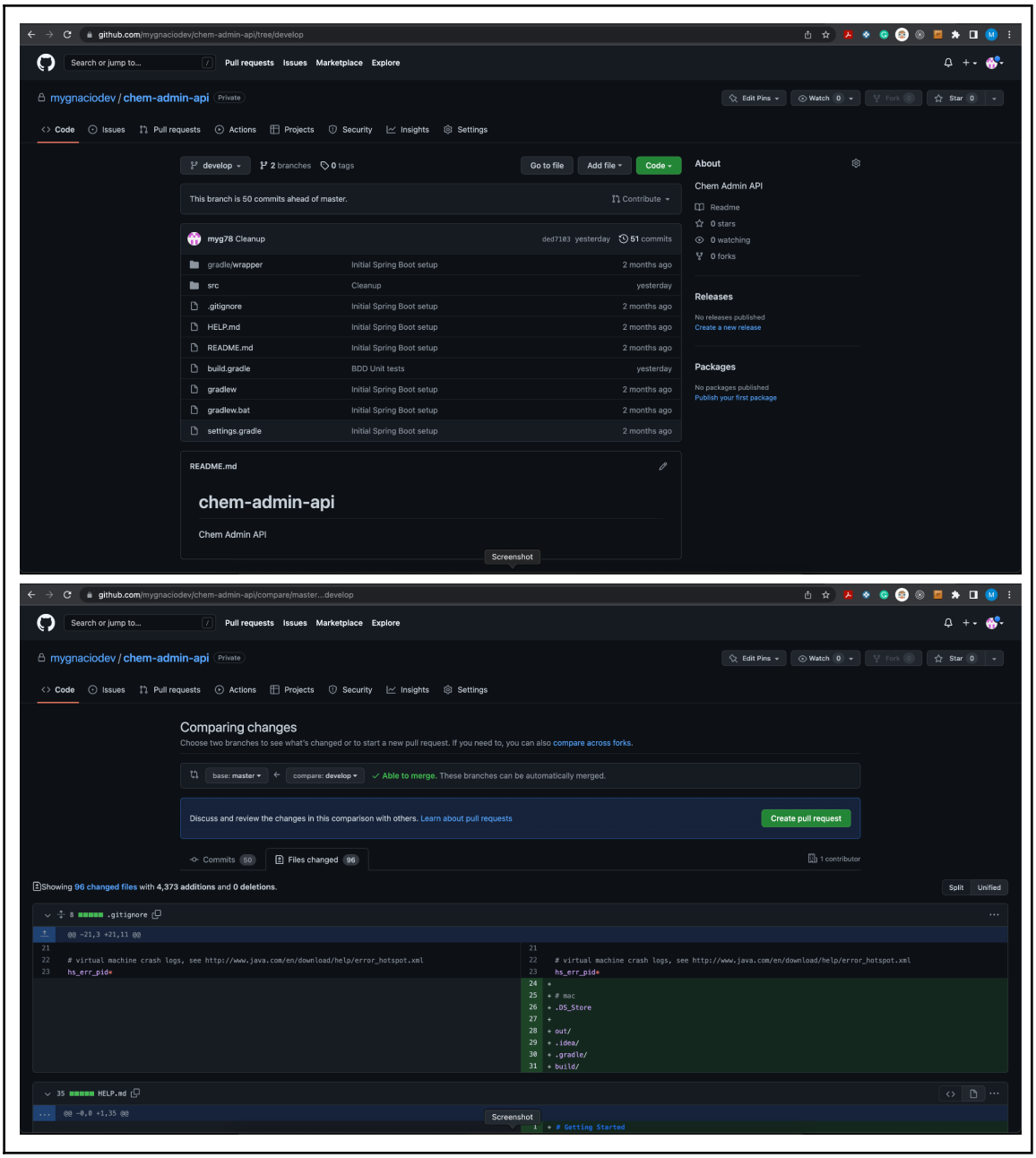
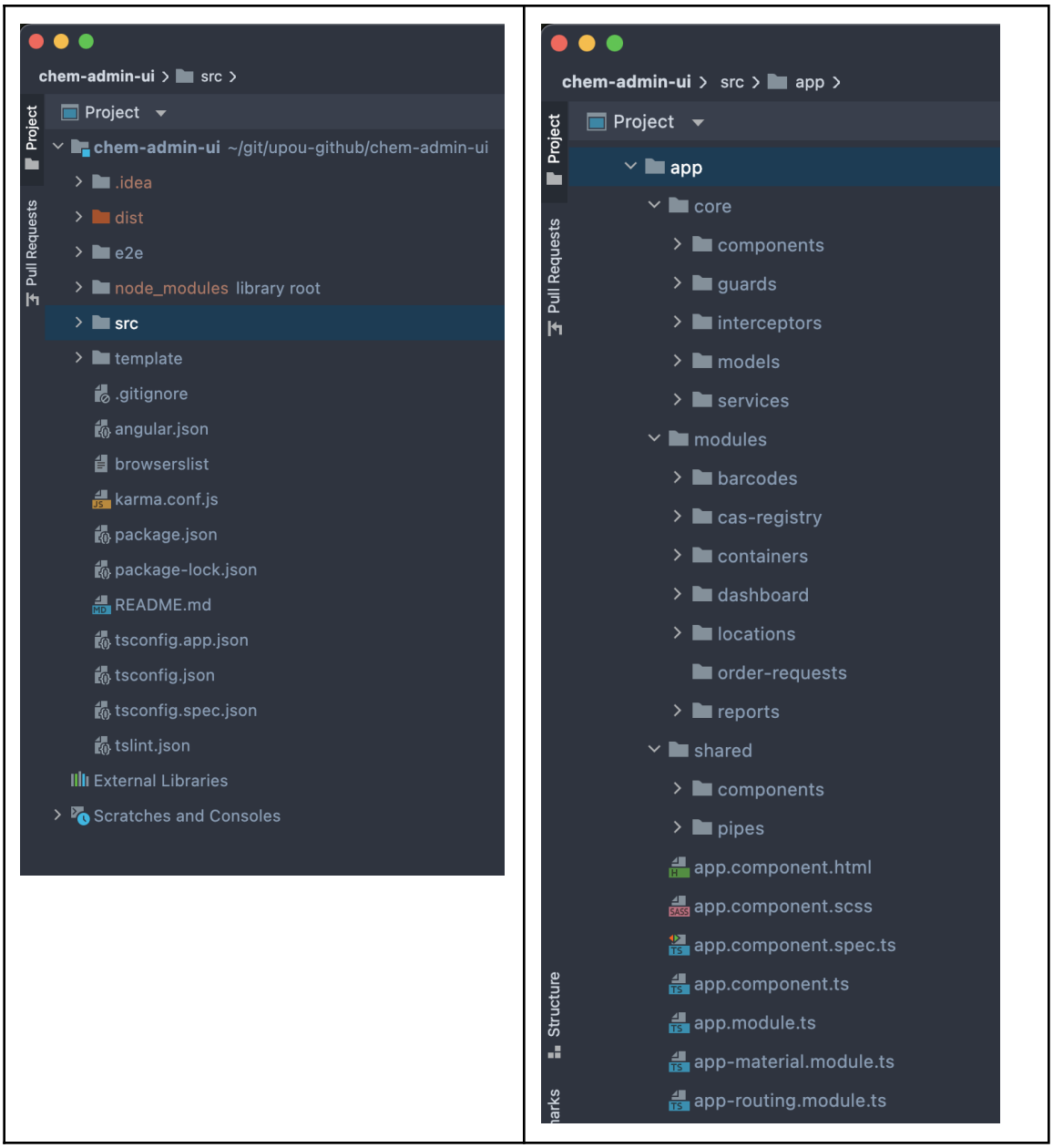


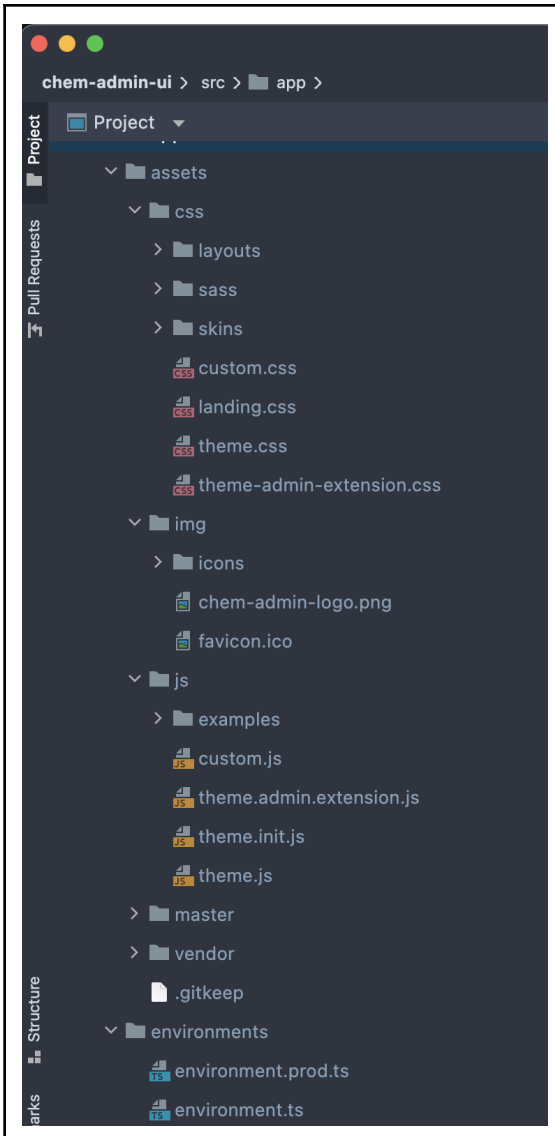
Figure 9.2 - Back-end source code preview



Given the significant number of program files, the complete program listing or source codes shall be provided separately via a compressed package containing the raw source files. The application directory structures are shown below for preview purposes.

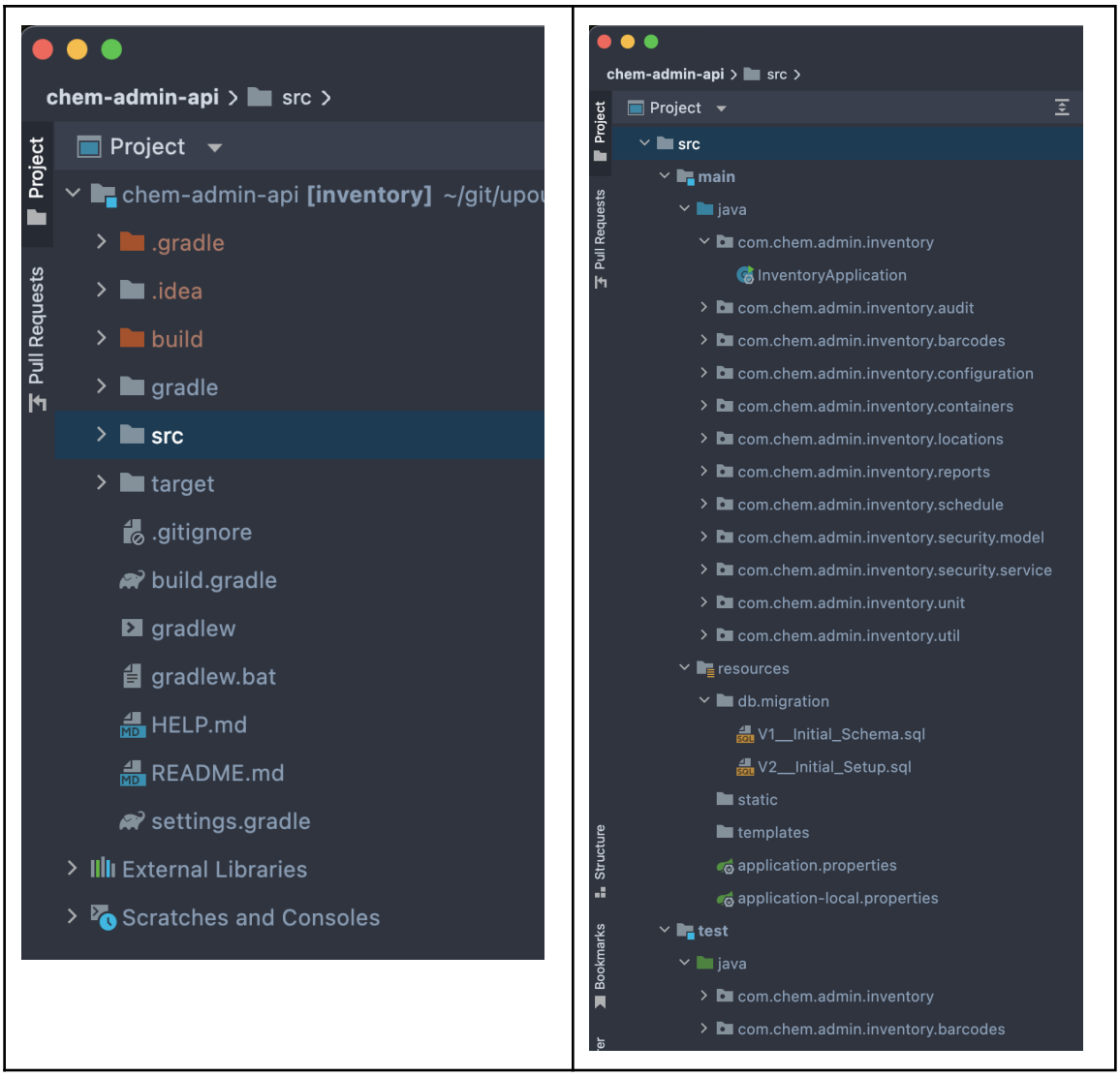
Figure 9.3 - Front-end application directory structure preview




chem-admin-ui > src > app >
Project
Project
Pull Requests
Structure
arks
assets
css
layouts
sass
skins
custom.css
landing.css
theme.css
theme-admin-extension.css
img
icons
chem-admin-logo.png
favicon.ico
js
examples
custom.js
theme.admin.extension.js
theme.init.js
theme.js
master
vendor
.gitkeep
environments
environment.prod.ts
environment.ts


package.json
{
"name": "chem-admin-ui",
"version": "0.0.0",
"scripts": {
"ng": "ng",
"start": "ng serve",
"build": "ng build",
"test": "ng test",
"lint": "ng lint",
"e2e": "ng e2e"
},
"private": true,
"dependencies": {
"@angular/animations": "~9.1.13",
"@angular/cdk": "^9.2.4",
"@angular/common": "~9.1.13",
"@angular/compiler": "~9.1.13",
"@angular/core": "~9.1.13",
"@angular/forms": "~9.1.13",
"@angular/material": "^9.2.4",
"@angular/platform-browser": "~9.1.13",
"@angular/platform-browser-dynamic": "~9.1.13",
"@angular/router": "~9.1.13",
"@circlon/angular-tree-component": "^11.0.4",
"ramda-adjunct": "^3.0.0",
"rxjs": "~6.5.4",
"tslib": "^1.10.0",
"zone.js": "~0.10.2"
},
"devDependencies": {
"@angular-devkit/build-angular": "~0.901.13",
"@angular/cli": "~9.1.13",
"@angular/compiler-cli": "~9.1.13",
"@types/jasmine": "~3.5.0",
"@types/jasminewd2": "~2.0.3",
"@types/node": "^12.11.1",
"codelyzer": "^5.1.2",
"jasmine-core": "~3.5.0",
"jasmine-spec-reporter": "~4.2.1",
"karma": "~5.0.0",
"karma-chrome-launcher": "~3.1.0",
"karma-coverage-istanbul-reporter": "~2.1.0",
"karma-jasmine": "~3.0.1",
"karma-jasmine-html-reporter": "^1.4.2",
"prettier": "2.5.1",
"protractor": "~7.0.0",
"ts-node": "~8.3.0",
"tslint": "~6.1.0",
"typescript": "~3.8.3"
}
}

Figure 9.4 - Back-end application package structure preview



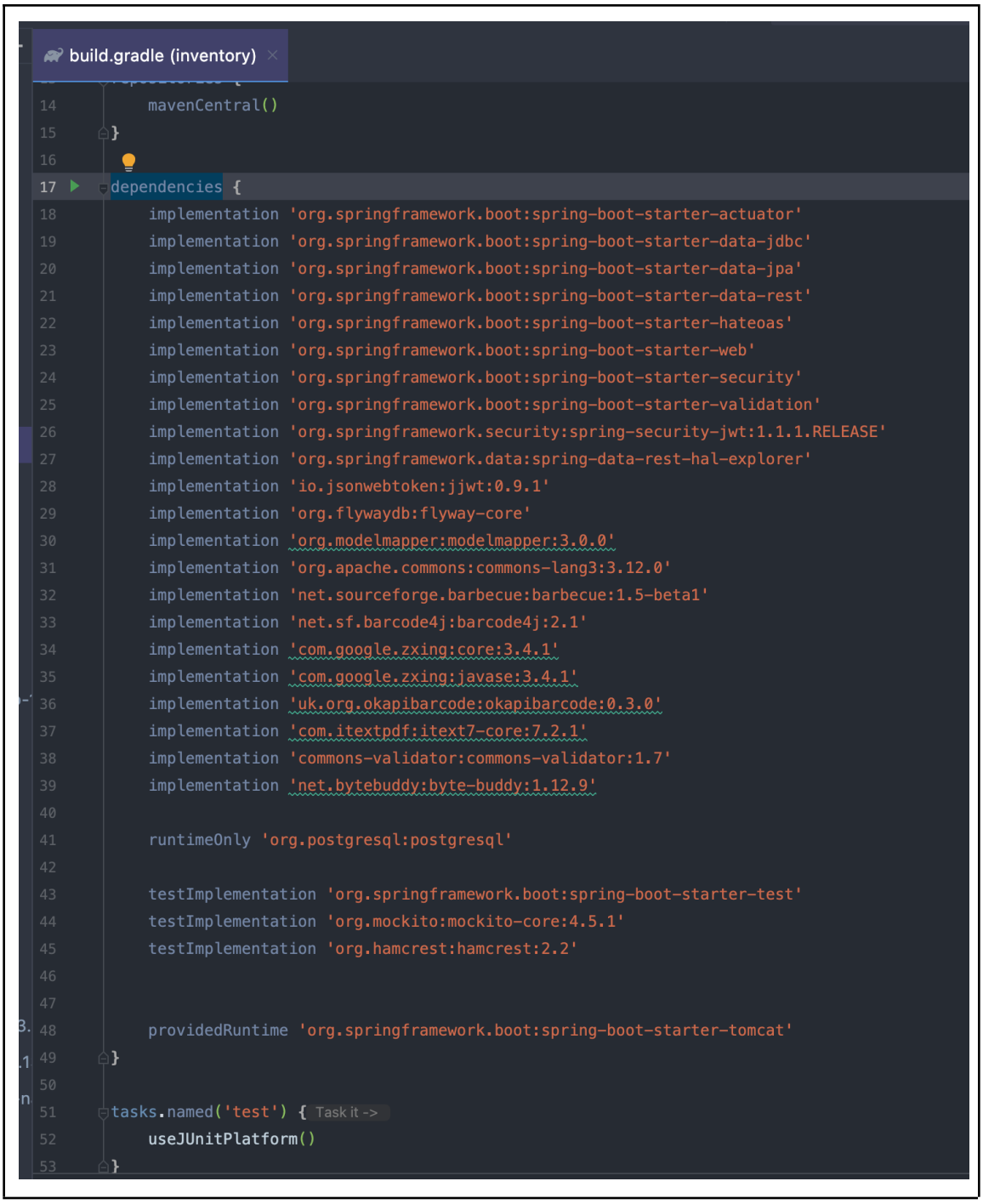

build.gradle (inventory)
mavenCentral()
}
dependencies {
implementation 'org.springframework.boot:spring-boot-starter-actuator'
implementation 'org.springframework.boot:spring-boot-starter-data-jdbc'
implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
implementation 'org.springframework.boot:spring-boot-starter-data-rest'
implementation 'org.springframework.boot:spring-boot-starter-hateoas'
implementation 'org.springframework.boot:spring-boot-starter-web'
implementation 'org.springframework.boot:spring-boot-starter-security'
implementation 'org.springframework.boot:spring-boot-starter-validation'
implementation 'org.springframework.security:spring-security-jwt:1.1.1.RELEASE'
implementation 'org.springframework.data:spring-data-rest-hal-explorer'
implementation 'io.jsonwebtoken:jjwt:0.9.1'
implementation 'org.flywaydb:flyway-core'
implementation 'org.modelmapper:modelmapper:3.0.0'
implementation 'org.apache.commons:commons-lang3:3.12.0'
implementation 'net.sourceforge.barbecue:barbecue:1.5-beta1'
implementation 'net.sf.barcode4j:barcode4j:2.1'
implementation 'com.google.zxing:core:3.4.1'
implementation 'com.google.zxing:javase:3.4.1'
implementation 'uk.org.okapibarcode:okapibarcode:0.3.0'
implementation 'com.itextpdf:itext7-core:7.2.1'
implementation 'commons-validator:commons-validator:1.7'
implementation 'net.bytebuddy:byte-buddy:1.12.9'
runtimeOnly 'org.postgresql:postgresql'
testImplementation 'org.springframework.boot:spring-boot-starter-test'
testImplementation 'org.mockito:mockito-core:4.5.1'
testImplementation 'org.hamcrest:hamcrest:2.2'
providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
}
tasks.named('test') { Task it ->
useJUnitPlatform()
}

Table 9.1 - Web Frameworks (UI)

Single page application framework (SPA)	Angular
UI/UX components	Angular Material Porto admin website template
Utility plugins	JQuery utilities Datatables Angular tree Flot charts

Table 9.2 - Application Frameworks (API)

Application development	Spring Boot
RESTful APIs	Spring Boot Starter Web
Relational data model	Spring JPA / Hibernate Spring JDBC
Database setup / migration	Flyway
Security	Spring Security (Authentication) JSON Web Tokens (Authorization) Spring Web MVC (CORS) Spring Security (Bcrypt)
Barcodes	Barbecue Barcode4j Googel zxing Okapibarcode IText
Scheduled process	Spring Scheduling (Cron)
Audit	Spring Data (Auditor)
Utilities	Apache Commons Model Mapper
Testing	JUnit Mockito Hamcrest

Application build	Gradle

Table 9.3 - 3rd Party Integration

CAS Registry	Service Integration to CAS Common Chemistry public API endpoints upon successful access request grant https://commonchemistry.cas.org

G. Technical Reference

POC System Specifications

The system has been initially deployed using a standard AWS distributed architecture as a proof-of-concept (POC) for utilizing various AWS services.

- Front-end application - AWS Cloudfront distribution (CDN) serving web content hosted in an AWS S3 bucket with static website hosting enabled
- Back-end application - AWS Application Load Balancer (ALB) routing traffic to an AWS EC2 server for application and database
- SSL/TLS encryption - AWS ACM certificates to support HTTPS traffic on Cloudfront distribution (in front of S3 website) and AWS load balancer (in front of EC2 server)

However, upon further evaluation, this approach would not serve as a cost-effective solution in the long term since the AWS free-tier limits typically expire in 12 months. Also, certain AWS-managed components incur unavoidable costs such as the ALB hourly cost given no option to scale down, CDN invalidations if needed during releases, and unutilized Elastic IP addresses when EC2 instance is turned off. See Budget section above for costs associated with this setup.

Table 9.4 - POC front-end application

Hardware	Not Applicable - AWS S3
Operating System	Not Applicable - AWS S3
Programming Languages	Typescript / Javascript
Frameworks	Angular 9
Servers	Not Applicable - AWS Cloudfront distribution + AWS S3 static website
Hostname	https://chemlabinventory.org

Table 9.5 - POC Back-end application

Hardware	AWS Application Load Balancer (ALB) AWS EC2 server t2.micro instance type 1 vCPU, 1 GiB Mem 8 GiB EBS storage
Operating System	Amazon Linux 2
Programming Languages	Java 17
Frameworks	Spring Boot

Servers	Apache Tomcat application server PostgreSQL database server
Hostname	https://api.chemlabinventory.org

Final System Specifications

To account for cost limitations, the final system has been deployed into a simpler deployment architecture consisting of a single EC2 server hosting the front-end, back-end and database components. This also allows scaling down or turning off the server during off-peak hours, e.g. during nighttime outside of regular school hours, to minimize on-demand server utilization costs. This systems configuration allows for the minimum viable cost-effective setup for the client over the long run.

Table 9.6 - Final setup

Hardware	AWS EC2 server t2.micro instance type 1 vCPU, 1 GiB Mem 8 GiB EBS storage
Operating System	Amazon Linux 2
Programming Languages	Typescript / Javascript Java 17
Frameworks	Angular Spring Boot
Servers	Apache HTTP web server Apache Tomcat application server PostgreSQL database server

Hostnames	https://app.chemlabinventory.org https://app.chemlabinventory.org:8443/api
Devices	Canon printer Canon Pixma TS6420 Barcode scanner Eyoyo Mini Laser Bluetooth scanner

Maintenance Plan for the Software System

Table 9.7 - UI Build

Package dependencies are managed through node package manager (npm).

Pre-requisites	**Software:** nvm v0.39.0 or higher node v12.13.1 or higher npm v6.12.1 or higher Angular v9.1.13 or higher
UI compile	**Command:** npm install
UI run	**Command:** npm start
Environment properties	**File:** src/environments/environment.ts src/environments/environment.prod.ts **Action:** Change `apiUrl` to corresponding hostname of API server. Indicate correct protocol (http or https) and port (80 or 8080 or 8443) if necessary.
UI production build	**Command:**

	ng build --aot --configuration production **Actions:** Verify build artifacts were generated in `dist` folder. Zip/compress `chem-admin-ui` folder in preparation for file transfer to remote server.
Deployment	**Tool:** Secure file transfer utility, e.g. FileZilla **Pre-requisites:** SSH network access to EC2 server (e.g. via security groups) PEM file available **Actions:** Connect to EC2 server via SFTP. Upload compressed file to EC2 server on default user home directory. Unzip deployment file. Copy extracted files to web server directory.

Table 9.8 - API Build

Software dependencies are managed through the gradle wrapper (gradlew).

Pre-requisites	**Software:** Java 17 (Amazon Corretto 17) PostgreSQL v11.14 or higher Gradle v7.3.1 or higher **Database:** Empty `chem-admin` database Db user `chem_app` with appropriate permissions
API compile	**Command:** gradlew build

API run	**Command:** gradlew bootRun
Environment properties	**File:** src/main/resources/application-local.properties src/main/resources/application.properties **Action:** Change database username and password to corresponding values.
CORS config	**File:** src/main/java/com/chem/admin/inventory/configuration/CorsConfig.java **Action:** Add UI hostname to allowed origins
API production build	**Command:** gradle bootWar **Actions:** Verify build artifact was generated in `build/libs` folder. Copy and rename war file to `ROOT.war` preparation for file transfer to remote server.
Deployment	**Tool:** Secure file transfer utility, e.g. FileZilla **Pre-requisites:** SSH network access to EC2 server (e.g. via security groups) PEM file available **Actions:** Connect to EC2 server via SFTP. Upload war file to EC2 server on default user home directory.

	If present, delete/undeploy existing ROOT.war and ROOT folder in tomcat webapps directory. Copy ROOT.war to tomcat webapps directory.

Table 9.9 - EC2 server setup/dependencies

Component	**Pre-requisite**
Server instance	Launch AWS EC2 instance with Amazon Linux 2 Configure appropriate network permissions via security groups (e.g. ports 80, 443, 8080, 8443, 22 from correponding source IP addresses)
Web server	Install and configure Apache HTTPD web server
Application server	Install and configure Apache Tomcat application server
Database server	Install and configure PostgreSQL
SLL/TLS setup	Install Certbot for issuing Let's encrypt SSL certificates Configure web server to support HTTPS Configure application server to support HTTPS
Server image / backup	Create AWS EC2 image to create backup once configurations are already in place

Table 9.10 - EC2 server notable directories or files

Component	**Path**
ec2-user home	/home/ec2-user
HTTPD configuration	/etc/httpd/conf/httpd.conf
HTTPD deployment	/var/www/html
HTTPD logs	/etc/httpd/logs

Tomcat configuration	/usr/share/tomcat/conf/server.xml
Tomcat deployment	/usr/share/tomcat/webapps
Tomcat logs	/usr/share/tomcat/logs
Postgres configuration	/var/lib/pgsql/12/data/pg_hba.conf
Let's Encrypt SSL	/etc/letsencrypt/live/

User management

Features related to managing system users have been deprioritized due to the shift in priorities as well as the low impact of user management given the small number of initial users upon first release. As such, user management shall be handled manually by the system administrator through the following:

- User accounts shall be added manually in the database. See `V2__Initial_Setup.sql` for sample user data records.
- Users will have to contact the system admin to reset passwords. The admin can define a temporary password and set the `reset_password` column to true. This will then allow users to login using the temporary credentials but will be required to change their password accordingly.
- The system utilizes Bcrypt, an adaptive hashing algorithm to safely store passwords. As such, even the system admin has no way of decoding user passwords once these have been changed by the user.
- The system also utilizes various security features such as Spring Security, JSON web tokens, and user locking on max failed attempts.

Given these, future recommended improvements to the system shall include user registration, user management, and password reset capabilities directly from the application's web interface.

H. Features Checklist

Table 9.11 - Features

Feature	Release	Status
F-01: Container Management ● UC-01: Add container ● UC-02: Search container ● UC-03: View container ● UC-04: Edit container ● UC-05: Move container ● UC-06: Dispose container ● Container quick search *	Release 1.0 - Essential	Completed
F-02: Location management ● UC-07: View locations ● UC-08: View location containers ● UC-09: Add location ● UC-10: Edit locations directory structure ● Delete location * ● View unassigned *	Release 1.0 - Expected	Completed
F-03: Barcode management ● UC-11: Print barcodes ● UC-12: Generate barcodes ● Scan barcodes *	Release 1.0 - Expected	Completed
F-04: Reports ● UC-13: Export inventory (Active, Low,	Release 1.0 - Expected	Completed

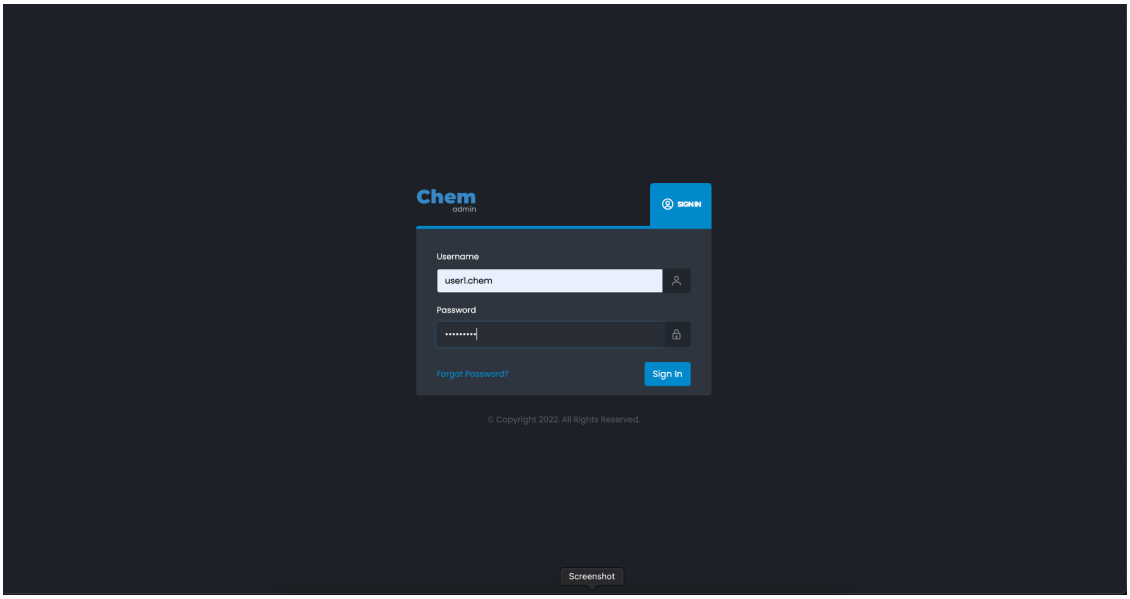
Disposed, Expiring) ● UC-14: Export barcodes assignment (Missing)		
F-05: Orders ● UC-15: Request order	Later release - Optional	Canceled. Replaced with Order reminders.
F-06: Notifications ● UC-16: Send Notifications	Later release - Optional	Completed
F-07: CAS references ● UC-17: Retrieve CAS reference ● CAS quick search / lookups *	Later release - Optional	Completed
Login * ● UC-18: User Login ● Reset password *	Release 1.0 - Essential	Completed
Dashboard * ● Show dashboard widgets ● Show inventory counts ● Show recent inventory totals ● Show recent containers ● Show low containers ● Show top containers/locations ● Show recent activity ● Show system status	Release 1.0 - Expected	Completed
Audit * ● Audit container changes	Release 1.0 - Expected	Completed
UI/UX site modernization * ● Modern look-and-feel	Release 1.0 - Expected	Completed

* Items marked with asterisk have not been defined or labeled in the original proposal but included in final implementation.

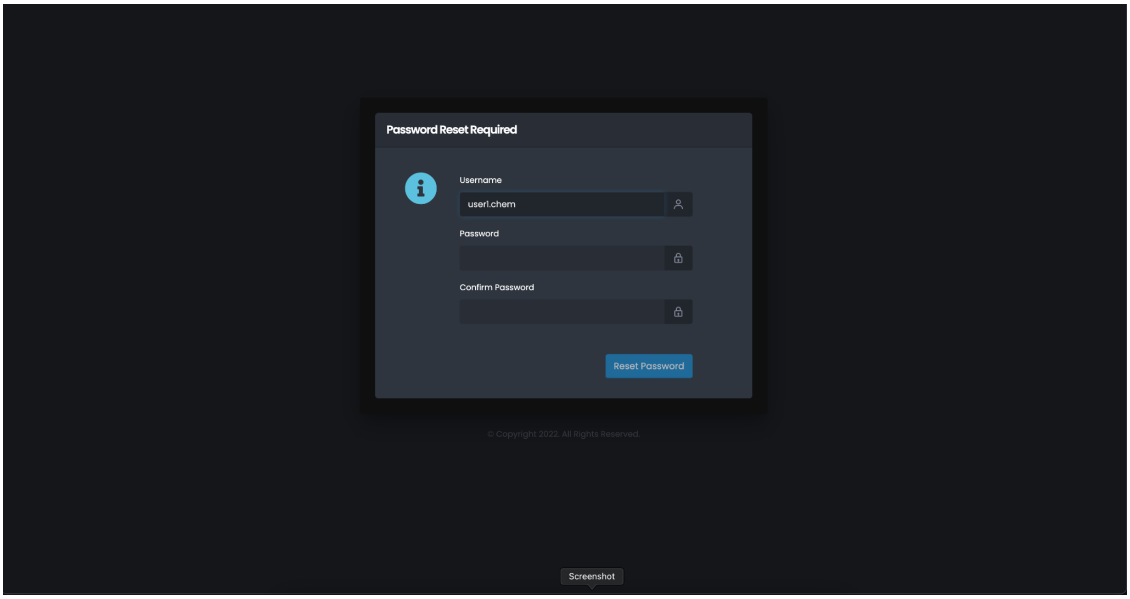
I. User Manual

The following section outlines the basic flows in the system:

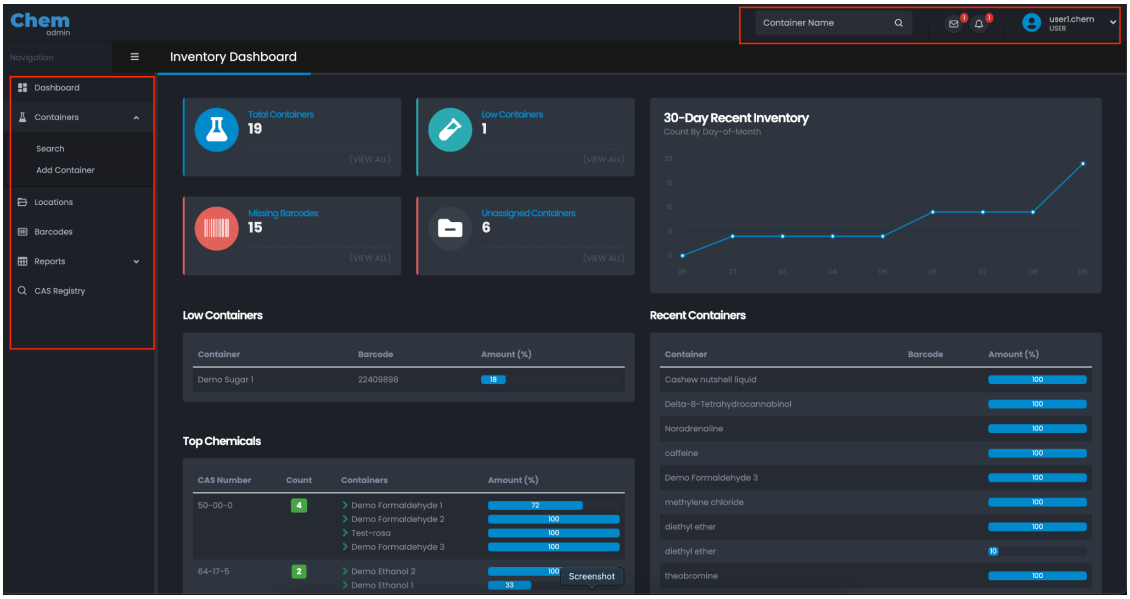
In the login screen, enter user credentials for username and password.



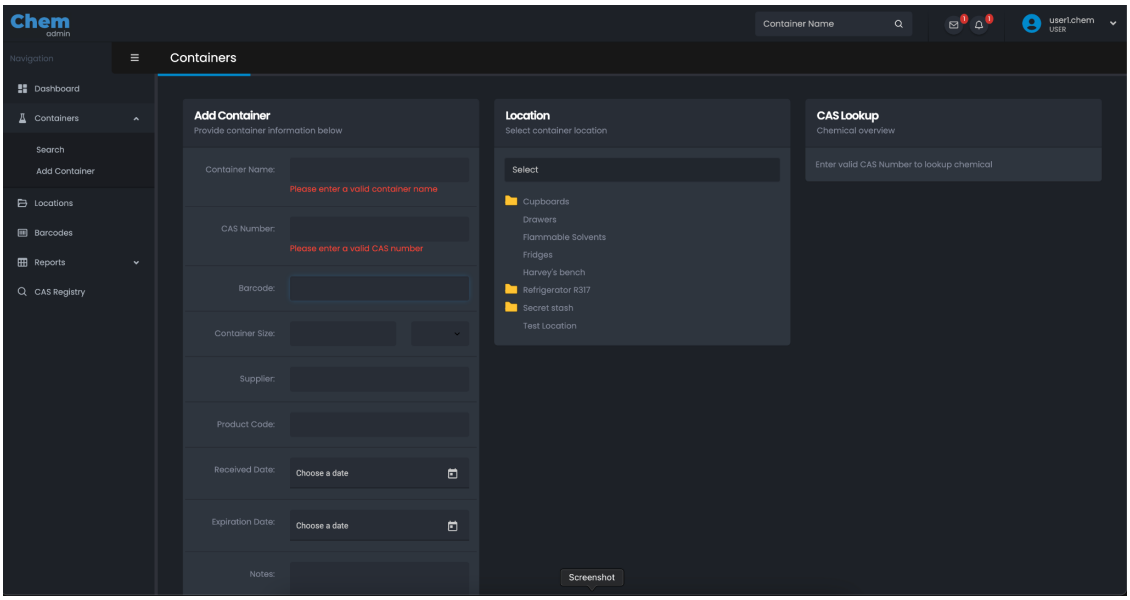
Upon first sign-in, enter and re-confirm the new password.



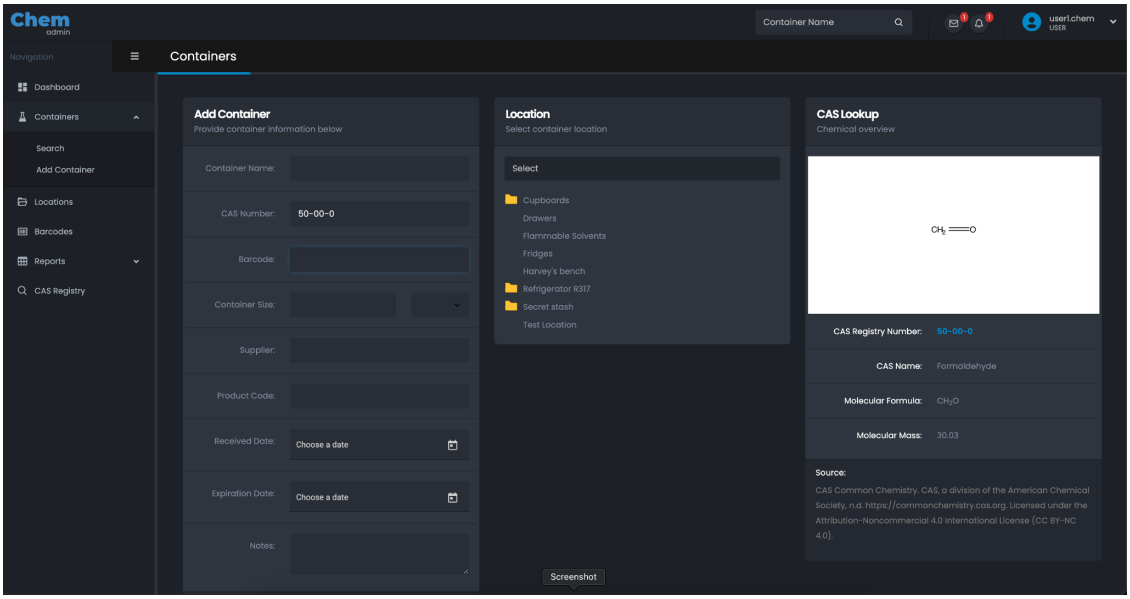
Upon successful login, the system displays the dashboard as the homepage. The top header displays the quick search box, notifications/alerts, login user name and role, and logout option. The left sidebar displays the navigation menu for the system features/modules. The center body displays the main active content.



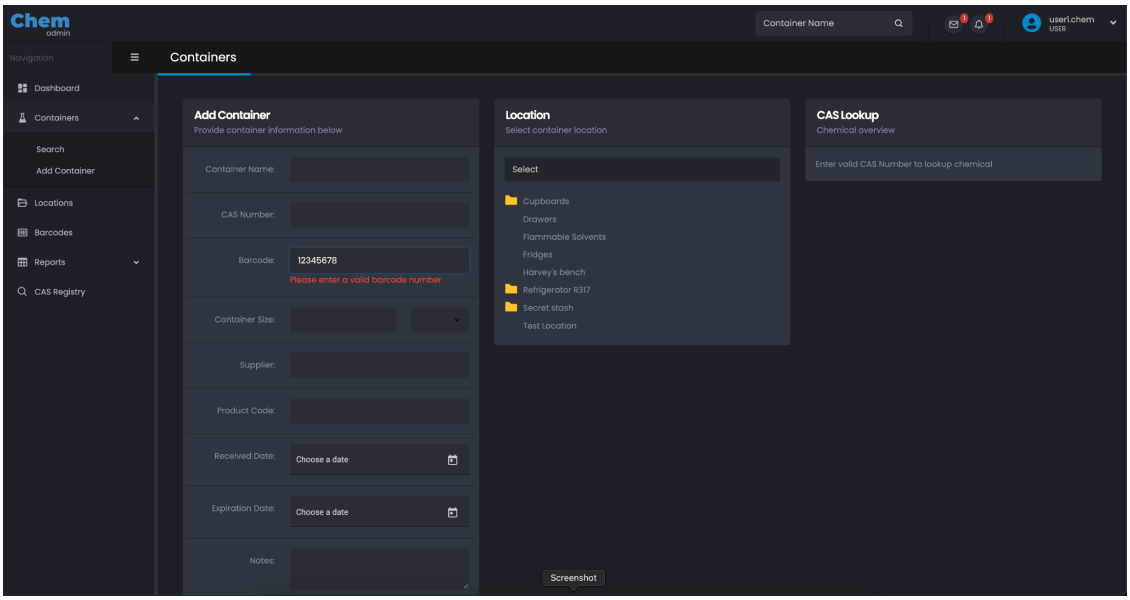
In the Add Containers module, enter corresponding container info, such as container name, CAS number, barcode, location, and so on.



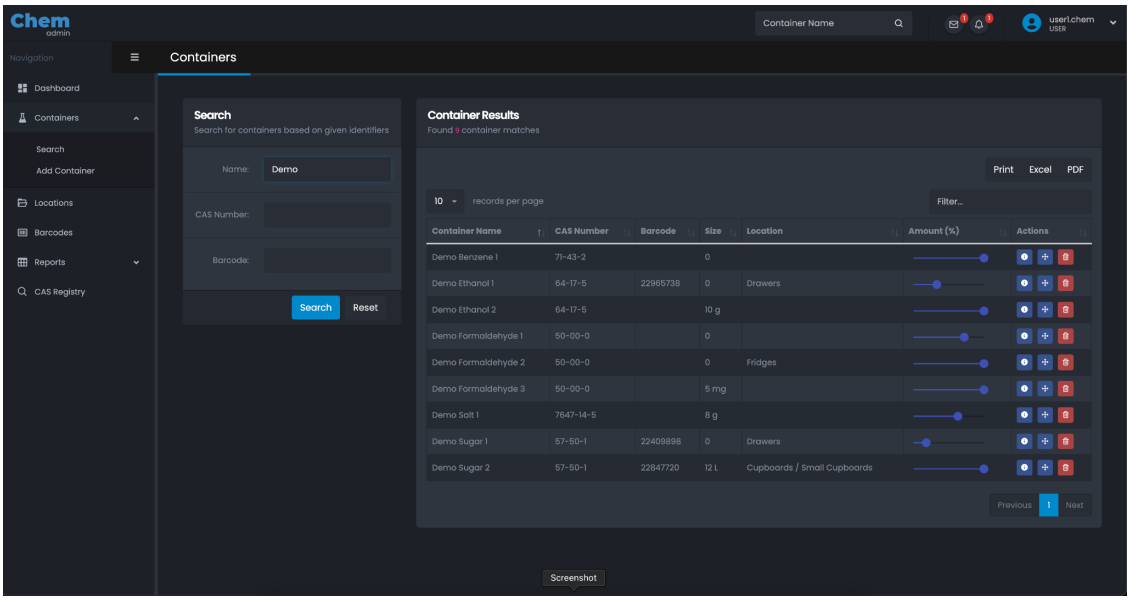
The CAS lookup section will display chemical info after a valid CAS number is inputted.

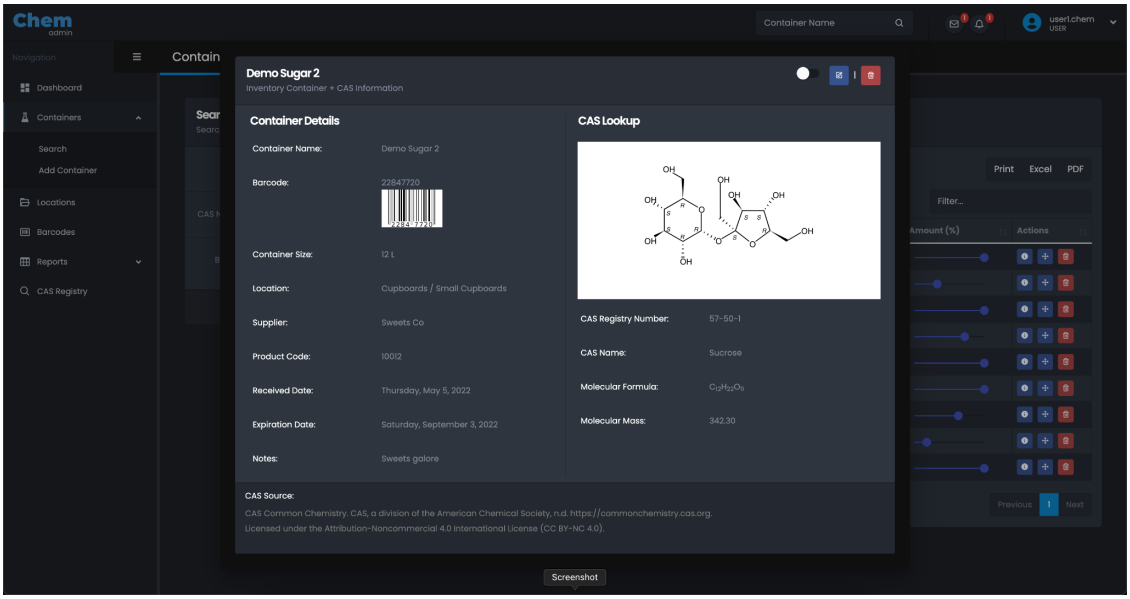


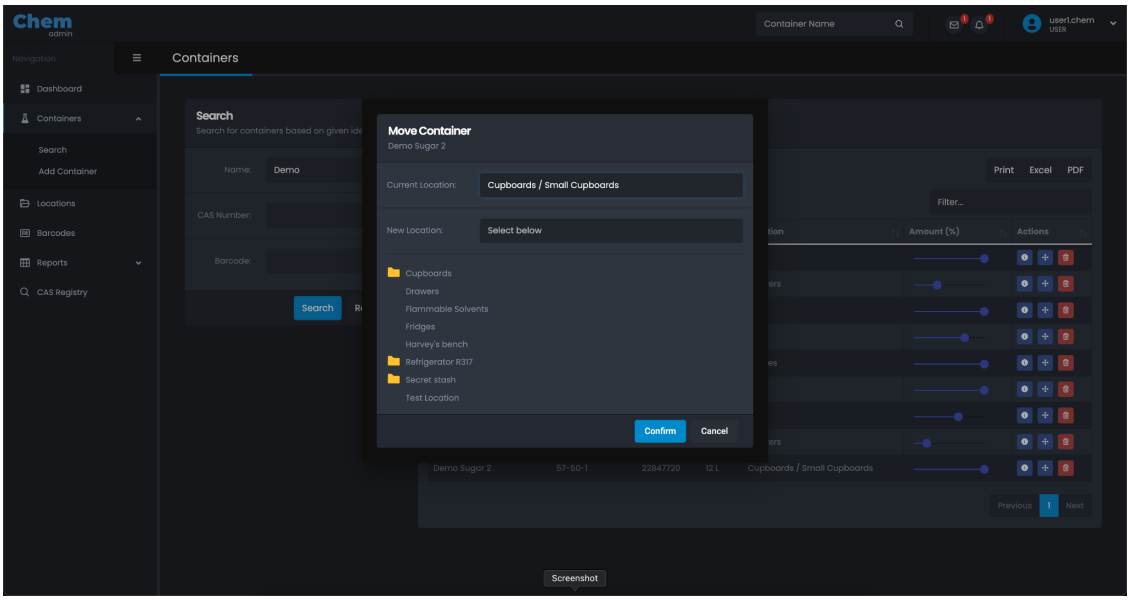
Barcodes are optional and can either be typed or scanned using any standard barcode scanner. Only valid barcode numbers, e.g. EAN-8 format, are accepted.

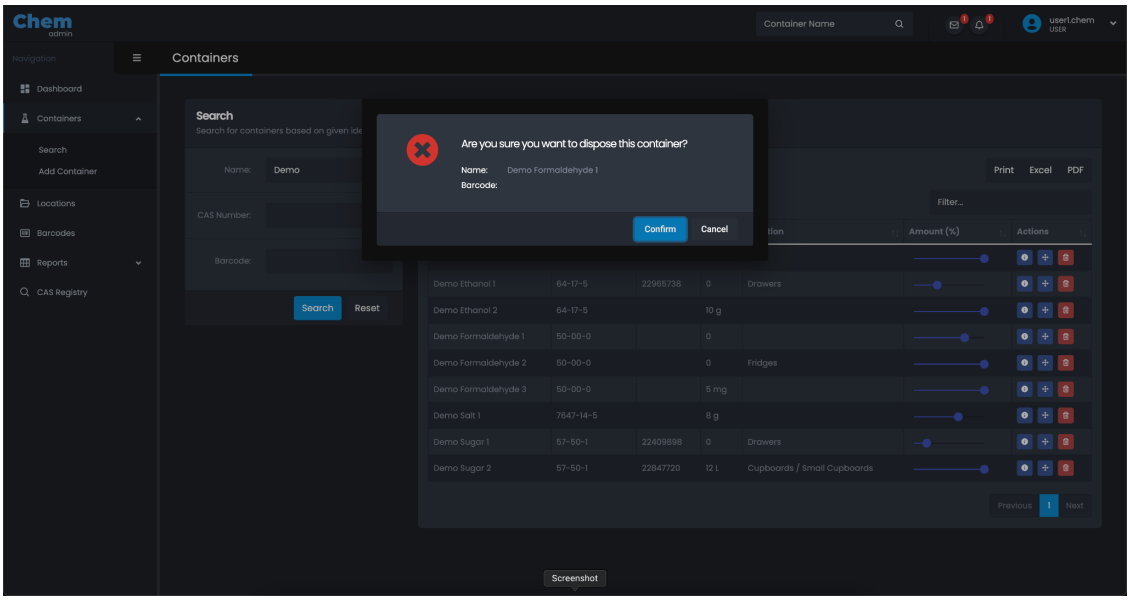


In the Search Containers module, enter the container name, CAS number, or barcode to find active containers. Each container can then be modified through various actions such as view/edit, move, dispose, or change level.

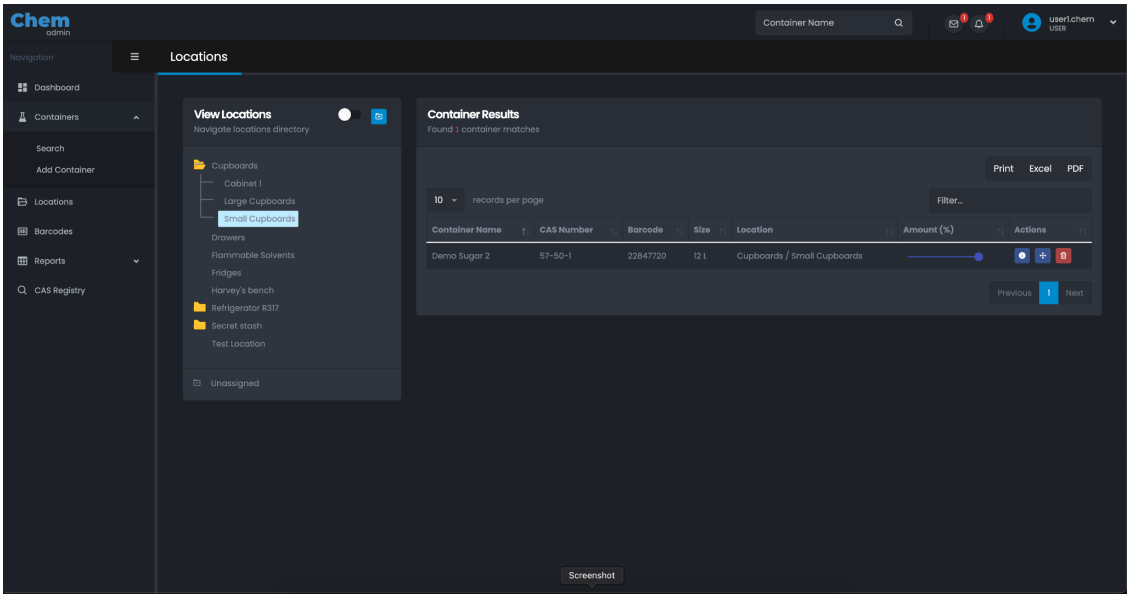




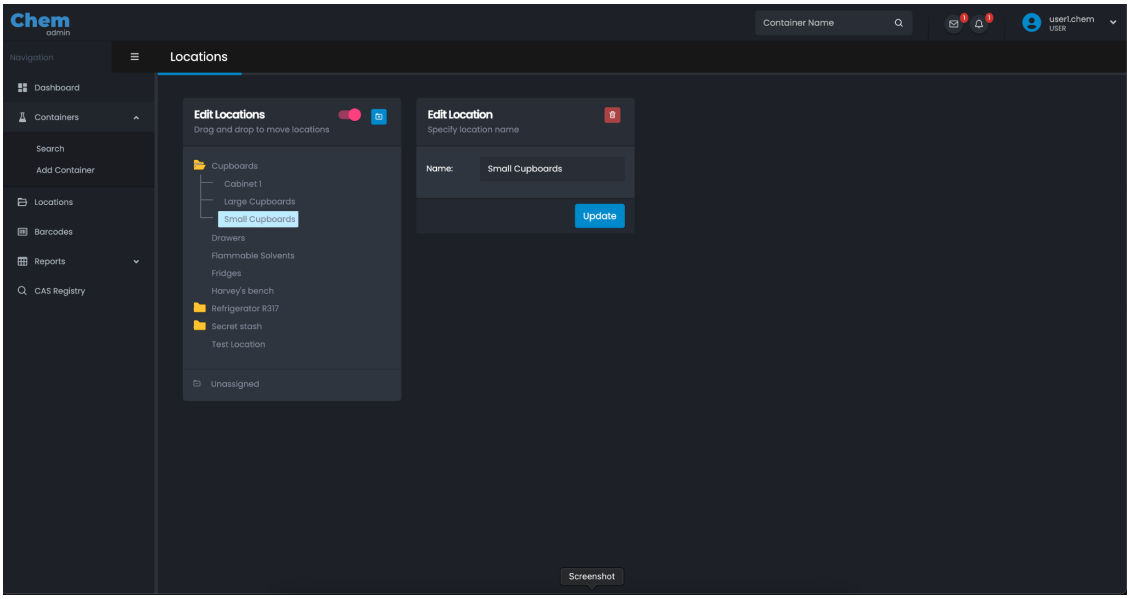




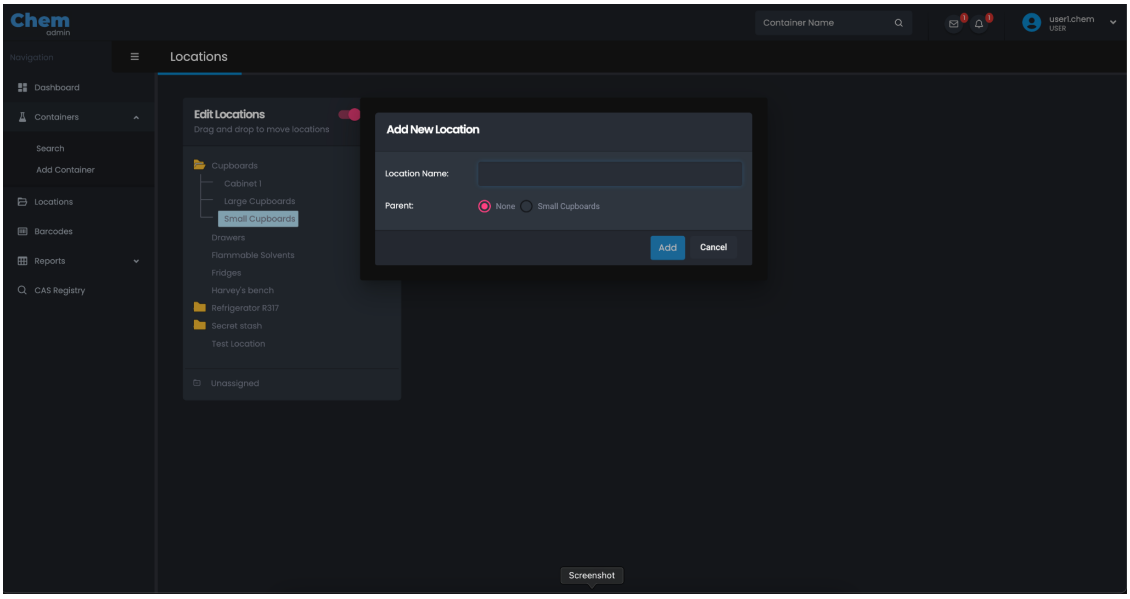
In the Locations module, navigate through the directory tree to view locations and assigned containers for a given location.



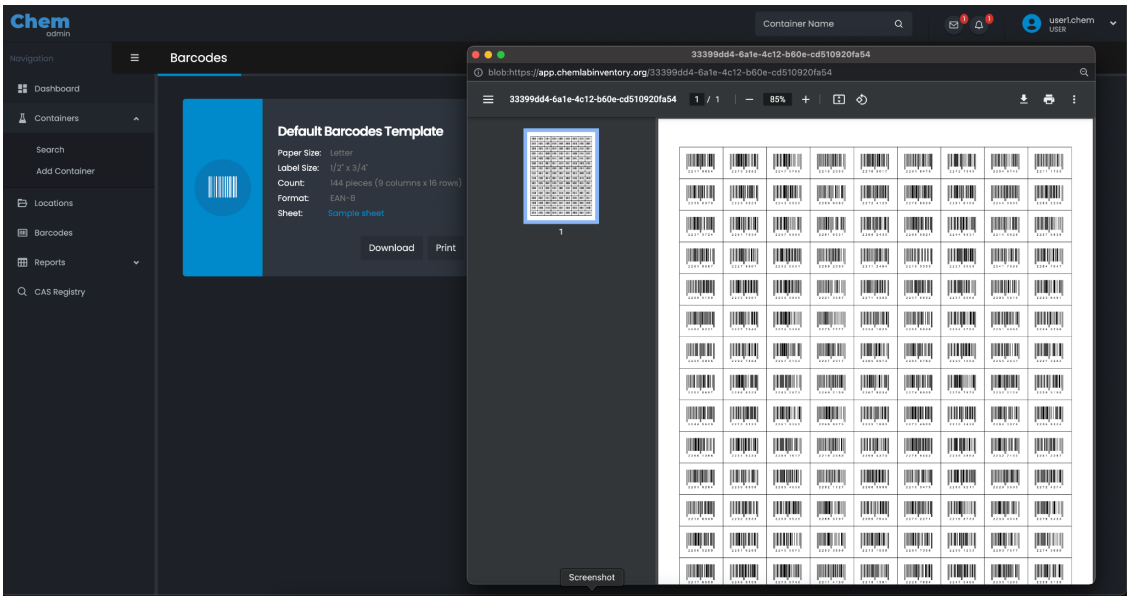
In the Locations module, click the edit toggle to edit/rename, move, or delete locations in the directory tree.

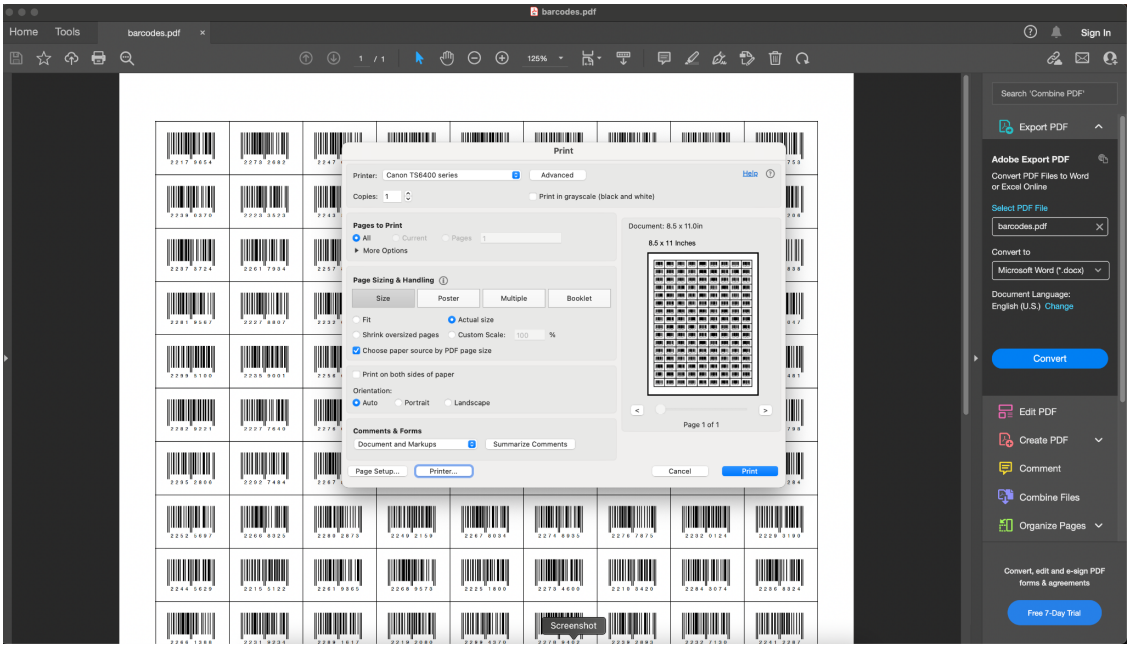


In the Locations module, locations can be added either at the root directory or under a given parent location.

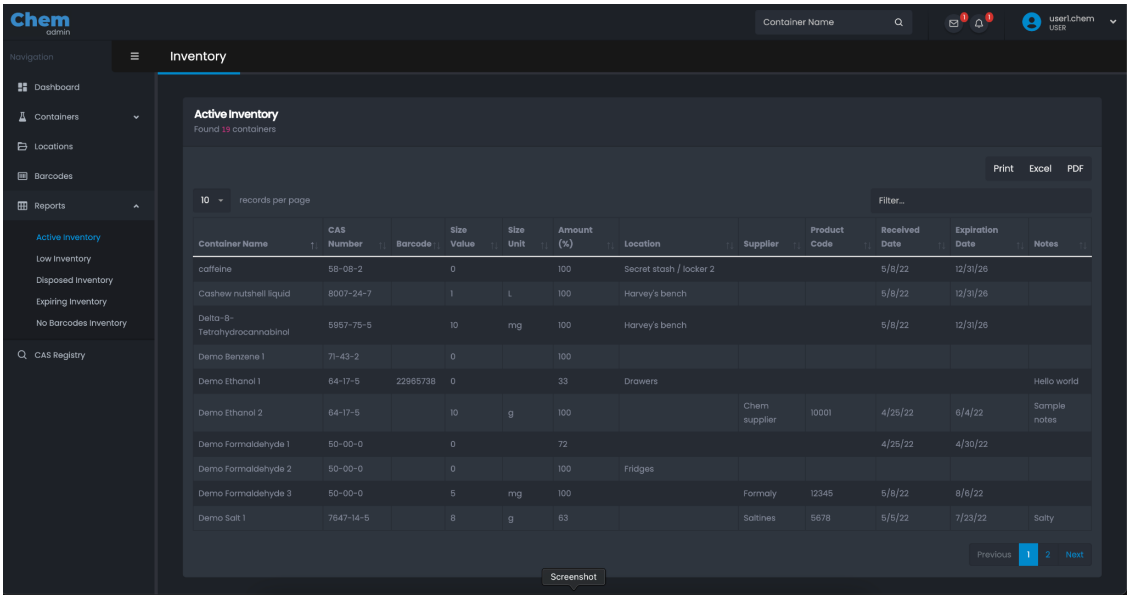


In the Barcodes module, download or print a pre-defined barcode sheet.

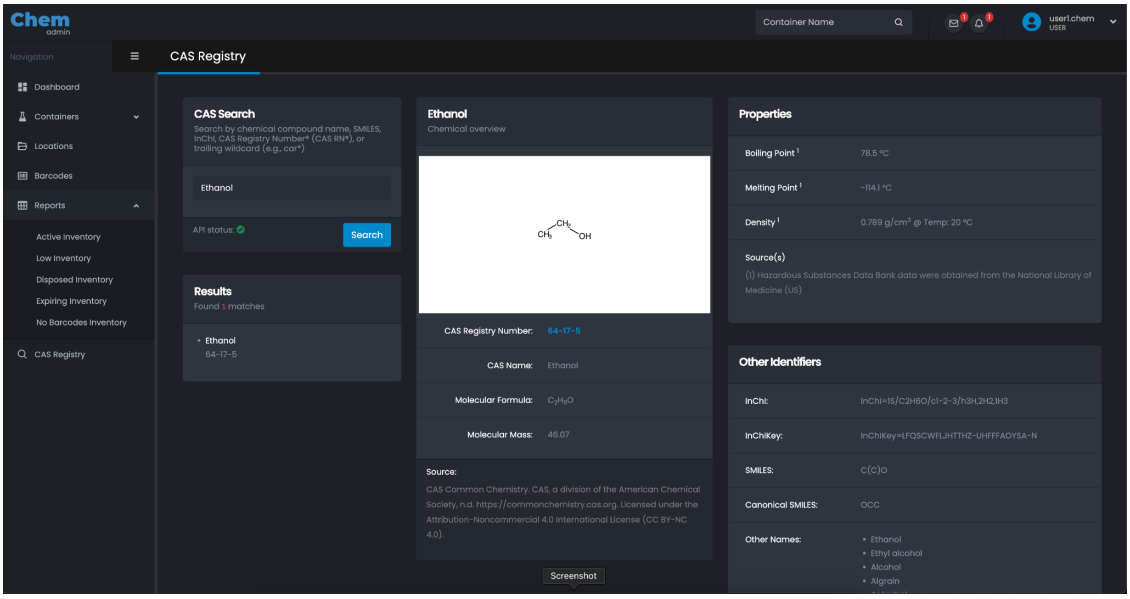




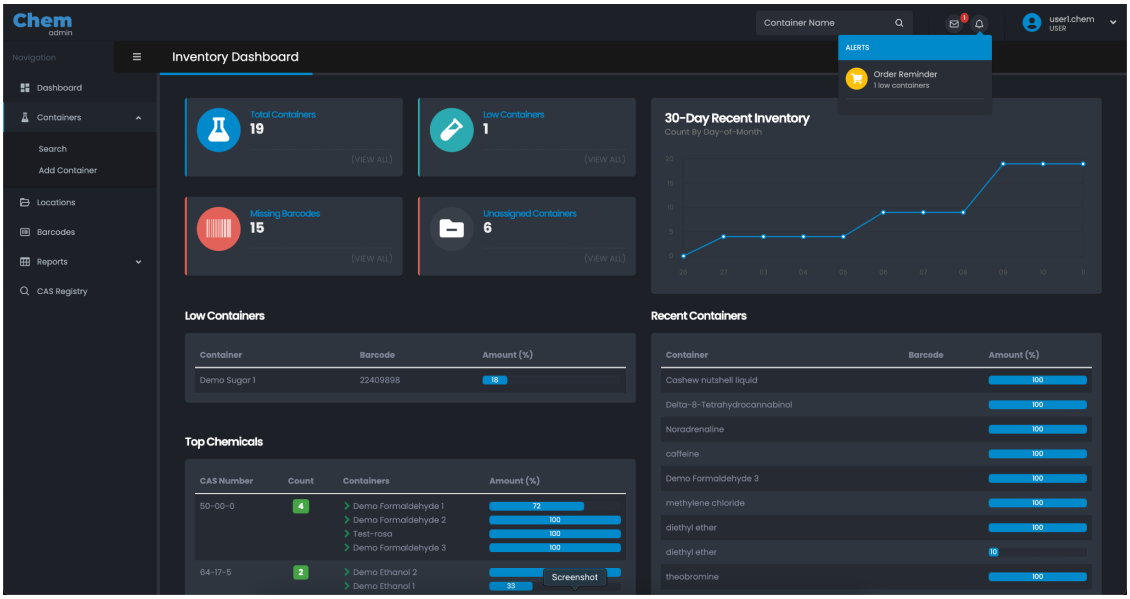
In the Reports module, click on a given report to view the tabular dataset. The results table allows for filtering, printing, and downloading as Excel or PDF.

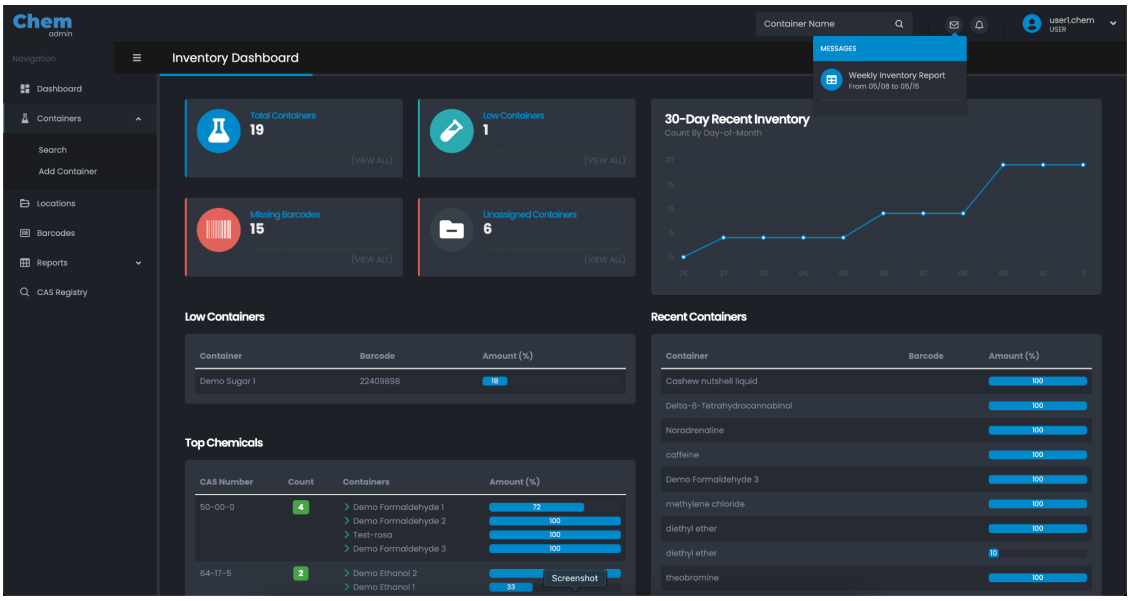


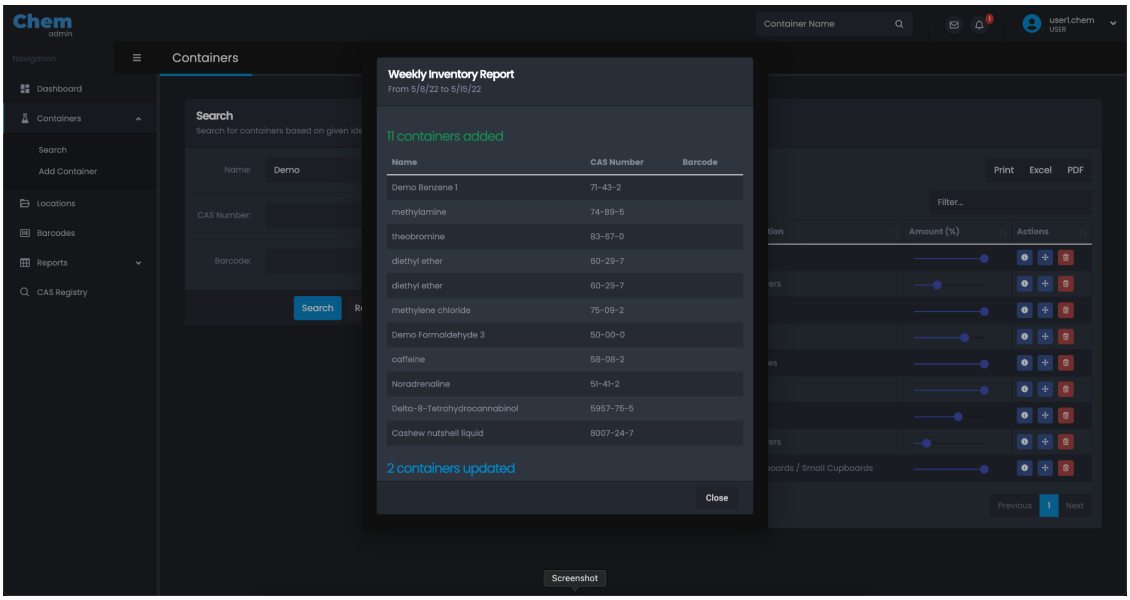
In the CAS Registry module, search for chemicals by name, CAS number, or wildcard search text.

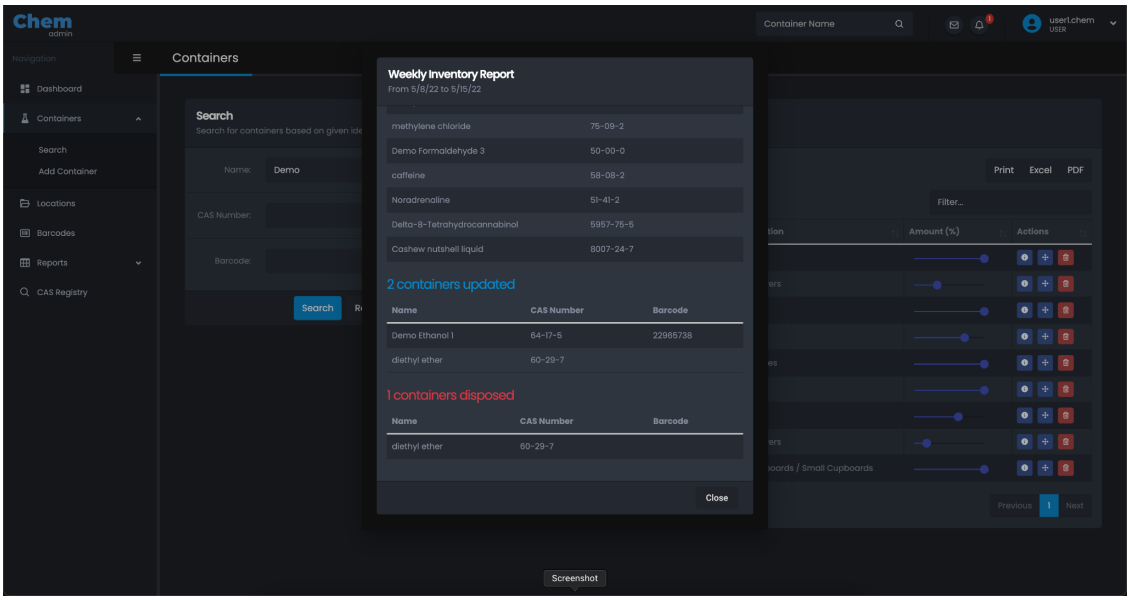


In the header section, click on the notifications icons to view messages and alerts.




Chem
admin
Container Name
user1.chem
USER
Navigation
Inventory Dashboard
MESSAGES
Weekly Inventory Report
From 05/08 to 05/15
Dashboard
Containers
Search
Add Container
Locations
Barcodes
Reports
CAS Registry
Total Containers
19
(VIEW ALL)
Low Containers
1
(VIEW ALL)
Missing Barcodes
15
(VIEW ALL)
Unassigned Containers
6
(VIEW ALL)
30-Day Recent Inventory
Count By Day-of-Month
Low Containers
Container
Barcode
Amount (%)
Demo Sugar 1
22409898
18
Recent Containers
Container
Barcode
Amount (%)
Cashew nutshell liquid
Delta-8-Tetrahydrocannabinol
Noradrenaline
caffeine
Demo Formaldehyde 3
methylene chloride
diethyl ether
diethyl ether
theobromine
Top Chemicals
CAS Number
Count
Containers
Amount (%)
50-00-0
4
Demo Formaldehyde 1
Demo Formaldehyde 2
Test-rosa
Demo Formaldehyde 3
72
100
100
100
64-17-5
2
Demo Ethanol 2
Demo Ethanol 1
33
Screenshot


Chem
admin
Container Name
user1.chem
USER
Navigation
Containers
Dashboard
Containers
Search
Add Container
Locations
Barcodes
Reports
CAS Registry
Search
Search for containers based on given ide
Name:
Demo
CAS Number:
Barcode:
Search
Weekly Inventory Report
From 5/8/22 to 5/15/22
11 containers added
Name
CAS Number
Barcode
Demo Benzene 1
71-43-2
methylamine
74-89-5
theobromine
83-67-0
diethyl ether
60-29-7
diethyl ether
60-29-7
methylene chloride
75-09-2
Demo Formaldehyde 3
50-00-0
caffeine
58-08-2
Noradrenaline
51-41-2
Delta-8-Tetrahydrocannabinol
5957-75-5
Cashew nutshell liquid
8007-24-7
2 containers updated
Close
Print
Excel
PDF
Filter...
Amount (%)
Actions
Previous
1
Next
Screenshot



The system also has a responsive user interface that will adjust accordingly to the device screen size.