Project 5

Chemical Inventory Management System

Backend FURPS+ Requirements Document

Prepared for:

Dorsey Weber, STEM Laboratory Manager and Safety Compliance Officer

Date:

April 2025

Institution:

Mass Bay Community College

Professor:

Shamsi Moussavi

Lead Developer:

Jordan A.

Lead Backend Developer:

Abraham A.

Backend Developers:

Baheeja M, Valerie H, David H, Bradley J, Anthony H, Andrew L, Gabe L

Table of Contents

1. Introduction

a. Purpose

i. The backend component of the Chemical Inventory Management System (CIMS) is engineered to deliver a secure, efficient, and scalable infrastructure for managing laboratory chemical inventories. It supports the entry and management of chemical records, automated SDS retrieval, hazard classification, and user access control, ensuring streamlined operations and compliance with safety regulations.

b. Scope

i. The backend system for the Chemical Inventory Management System (CIMS) will serve as the foundational infrastructure for managing laboratory chemical records at Mass Bay Community College. This includes handling user authentication, secure data storage, automated barcode scanning and entry, SDS document management, hazard information retrieval, and real-time inventory tracking. The scope encompasses database design, API development, role-based access control, integration with SDS data sources, system monitoring, and support for compliance with institutional and OSHA safety standards. The backend will interface with external services (e.g., barcode scanners, SDS APIs) and provide RESTful endpoints for use by a future frontend system. The backend does not cover UI design, client-side applications, or physical inventory equipment, which are considered out of scope.

2. Functionality Requirements

a. Authentication and User Management

i. Secure login system with username and password.

ii. Role-based access with privilege levels:

1. Admin:

a. Create, update, lock, or delete user accounts. Add/edit/remove chemical records.
b. Track inventory updates.
c. Create new users

2. User:
    a. View, search, and filter inventory.
    b. Download SDS sheets.

b. Inventory Management

i. Support for creating, retrieving, updating, and deleting chemical records.
ii. Data fields include:
    1. Chemical name, CAS number, manufacturer/vendor, bottle size, quantity, location, date acquired.
    2. Hazard classification (GHS/NFPA), PPE recommendations, shelf-life (manufacture date, expiration date, days-until-expiration), amount remaining.
iii. Automatic low-inventory notification system based on customizable threshold values.

c. Barcode Integration

i. Support for barcode scanner and camera detection.
ii. Auto-population of chemical fields including name, manufacturer, CAS number, and bottle size.

d. SDS Management

i. Auto-fetch SDS documents using manufacturer/vendor data (web scraping or API).
ii. Fallback to manual upload of SDS documents.
iii. SDS files stored securely and linked to their respective chemical entries.
iv. Enable individual, batch, or full inventory SDS download.

e. Sorting, Filtering, and Searching
    i. Filter chemicals by keyword, field attributes, and hazard data.
    ii. Sortable by name, location, CAS number, quantity, acquisition date, expiration date, etc.
    iii. Export inventory lists and SDS data for compliance reporting and offline reference.

f. Export & Reporting
    i. Download/export inventory data and SDS documentation.
    ii. System notifications when chemical levels fall below threshold.

3. Usability Requirements

a. RESTful API
    i. consistent naming conventions and structured endpoints.
b. Errors
    i. JSON-based request/response formats with clear validation messages and error handling.
c. Comprehensive API documentation with Swagger/OpenAPI.
d. Users
    i. Users should be able to log in
    ii. Users should be able to sort chemicals by keywords and attributes
    iii. Users should be able to search chemicals by keywords or attributes.
    iv. Users should be able to download one, multiple, or all SDS sheets.
    v. Users should be able to scan or manually enter the chemical barcode to enter it into the database.
    vi. Users should be able to view all chemicals in the database.
    vii. Show shelf life info of the chemicals to the program (i.e, manufacturing date, days-until-expiration, expiration date).

viii. It should also show the hazard level of the chemical and recommend PPE (personal protective equipment accordingly.
ix. show the quantity of said chemical in storage and notify an admin when it is below a certain percentage.
x. User feedback on invalid logins, failed barcode scans, or SDS retrieval issues.

e. Logged actions for traceability, including chemical updates and SDS accesses.

4. Reliability Requirements

a. Robust input validation and sanitation for all user input and data fields.
b. Automated backups for chemical and SDS data at scheduled intervals.
c. SDS fetch failures logged and flagged for admin review.
d. Continuous availability of core services without crashes, lags, or data loss.

5. Performance Requirements

a. Optimized query performance for large inventories and concurrent user access.
    i. Indexed database queries for sorting, searching, and filtering.
    ii. Real-time barcode scanning and lookup under load.
b. Caching of high-frequency data (e.g., SDS links, chemical lists).

6. Supportability Requirements

a. Modular service design following SOLID principles and clean architecture.
b. Environment-based configuration to adapt to different lab deployments.
c. The system can support manual entry.
d. The system can display information about hazards and data.
e. The system can support user login.
f. Inline documentation, detailed developer notes, and a versioned API reference.
g. Admin dashboard for user management, inventory oversight, and hazard alerts.

7. Plus Features

a. Integration Requirements
    i. Support barcode scanner and camera-based scanning as input mechanisms.
    ii. Connect to external SDS data sources through structured APIs or parsing engines.

b. Data Management & Security
    i. Store structured chemical data in a relational database (e.g., PostgreSQL or MySQL).
    ii. SDS documents stored in encrypted, access-controlled cloud storage (e.g., AWS S3).
    iii. Enforce HTTPS for all backend services and endpoints.
    iv. Store passwords using strong hashing (e.g., bcrypt), implement token-based auth (e.g., JWT).
    v. Enable full audit logging and access tracking for compliance.

c. Compliance & Safety Protocols
    i. Align with OSHA hazard communication standards and institutional lab policies.
    ii. Provide detailed hazard metadata and recommend PPE based on chemical classification.

d. Deployment Architecture
    i. Backend delivered via Docker containers with automated CI/CD pipeline
    ii. Logging and alerting systems for SDS failures, login attempts, and threshold breaches
    iii. Designed to scale for growing chemical databases and lab environments