Web Application for Voiland Food Pantry

Project Solution Approach

Maynard Siev

11-FA25-SP26-F-WEB

Jodie Butterworth

Kaitlyn Cornish

Matthew Hill

Alex Langland

10/XX/25

TABLE OF CONTENTS

I. Introduction

This document details our team's solution approach for the Voiland Food Pantry's web application. While considering the goals and intended outcomes of this project, we will outline the application's overall software architecture design, data design for database storage and data manipulation, and graphical user interface design for user interactions with the application. Project developers, technical stakeholders, and the project's clients are the intended audience for this document as a way to view the team's development process and chosen methods of approach.

Food insecurity for college students is a rising problem, which directly affects students' overall well-being and academic performance. Many students face financial stress that may impact their ability to afford necessities such as food. The Voiland College of Engineering and Architecture (VCEA) works to lift this burden from the shoulders of students, faculty, and staff with a food pantry that supplies non-perishable food items. Volunteers help to ensure pantry operations run smoothly while also providing them with volunteer hours.

This food pantry operates using a paper sign-in sheet for clientele to provide information for the purposes of funding. Relying solely on manual processes can be time-consuming, inefficient, and difficult to properly manage. This project seeks to make this process easier for food pantry customers, administrators, and volunteers by integrating a Cougar Card swiping system with a custom domain website created using WordPress. Our goal is to create a sign-in method that is easy to use, trustful, and preferable for all parties and handles clientele information, tracks inventory data as well as volunteer hour tracking.

II. System Overview

This project consists of three major functions for the Voiland Food Pantry: customer and volunteer sign-in, food inventory management, and volunteer hour tracking. This project's primary goal is to replace the Voiland Food Pantry's current state of operation, which functions using a paper sign-in sheet to collect basic clientele information for funding. Using hardware integration with a client-provided magstripe reader and barcode scanner, as well as WSU Web Communication provided WordPress access, we will implement a streamlined system of operation for the Voiland Food Pantry. This will give food pantry staff and volunteers a better, more organized way to store, view, and manage food inventory, pantry visit analytics, and volunteer hours.

As such, Data Design and User Interface Design will be our main areas of interest for the development of this application. This project will require the use of an RDBMS to manage the database tables and their relations with each other, which will, in turn, relate heavily to the User Interface Design. We want this application to be the preferred method of interaction with the Voiland Food Pantry for every expected type of user. This means that our design considerations will focus on usability, performance, extensibility, security, and reliability.

III. Architecture Design

III.1. Overview

Our web application follows a three-tier MVC architecture implemented using Spring Boot for the backend, Thymeleaf for the server-side rendering of HTML views, and MySQL as the relational database layer. This pattern clearly separates responsibilities between presentation, business logic, and data persistence, enabling maintainability and scalability as the system grows.

This architecture was chosen because it naturally supports our three primary stakeholder flows, while allowing future modular expansions. The MVC pattern also aligns well with Spring Boot conventions and Thymeleaf templating, helping ensure rapid development with clean separation of concerns.

Layer / Component Summary:

- **Presentation Layer (View)**
  HTML pages using Thymeleaf (e.g., home.html, login.html, inventory.html) served dynamically with injected data. Responsible for displaying UI to users.

- **Controller Layer**
  Spring Boot controllers (e.g., HomeController, InventoryController) handle HTTP routing (`/login`, `/inventory`, etc.), validate input, and serve the appropriate Thymeleaf views.

- **Service Layer (Business Logic)**
  Where data processing, role validation, visit logging, and inventory rules will live. Currently minimal but planned to expand with inventory logic and real WSU Cougar Card verification.

- **Repository / Persistence Layer**
  JPA repositories (e.g., StudentRepository, InventoryRepository) connecting to MySQL to store customer visits, inventory records, and volunteer hours.

  *Component diagram placeholder

III.2. Subsystem Decomposition

The Voiland Food Pantry web application has been decomposed into three primary subsystems to promote clear responsibilities, maintain high cohesion, and minimize interdependence (coupling) between subsystems. Each subsystem corresponds to a manageable amount of work for a single developer, facilitating parallel development and easier maintenance. The decomposition also allows for future expansion, such as integrating card readers, barcode scanners, or enhanced authentication systems.

The proposed decomposition and rationale are as follows:

- **High cohesion** is maintained by grouping related functions within each subsystem. For example, all inventory-related operations are located in the Inventory Management Subsystem.
- **Low coupling** is maintained by clearly defining interfaces between subsystems, ensuring that each subsystem communicates only what is necessary with others. The Data Persistence Subsystem provides storage services to multiple subsystems without containing business logic.

I.1.1. Sign-In System

a) Description

The Sign-In System handles capture and recording of pantry visits by customers and volunteers. Its responsibilities are presenting the sign-in UI, validating and normalizing user input, creating sign-in records, and forwarding sign-in events to persistent storage. The subsystem is implemented as a Presentation → Controller → Service sequence: Thymeleaf views present the form, controller endpoints receive requests, and a SignInService enforces business rules before persisting via JPA repositories. The design deliberately supports extension to Cougar Card swipe processing and role detection (customer vs volunteer vs admin) in later sprints.

b) Concepts and Algorithms Generated

The Sign-In System was designed around user authentication, data validation, and event logging concepts. Several approaches were considered, including:

1. Manual Input Form Only: Simple HTML forms with minimal backend validation. While straightforward, this approach is prone to human error and lacks scalability.
2. Cougar Card Swipe Integration: Reads the magnetic stripe on student ID cards to automatically identify users. This approach reduces manual errors and accelerates sign-in, but requires hardware support and driver integration.
3. Hybrid Approach (Selected): Supports both manual input and Cougar Card swipes. Users can sign in using either method. Business logic is implemented in the `SignInService`, which validates input, checks for duplicates, and logs events.

The selected hybrid solution balances usability, accuracy, and future extensibility. Trade-offs considered include potential delays for card reader integration versus the simplicity of manual entry. Security considerations were included, ensuring that personally identifiable information (PII) is validated and safely persisted.

c) Interface Description

The Sign-In System exposes services that allow other subsystems to access and process sign-in events. These interfaces ensure that user check-ins are properly recorded and made available to the Volunteer Hour Tracking System and reporting modules.

Services Provided:

1. Service name: RecordSignInEvent
   Service provided to: Volunteer Hour Tracking, Reporting modules
   Description: Receives sign-in input (student ID, volunteer ID, role, timestamp) and validates it. Creates a sign-in record in the database and returns a confirmation or error message. Provides duplicate detection and role classification.

Services Required:
Database Persistence Service – provided by Data Persistence Subsystem / JPA Repositories
User Verification Service – provided by backend UserService for validating IDs against stored user data

I.1.2. Inventory Management System

Include the following sub-sections for each subsystem.

a) Description

The Inventory Management System maintains canonical records of pantry items and stock levels. Responsibilities include CRUD for inventory items, quantity adjustments (stock in/out), search and filtering, low-stock threshold checks, and generation of inventory reports. The system exposes controller endpoints for the UI, encapsulates business rules in an InventoryService, and persists data with JPA repositories to MySQL. The design supports barcode scanner integration and automated item lookup (via UPC) in future iterations.

b) Concepts and Algorithms Generated

The Inventory Management System focuses on maintaining accurate records of pantry items and stock levels. Key considerations included:

1. Simple CRUD Interface: Basic creation, reading, updating, and deletion of inventory items through the web UI. This is simple but does not support advanced features like automated stock alerts.
2. Threshold-Based Stock Alerts: Implements a low-stock alert algorithm that monitors item quantities and notifies administrators when items fall below predefined thresholds.
3. Barcode Scanner Integration: Planned to enable fast inventory updates using UPC scans. Initially, item searches can be done via text entry, with UPC support layered later.

The final approach integrates all three methods. Manual entry ensures flexibility for ad-hoc adjustments, barcode scanning improves efficiency for frequent inventory updates, and the alert system ensures items are restocked on time. Trade-offs considered include ensuring transactional integrity when multiple updates occur simultaneously and making the UI intuitive for both manual and automated interactions.

c) Interface Description

The Inventory Management System interfaces provide access to inventory data and allow other subsystems to add, update, remove, or query pantry items. These interfaces support both

automated updates via barcode scanning and manual inventory management, as well as low-stock alert notifications.

Services Provided:
1. Service name: ManageInventoryItem
   Service provided to: Sign-In System (for reporting item use), Volunteer Hour Tracking System (for inventory-affecting volunteer activity)
   Description: Handles creation, updating, deletion, and retrieval of inventory items. Accepts item ID, name, quantity, category, and optionally UPC. Returns confirmation of actions and current stock levels. Supports low-stock threshold checks.

Services Required:
Database Persistence Service – provided by Data Persistence Subsystem / JPA Repositories
Sign-In Event Service – provided by Sign-In System for associating inventory updates with visits

I.1.3. Volunteer Hour Tracking System

a) Description

The Volunteer Hour Tracking System captures volunteer sign-in/out events, computes hours worked, and provides administrative reporting and approval workflows. Responsibilities include recording volunteer sessions, aggregating hours per volunteer, allowing admin edits/approvals, and exposing summaries for reports. The subsystem interfaces with the Sign-In System for initial check-in events and with the Inventory System when volunteer activity affects inventory.

b) Concepts and Algorithms Generated

Volunteer Hour Tracking System manages volunteer work session recording, aggregation, and reporting. Considered solutions include:

1. Manual Timesheet Entry: Volunteers or admins manually log start/end times. This approach is prone to errors and requires administrative overhead.
2. Automated Sign-In/Sign-Out Logging (Selected): Uses integration with the Sign-In System to automatically capture check-in and check-out times. Aggregates total hours per volunteer using the `VolunteerHourService`.
3. Approval Workflow (Selected): Admins can review, edit, and approve volunteer hours to ensure accuracy before reports are generated.

By combining these last two approaches, the system achieves both efficiency and reliability. Automated logging minimizes human error, while the approval workflow allows administrators to correct discrepancies, approve volunteer hours, and generate accurate reporting. Trade-offs considered include designing an intuitive admin interface for approvals and ensuring proper handling of edits without overwriting automated logs.

c) Interface Description

The Volunteer Hour Tracking System exposes interfaces for tracking volunteer sessions, computing total hours, and supporting admin approval workflows. These interfaces depend on sign-in events to automatically log hours and may interact with inventory updates for volunteer activities.

Services Provided:
1. Service name: AggregateVolunteerHours
   Service provided to: Reporting modules, Admin Dashboard
   Description: Retrieves and computes total volunteer hours per individual over a specified period. Accepts volunteer ID and date range as input, returns total hours, session breakdown, and approval status. Supports admin edits and approval workflows.

Services Required:
Sign-In Event Service – provided by Sign-In System for capturing check-in/out timestamp
Inventory Management Service – provided by Inventory System when volunteer activity affects inventory
Database Persistence Service – provided by Data Persistence Subsystem / JPA Repositories

I.1.4. Data Persistence System

a) Description

The Data Persistence Subsystem provides all storage and retrieval operations for the Voiland Food Pantry web application. It manages access to the MySQL database through JPA repositories, ensuring data integrity, transactional consistency, and secure storage of customer visits, inventory items, and volunteer hours. This subsystem contains no business logic, allowing other subsystems to focus purely on their domain responsibilities.

b) Concepts and Algorithms Generated

The persistence layer is designed using the repository pattern with Spring Data JPA. Key concepts include:

- **Entity Mapping:** Mapping database tables to Java entity classes (Student, InventoryItem, VolunteerSession).
- **CRUD Operations:** Standard create, read, update, delete methods for each entity.
- **Transactions:** Ensuring consistency when multiple operations occur simultaneously.
- **Query Methods:** Custom JPA queries for filtering, reporting, or retrieving aggregate data (e.g., total volunteer hours, low-stock items).

c) Interface Description

The Data Persistence Subsystem exposes standard repository interfaces to all business logic subsystems. These interfaces abstract database operations and allow subsystems to request data without knowing implementation details.

Services Provided:
1. Service name: CRUD Operations
   Service provided to: Sign-In System, Inventory Management System, Volunteer Hour Tracking System
   Description: Provides methods for adding, retrieving, updating, and deleting entities. Accepts entity objects or identifiers and returns confirmations or queried data.

Services Required:
None; this subsystem is the lowest layer and does not depend on other subsystems.

IV. Data Design

This section describes the chosen methods of data manipulation and storage for this project. This application for the Voiland Food Pantry will require persistent storage for user sign-in information, food inventory stock, and volunteer tracking. The data design and database structure will be primarily handled by another team we are working in tandem with. Database storage will use MySQL to manage these relational database tables, which will allow staff members with the proper clearance to view, edit, maintain, and print visit registrations, food inventory stock, and volunteer hour data as needed.

The database team has provided us with an ER Diagram (Figure IV.I found in the appendix) as a visual representation of the included entities, attributes, and relationships between database items. Primary entities include different levels of users (i.e., student, volunteer, administrator), as well as tracked information such as pantry usage, food pantry items, and volunteer hours worked.

Because of the data-sensitive nature of this project (e.g., student ID, usernames and passwords), data will be encrypted using one of MySQL's provided data encryption features. This will ensure the security and privacy of all levels of users. Using RBAC will further guarantee data security, giving each user level a set of permissions pertaining to their roles. For example, administrators will have full access to each table in the relational database, where customers will not.

Because the database team is operating separately in a different course, our development team will need to adapt along with the updates provided to us. We are unsure if the database team is even working on the same documents as us, and as such, we will update our documentation accordingly as this becomes clearer with time. This section provides a basic outline of the Data Design based on what has been communicated with our team at the time of this document's creation.

V. User Interface Design

Our current model uses an HTML front with templates to connect to the Spring Boot Java backend. The website features need to conform to the standards of the WSU website and branding.

- The system will collect data from barcode scans, card swipes and user interface entries.
- The file editing will be accessed through the website and will be available on the terminal in the Food Pantry and on a website.
- The database is accessed through the website interface after logging in with the proper permissions.

The Homepage will include links to the Volunteer Form, Login and Card Reader pages. (See Appendix for images of the Homepage, Login and Card Reader UI). The Homepage will be the first page seen when accessing via the internet. The terminal in the Food Pantry will usually display the Card Reader page. The Toggles on the Card Reader page change whether the swiped card data will be used to register a volunteer shift or a customer. The Login will determine what the user can access. Additional features are available to volunteers so that they can see, but not edit, their hours. Admin login gives access to the database and website features for editing. The Barcode scanning will only be accessible to staff that are logged in.

The Homepage features are accessible to all users, and the card swipe is a very easy way to register. Some paperwork and possible training on barcode scanners are necessary for volunteers. Database editing and printing will be done through an interface and only allowed for administrators.

Figures V.I – V.IV show basic user interface elements including the homepage, customer login, card reader input, and the pantry inventory database. These UI mockups were initially created with the assistance of generative AI and further refined by our development team as an early prototype for UI design. As this project's development evolves, we will continue to improve these bases according to our client's feedback.

VI. Constraints

Having described the team's approach for architecture, data, and user interface design, we must also acknowledge the constraints we will be working around. This project's design has been influenced by certain real-world limitations; however, we have noted these considerations when making decisions for this project:

- **Budget Constraint:**
  This project includes a $0 budget. Fortunately, our team has been provided with the proper resources to make the development process happen. Access to WSU's WordPress website and hardware necessary to create a functional product has given us a way around this budget constraint.
- **Time Constraint:**
  Because this project is associated with coursework, our four-member development team and two-member database team have adopted an Agile development style with 4-week work periods. This allows our teams to balance coursework from other courses and focus on implementing higher-priority features to create functional MVPs for progress updates.

- **Accessibility Constraint:**
  Usability and accessibility is highly emphasized in this project. Our clients wish for this application to be the preferable system for all Voiland Food Pantry users involved. As such, the design of this project focuses on ease of use for the average user who has interacted with any sort of WSU-implemented websites and facilities. Aesthetic decisions for the UI will be made with WSU brand standards in mind while ensuring they comply with Trademark Licensing regulations.

These project constraints provided real-world limitations for our team to apply cooperative problem-solving skills.

VII. Glossary

ER Diagram: Entity-Relationship diagram. A model that visualizes the structure of a database through defined entity relationships.

MVC: Model-View-Controller. A software architecture pattern that divides logic into three: the model, which handles internal information representation; the view, which renders model information in a user-friendly format; and the controller, which links the prior two to provide user input.

MVP: Minimum Viable Product. An early version of a product that has enough features to be properly functional and serves as a means of gathering user feedback.

RBAC: Role-Based Access Control. A mechanism for access control centered around roles and privileges, giving different roles a different set of access permissions.

RDBMS: Relational database management system. A database management system that stores data in an organized row-and-column format and allows related data to interact.

WSU: Washington State University.

VIII. References

This document does not use outside references at the time of its creation. This may change in future revisions and will be updated accordingly to reflect such changes.

IX. Appendices

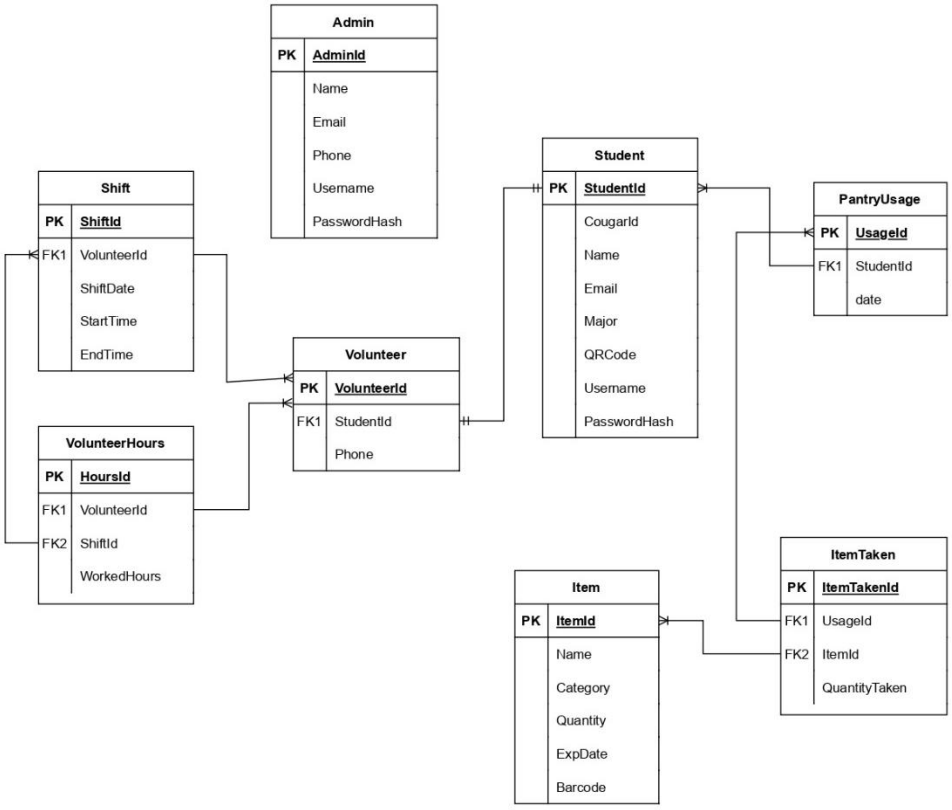

Figure IV.I – Database ER Diagram

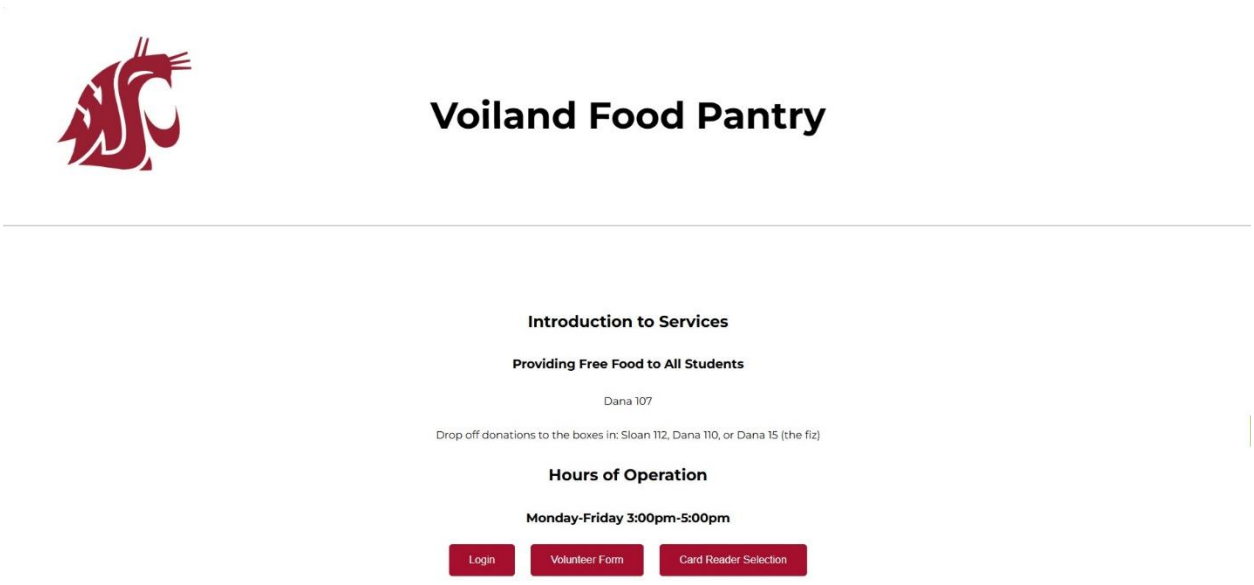

Figure V.I – Homepage UI

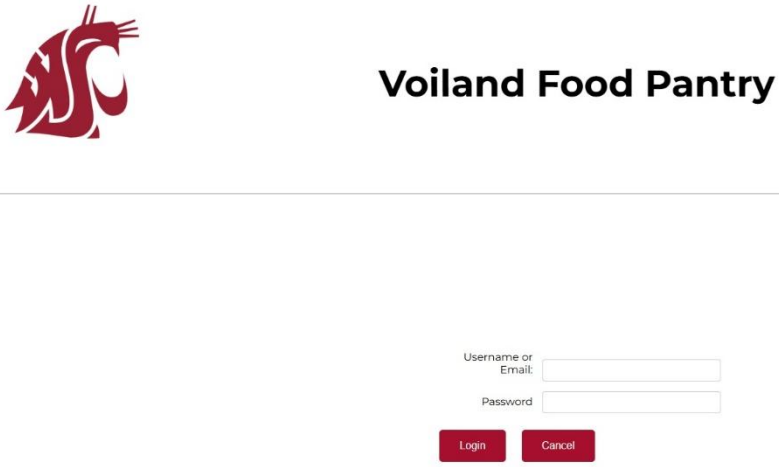

Figure V.II – Login UI

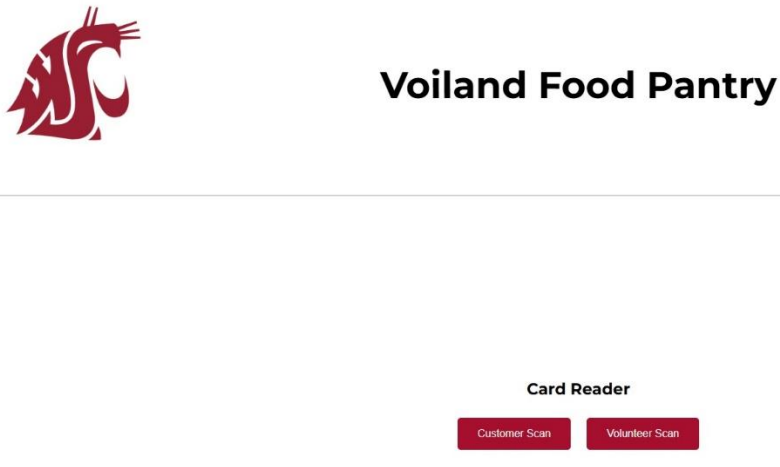

Figure V.III – Card Reader UI

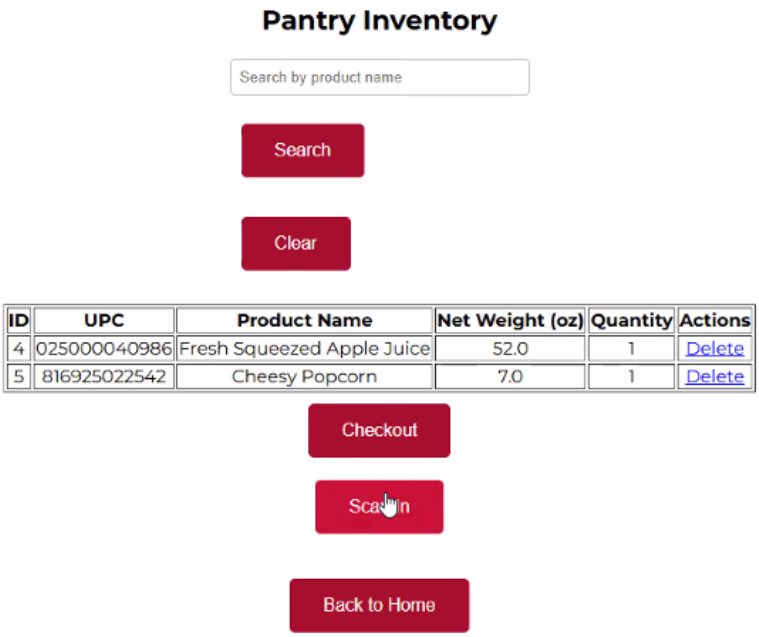

Figure V.IV – Pantry Inventory UI