Web Application for the Voiland Food Pantry

Project Alpha Prototype Report

FIZ, VCSS

11-FA25-SP26-F-WEB

Jodie Butterworth

Kaitlyn Cornish

Matthew Hill

Alex Langland

December 7, 2025

Table of Contents

I. Introduction

I.1. Project Introduction

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. According to the GAO report on student food insecurity, ". . . an estimated 23 percent of students (3.8 million) reported experiencing food insecurity. [1]" 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.

I.2. Background and Related Work

This project's domain can be described according to the three major system functionalities we are focusing on. As such, we broadly define this domain as "an adaptable web application with role-based access for a facility's specific data management needs." This project will be used only by the Voiland Food Pantry, meaning that we will not technically be making a public contribution to this domain, though we will be researching other works that have done so.

One such example is the Cougar Food Pantry [2], which utilizes a magstripe reader with student and staff Cougar Cards for a quick and easy method to record pantry visits. This is the kind of ease of use we would like to be associated with our own visit recording method, which will also require hardware integration with our own magstripe reader provided by our client. We will continue this hardware integration with a barcode scanner to further facilitate both a streamlined experience for pantry clientele, as well as increase efficiency for pantry staff and volunteers to collect data from regular pantry operation.

The front-facing part of our project will be inspired directly by WSU website brand standards [3] to keep this application's user interface sufficiently familiar for students and staff alike. This will not only ensure that our system is intuitive, but it will also serve as a way to enhance trust with pantry clientele. Transitioning from a physical method of recording pantry visits to a webpage sign-in with a card reader may be daunting, so keeping the webpage consistent for a WSU-related facility will assist with this transition greatly. Keeping the application's user interface design professional is also a way to guarantee that pantry clientele will continue to trust that the Voiland Food Pantry is similarly safe as other campus food pantries.

For successful execution of these desired results, our team will require technical knowledge and skills we both already possess and need to improve. Along with this, we will need to gain new expertise with tools and frameworks we have not worked with in the past. Because this project will be made with WordPress and using MySQL, we will need to develop a strong understanding of both of these frameworks. We will also need to explore the hardware given to us by our clients to ensure that we can properly integrate these aspects into our system. More generally, web development skills will be required for this project to create a straightforward web application that is simple but attractive for all users of our eventual end product.

I.3. Project Overview

The Web Application for the Voiland Food Pantry project is designed to modernize how the food pantry operates by moving away from a fully paper-based system and introducing a digital solution that is both efficient and user-friendly. The current system relies on paper sign-in sheets for clientele and manual logging of volunteer hours. While this approach works on a basic level, it creates a number of problems: information can be lost or misstated, compiling reports for funding is time-consuming, and there is no easy way to analyze patterns of pantry use or volunteer activity. As a result, the pantry's ability to serve students effectively and report accurate data to stakeholders is limited.

To address these issues, our project focuses on creating a WordPress-based website that will serve as the main point of integration for pantry users and staff. The system will connect to a Cougar Card Reader, allowing students, faculty, and staff to swipe their card upon arrival for quick and accurate sign-in. Volunteers will also use the same site to log their hours, with a dedicated option to distinguish their roles from that of clientele. This approach reduces waiting times and ensures that information is captured in a consistent format. In cases where the card reader is unavailable or a user does not have their card, the website will act as a digital backup, ensuring that no visit or contribution goes unrecorded.

Equally important is the administrative side of the project. Pantry supervisors will be able to access records directly through the system, giving them the ability to manage data, generate reports, and verify contributions without having to sort through stacks of paperwork. These features will not only save time but will also improve accuracy and consistency in reporting, which is essential for maintaining funding and demonstrating the impact of the pantry's services.

The project is being developed in collaboration with another student team responsible for the database infrastructure. While our team focuses on the design and implementation of the website interface, the database team will handle secure storage and organization of the collected data. This partnership ensures that the project addresses both the user-facing and back-end requirements, creating a more complete and robust system. Looking ahead, the client expressed interest in making the application adaptable so that other departments within Washington State University can adopt similar sign-in and tracking tools. By designing the system with flexibility in mind, our work can serve not only the Voiland Food Pantry but also the broader university community, like the Wellness Center, which may serve as a focal point of the future of this project.

Our main objectives for this project are as follows:

- Create a website using WordPress to act as a default food pantry clientele sign-in page for a computer terminal set up in the food pantry.
    - It will include a link to another page specifically for volunteers to sign in, allowing them to track their volunteer hours worked at the VCEA food pantry.
- Clientele and volunteers should be able to sign in by swiping their Cougar Card in an attached card reader.
- If the reader does not work or the customer forgets their card, the website will act as digital sign-in backup.
- Allow food pantry administrators to manage food pantry inventory, as well as information collected from customers who have used the pantry.

    - A paper sign-in sheet will still be provided for use. Volunteers will be available to collect and record this information if customers prefer to use the sheet instead.
- Allow a supervisor to access and print volunteer hour information to sign off and confirm these hours.

I.4. Client and Stakeholder Identification and Preferences

The primary client for this project is Maynard Siev, who serves as the main point of contact for the project's first sponsor, FIZ. Other clients include Alena Hume, Lisa Carmack, and Gary Offerdahl, all of whom are associated with our second sponsor, VCSS. Our faculty mentor, Parteek Kumar, also plays a key role by providing guidance and ensuring that the project aligns with both course requirements and the pantry's operational needs. Together, they represent the direct clients who will oversee the system's development and evaluate its effectiveness.

The most frequent users of the system will be pantry customers. Customers, including students, faculty, and staff, need a sign-in process that is quick, private, and easy to use. For most, swiping a Cougar Card will be the preferred method, though the option of a digital backup form ensures inclusivity for those without a card.

Volunteers rely on the system to log their service hours accurately. Their preference is for a simple and dependable method that allows them to check in quickly while guaranteeing their hours are properly recorded and verified.

Pantry administrators and supervisors form another critical stakeholder group. They are responsible for overseeing client usage and volunteer participation, preparing reports for funding, and managing daily operations. Their primary preference is for a system that reduces paperwork, improves accuracy, and provides easy access to data. Features like report generation, digital recordkeeping, and minimal manual entry are essential for making their work more efficient.

Additional stakeholders include the university IT staff and the student team responsible for the database. The IT staff will support hardware approval and ensure the system meets campus standards, while the database team will handle the back-end infrastructure. Both groups prefer a solution that integrates smoothly with existing resources and requires little long-term maintenance. Looking ahead, the pantry hopes this system can serve as a model for other departments, so adaptability and scalability are important considerations for future stakeholders as well.

II. Team Introductions

II.1. Jodie Butterworth

Jodie is a senior Computer Science major at Washington State University. Her skills include UI/UX mobile and web development, graphic design, and technical writing. She has prior experience with web development and has worked with WordPress, as well as programming languages such as Python, C/C++/C#, HTML, and CSS. For this project, her responsibilities include front-end development and UI design.

II.2. Kaitlyn Cornish

Kaitlyn is a senior Computer Science major at Washington State University. Her skills include back-end web development, project management, technical writing, and database design and management. She has prior experience with web development and cybersecurity, as well as programming languages such as Python, C/C++, Java, and database management systems such as MySQL. For this project, her responsibilities include back-end development and architecture design.

II.3. Matthew Hill

Matthew is a senior Computer Science major at Washington State University. His skills include project management, mobile and web development, DevOps, graphic design, and technical writing. He has prior experience with web development, cybersecurity, and machine learning, as well as programming languages such as Python, C/C++, HTML, Java, and Haskell. For this project, his responsibilities include project management as team lead, acting as liaison between stakeholders, and front-end development.

II.4. Alex Langland

Alex is a senior Computer Science major at Washington State University. His skills include back-end development, database design and management, troubleshooting, and technical writing. He has prior experience with cybersecurity and machine learning, as well as programming languages such as Python and C/C++/C#, and database management systems such as MySQL. For this project, his responsibilities include back-end development, data design, and hardware integration.

III. System Requirements Specification

The System Requirements Specification details the key functional and non-functional requirements for the Web Application for the Voiland Food Pantry. Functional requirements focus on core system capabilities such as card reader integration, barcode scanning, website sign-in, and administrative reporting. Non-functional requirements outline operational qualities, such as performance, security, and usability.

This section also includes use cases and user stories that demonstrate real-world interactions between users and the system. Each use case is linked to the relevant functional requirement, ensuring a cohesive system design that aligns with stakeholder expectations.

III.1. Use Cases

The use cases describe common scenarios of user interactions with the system, explaining how various functional requirements are applied in specific situations. The use cases of the proposed are represented in the Use Case Diagram given in Figure 1.

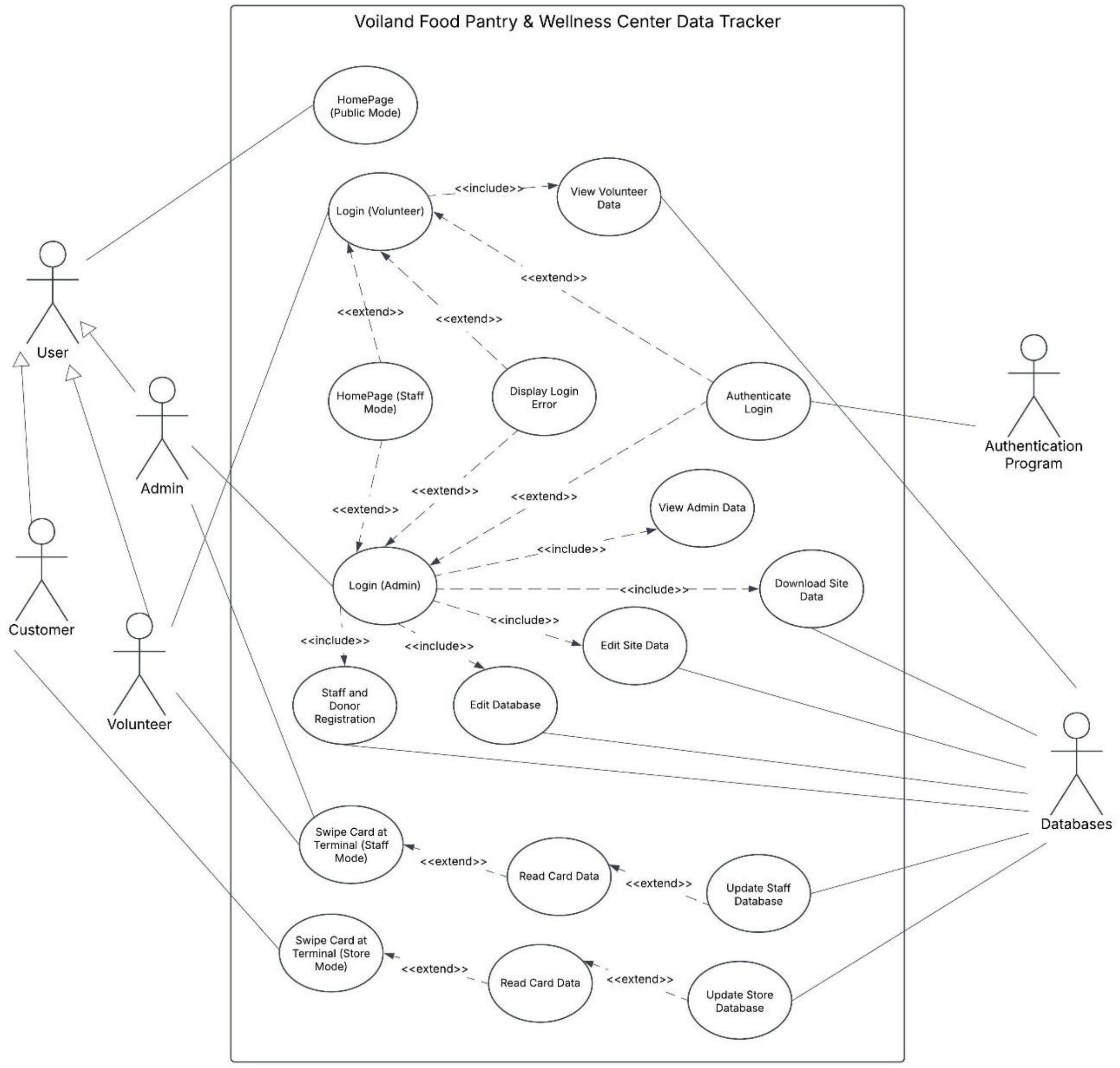


Figure 1: Use Case Diagram

The description of each use case is given in the tables below.

Use Case 1: Customer Sign-In

Actors	Pantry Customer (Student, Faculty, or Staff)
Pre-condition	• The customer is at the pantry. • Card reader and/or website is available and operational. • Customer has a valid Cougar Card.
Post-condition	• Customer visit is recorded in the database. • Volunteer and administrative records are updated.
Main Flow	• Customer approaches the pantry terminal. • Customer swipes Cougar Card at the reader. • System verifies the card and logs the visit. • Confirmation message displayed to customer.
Alternative Flow	**Alternative Flow 1 (Website Backup):**

	• Card reader is unavailable or card is not present. • Customer enters information through the website form. • System records visit in the database. **Alternative Flow 2 (Paper Backup):** • Both card reader and website are unavailable. • Customer signs in on paper form. • Volunteer later enters data into the system manually.
Related Requirements	• [FR-2]: User Sign-in • [FR-3]: Card Swipe Data

Use Case 2: Volunteer Hour Logging

Actors	Pantry Volunteer
Pre-condition	• Volunteer is present at the pantry. • Card reader and/or website are operational. • Volunteer has a valid Cougar Card.
Post-condition	• Volunteer hours are recorded in the database. • Administrators can access accurate volunteer records.
Main Flow	• Volunteer approaches the terminal. • Volunteer swipes Cougar Card at the reader. • System verifies the card and logs volunteer hours. • Confirmation message displayed to volunteer.
Alternative Flow	**Alternative Flow 1 (Website Backup):** • Card readers are unavailable, or card is not present. • Volunteer enters information on the website. • System records volunteer hours in the database. **Alternative Flow 2 (Paper Backup):** • Both card reader and website are unavailable. • Volunteer records hours on paper form. • Administrator enters hours into the database later.
Related Requirements	• [FR-2]: User Sign-In • [FR-3]: Card Swipe Data • [FR-6]: Volunteer Database

Use Case 3: Inventory Update via Barcode Scanning

Actors	Pantry Volunteer / Pantry Staff
Pre-condition	• Volunteer/Staff is present at the pantry. • Barcode scanner is connected and operational. • Item to be stocked or distributed is available.
Post-condition	• Food inventory database is updated with new items or adjusted quantity.
Main Flow	• Volunteer scans food item barcode. • System retrieves item details or prompts entry if item is new. • Inventory count is updated in the database. • Confirmation message is displayed.
Alternative Flow	**Alternative Flow 1 (Manual Entry):** • Barcode scanner is unavailable. • Volunteer manually enters item code and details into system. • Database updates accordingly.

Related Requirements	• [FR-4]: Barcode Scanning • [FR-5]: Inventory Database

Use Case 4: Generate Administrative Reports

Actors	Pantry Administrators / Supervisors
Pre-condition	• Administrator is logged in with valid credentials. • Database contains up-to-date volunteer and/or inventory data.
Post-condition	• Report is generated and available in printable or downloadable format.
Main Flow	• Administrator logs in to the system. • Administrator selects type of report. • System retrieves relevant data. • Report is generated and displayed in chosen format.
Alternative Flow	**Alternative Flow 1 (Manual Entry):** • Administrator chooses to export data. • Report is saved as CSV/PDF for external use.
Related Requirements	• [FR-7]: Administrative Reporting • [FR-8]: Administrative Login

Use Case 5: Low Stock Alert

Actors	Pantry Administrators
Pre-condition	• Inventory items exist in database and set quantity thresholds. • System is actively monitoring inventory updates.
Post-condition	• Administrator is alerted about low stock items via dashboard notification or email.
Main Flow	• Volunteer scans item and inventory is updated. • System checks stock levels against thresholds. • If stock is below threshold, alert is triggered. • Administrator receives dashboard notification and/or email.
Alternative Flow	**Alternative Flow 1 (Threshold Adjustment):** • Administrator adjusts stock threshold. • System recalibrates alerts accordingly.
Related Requirements	• [FR-5]: Inventory Database • [FR-10]: Low Stocks Alert

Use Case 6: Administrative Login and Database Management

Actors	Pantry Administrators
Pre-condition	• Administrator has valid login. • System is operational and connected to database.
Post-condition	• Administrator is logged in with elevated access rights. • Database can be updated or managed as needed.
Main Flow	• Administrator navigates to login page. • Administrator enters credentials. • System verifies credentials and grants access. • Administrator registers new user logins/ edits volunteer/inventory data/ downloads data.
Alternative Flow	**Alternative Flow 1 (Failed Login):**

	• Administrator enters incorrect credentials. • System denies access and prompts reentry.
Related Requirements	• [FR-1]: User Registration • [FR-5]: Inventory Database • [FR-6]: Volunteer Database • [FR-8]: Administrative Login

III.2. Functional Requirements

Each functional requirement is listed below with a detailed description, source, and priority level.

III.2.1. User Management

[FR-1]: User Registration

Description	The system shall allow admin to register users to login using valid credentials.
Source	Requirements elicitation from the client.
Priority	Level 0 (Essential)

[FR-2]: User Sign-In

Description	The system shall allow users to sign in via the website or card reader, distinguishing between customer and volunteer roles.
Source	Requirements elicitation from the client.
Priority	Level 0 (Essential)

III.2.2. Inventory and Volunteer Management

[FR-3]: Card Swipe Data

Description	The system shall read user card data into the database upon swipe.
Source	Requirements elicitation from the client.
Priority	Level 0 (Essential)

[FR-4]: Barcode Scanning

Description	The system shall scan food item barcodes into the database and update inventory status.
Source	Client request
Priority	Level 0 (Essential)

[FR-5]: Inventory Database

Description	The system shall maintain a database of all pantry inventory items.
Source	Requirements elicitation from the client.
Priority	Level 0 (Essential)

[FR-6]: Volunteer Database

Description	The system shall maintain a database of volunteer information and hours worked.
Source	Requirements elicitation from the client.
Priority	Level 0 (Essential)

[FR-7]: Administrative Reporting

Description	The system shall allow supervisors to generate and print reports for volunteer and inventory data.
Source	Internal requirements elicitation among member of the team.
Priority	Level 1 (Desirable)

[FR-8]: Administrative Login

Description	The system shall allow administrative access to edit and manage databases.
Source	Requirements elicitation from the client.
Priority	Level 0 (Essential)

[FR-9]: WordPress User Interface

Description	The system shall provide a user interface on WordPress that is compatible with the WSU website.
Source	Requirements elicitation from the client and alignment with WSU web standards.
Priority	Level 0 (Essential)

[FR-10]: Low Stocks Alerts

Description	The system shall alert administrators when stock of an item falls below a set threshold via dashboard and email.
Source	Internal requirements elicitation among team members.
Priority	Level 1 (Desirable)

III.3. Non-Functional Requirements

The non-functional requirements outline the system's operational qualities, such as performance, security, and usability, to ensure it meets quality standards beyond core functionality. These qualities are identified and defined in Table 1.

Table 1: Non-Functional Requirements

Non-Functional Requirement	**Description**
[NFR-01] Security	The system shall encrypt and secure all database data.
[NFR-02] Performance	The system shall verify card swipe and barcode reader data within 3 seconds.
[NFR-03] Uptime	The system shall maintain an uptime of at least 99%.
[NFR-04] Accuracy	The barcode reader shall scan items with at least 99% accuracy.
[NFR-05] Usability	The website interface shall follow WSU Web Standards and Style Guide for typography, UI elements, and navigation.

[NFR-06] Learnability	At least 80% of users shall complete tasks without assistance after ≤20 minutes of familiarization.
[NFR-07] Reporting	The system shall generate reports for datasets ≤10,000 records within 5 seconds.

III.4. User Stories and Traceability Matrix

The following user stories describe specific actions a user can take in the system, detailing what the user wants to accomplish and why. Each user story specifies system behaviors clearly. Additional user stories are provided in Appendix A.

User Story US-1: User Sign-In
As a food pantry user, I want to sign in using my WSU card or website so that my visit is registered.
Feature: Card Sign-In
Scenario: Customer uses WSU card

- **Given:** I am in the food pantry
- **When:** I swipe my WSU card at the card reader
- **Then:** My visit is recorded in the database
- **And:** My role (customer or volunteer) is correctly identified

Feature: Website Sign-In Backup
Scenario: Customer uses website

- **Given:** I am in the food pantry
- **When:** The card reader is unavailable, or I do not have my card
- **Then:** I can sign in using the website
- **And:** My visit is recorded in the database

Feature: Paper Sign-In Backup
Scenario: Card reader and website are unavailable

- **Given:** I am in the food pantry
- **When:** Both card reader and website are inoperable
- **Then:** I sign in on paper
- **And:** A volunteer later enters the visit into the database

User Story US-2: Barcode Scanning for Inventory
As a pantry volunteer, I want to scan items so that the inventory database is updated.
Feature: Inventory Barcode Scanning
Scenario: Successful scan

- **Given:** I am a volunteer in the pantry
- **When:** I scan an item with the barcode scanner
- **Then:** The item is added or updated in the inventory database

Feature: Manual Inventory Entry
Scenario: Barcode scanner fails

- **Given:** I am a volunteer in the pantry
- **When:** The barcode scanner fails
- **Then:** I record the item on a paper form
- **And:** I later update the database manually

User Story 3: Administrator Login for Editing
As a pantry administrator, I want to log in to edit volunteer data.

Feature: Admin Login
Scenario: Successful login
- **Given:** I am an administrator
- **When:** I log in with valid credentials
- **Then:** I can access and edit volunteer data
- **And:** I can register new volunteers
- **And:** The changes are saved in the database

Scenario: Login failure
- **Given:** I am an administrator
- **When:** I enter invalid credentials
- **Then:** Login fails
- **And:** I remain on the login page

Scenario: Database unavailable
- **Given:** I am logged in
- **When:** The database is unavailable
- **Then:** I record changes on paper or another program
- **And:** Technical issues are reported

The Traceability Matrix, Table 2, maps functional requirements to their respective use cases and user stories. This ensures that all requirements are accounted for and linked to user scenarios.

Table 2: Traceability Matrix

Functional Requirement	Use Case	User Story	Priority
[FR-1]: User Registration	UC-6: Administrative Login and Database Management	US-3: As a user, I want to register an admin account with the website.	Level 0
[FR-2]: User Sign-In	UC-1: Customer Sign-In UC-2: Volunteer Hour Logging	US-1: As a user, I want to sign in using my Cougar Card or the website.	Level 0
[FR-3]: Card Swipe Data	UC-1: Customer Sign-In UC-2: Volunteer Hour Logging	US-1: As a user, I want to entrust the food pantry with my Cougar Card data.	Level 0
[FR-4]: Barcode Scanning	UC-3: Inventory Update via Barcode Scanning	US-2: As a pantry volunteer, I want to scan items' barcodes.	Level 0
[FR-5]: Inventory Database	UC-3: Inventory Update via Barcode Scanning UC-5: Low Stock Alert UC-6: Administrative Login and Database Management	US-4: As a pantry administrator, I want to access and maintain inventory data. US-8: As the system, I want to notify administrators when stock is low so that inventory can be replenished in time.	Level 0
[FR-6]: Volunteer Database	UC-2: Volunteer Hour Logging UC-4: Generate Administrative Reports	US-3: As a user, I want to register an admin account with the website. US-5: As a pantry administrator, I want to access and maintain volunteer data.	Level 0

	UC-6: Administrative Login and Database Management		
[FR-7]: Administrative Reporting	UC-4: Generate Administrative Reports UC-6: Administrative Login and Database Management	US-4: As a pantry administrator, I want to access and maintain inventory data. US-5: As a pantry administrator, I want to access and maintain volunteer data.	Level 1
[FR-8]: Administrative Login	UC-6: Administrative Login and Database Management	US-3: As a pantry administrator, I want to log in to edit volunteer and inventory data.	Level 0
[FR-9]: WordPress User Interface	UC-1: Customer Sign-In UC-2: Volunteer Hour Logging UC-6: Administrative Login and Database Management	US-1: As a user, I want to sign in using my Cougar Card or the website. US-3: As a pantry administrator, I want to log in to edit volunteer and inventory data. US-6: As a food pantry donator, I want to access the food pantry website so that I can contact supervisors. US-7: As a customer, I want to access the food pantry website so that I can view the hours.	Level 0
[FR-10]: Low Stock Alert	UC-5: Low Stock Alert	US-8: As the system, I want to send an alert to pantry administrators when stock of a particular item is low.	Level 1

III.5. Standards

This project aims to adhere to professional standards relevant to the development, documentation, and design processes throughout the project's creation timeline. We will ensure structured, standardized documentation practices by following **IEEE 830** [4] and **IEEE 1016** [5]. This application must follow **W3C WCAG 2.2** [6] to provide web content accessibility for clientele, volunteers, and administrators of the Voiland Food Pantry. Finally, while personal data will not be collected by this application, it will still follow the **ACM Code of Ethics** [7] to guarantee our team stays mindful of professionalism and fairness during this process.

IV. System Evolution

The Web Application for the Voiland Food Pantry needs to be flexible and scalable to handle growth in both users and inventory. The database will be designed to expand easily as more food items and volunteer records are added, keeping the system accurate and responsive.

The interface will remain fast and easy to use even as the amount of data grows, so customers, volunteers, and administrators can interact with the system smoothly. Admins will be able to manage inventory, track volunteer hours, and generate reports without needing advanced technical skills.

The system will use a modular design, so new features—like automated alerts or additional reports—can be added without disrupting existing functionality. We'll also work with pantry staff to make sure updates and changes fit their needs and are easy to use.

V. Architecture Design

V.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 [8].

- **Controller Layer**
  Spring Boot controllers (e.g., HomeController, InventoryController) handle HTTP routing (`/login`, `/inventory`, etc.), validate input, and serve the appropriate Thymeleaf views [9].

- **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.

Figure 2 below is a System Architecture Diagram showcasing these major components and their interactions with one another.

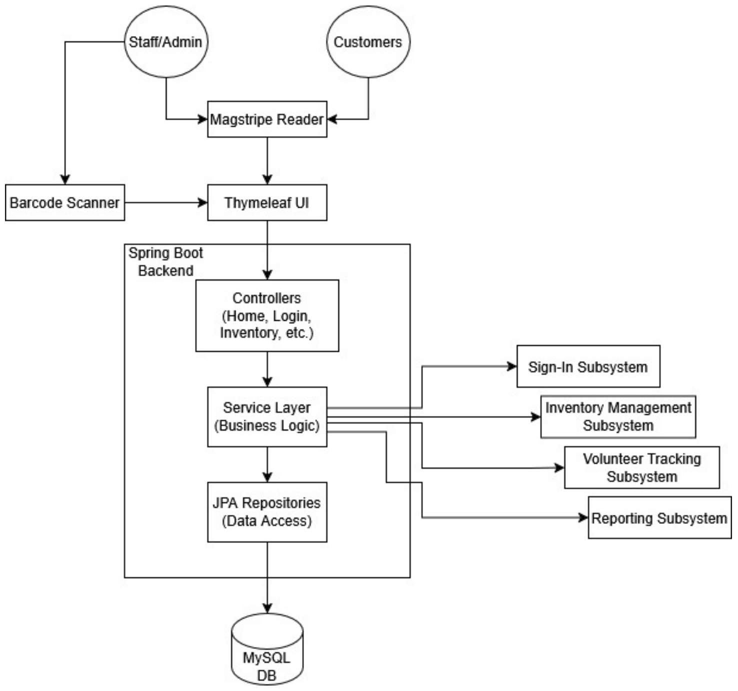

Staff/Admin
Customers
Magstripe Reader
Barcode Scanner
Thymeleaf UI
Spring Boot Backend
Controllers (Home, Login, Inventory, etc.)
Service Layer (Business Logic)
JPA Repositories (Data Access)
MySQL DB
Sign-In Subsystem
Inventory Management Subsystem
Volunteer Tracking Subsystem
Reporting Subsystem

Figure 2: System Architecture Diagram

V.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.

V.2.1. 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:

- Manual Input Form Only: Simple HTML forms with minimal backend validation. While straightforward, this approach is prone to human error and lacks scalability.
- 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.
- 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.

<u>Services Provided:</u>

- 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.

<u>Services Required:</u>

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

V.2.2. Inventory Management System

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:

- 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.
- Threshold-Based Stock Alerts: Implements a low-stock alert algorithm that monitors item quantities and notifies administrators when items fall below predefined thresholds.
- 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:

- 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

V.2.3. 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:

- Manual Timesheet Entry: Volunteers or admins manually log start/end times. This approach is prone to errors and requires administrative overhead.
- 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`.
- 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.

<u>Services Provided:</u>

- 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.

<u>Services Required:</u>

- 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

V.2.4. 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:
- 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.

VI. 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 3) 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.

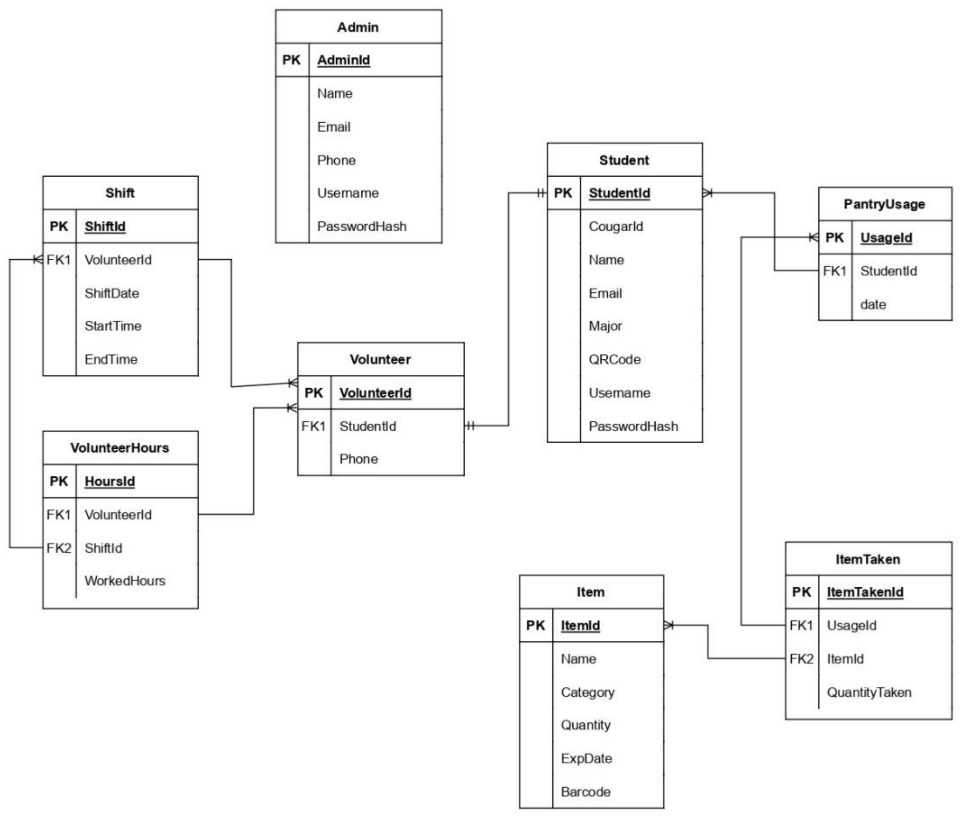


Figure 3: ER Diagram

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.

VII. 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.

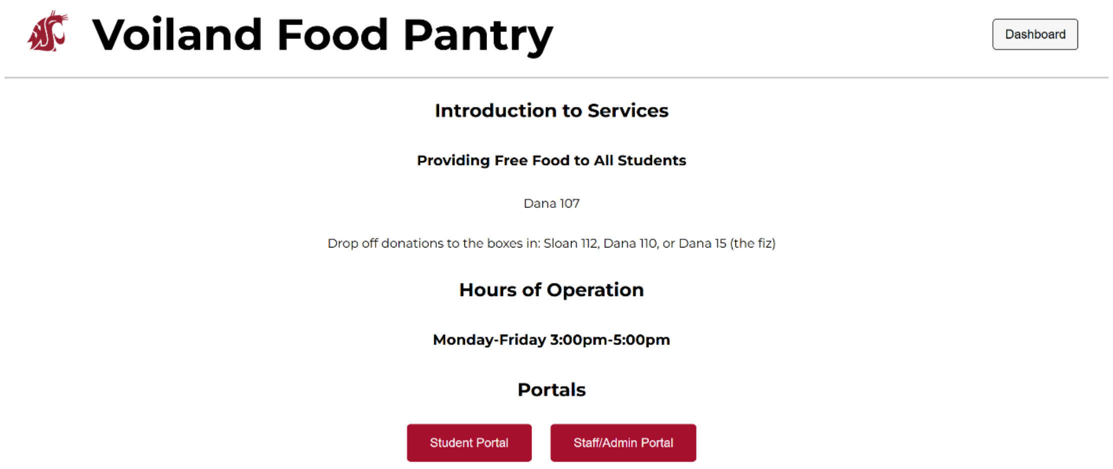
- The system will collect data from barcode scans, card swipes and user interface entries.
- The file editing will be accessed through the website and will be available on the terminal in the Food Pantry and on a website.

- 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 C 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 4 – 9 below show basic user interface elements including the landing page, customer and admin login, admin homepage, 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.


Voiland Food Pantry

Dashboard

Introduction to Services

Providing Free Food to All Students

Dana 107

Drop off donations to the boxes in: Sloan 112, Dana 110, or Dana 15 (the fiz)

Hours of Operation

Monday-Friday 3:00pm-5:00pm

Portals

Figure 4: Landing Page

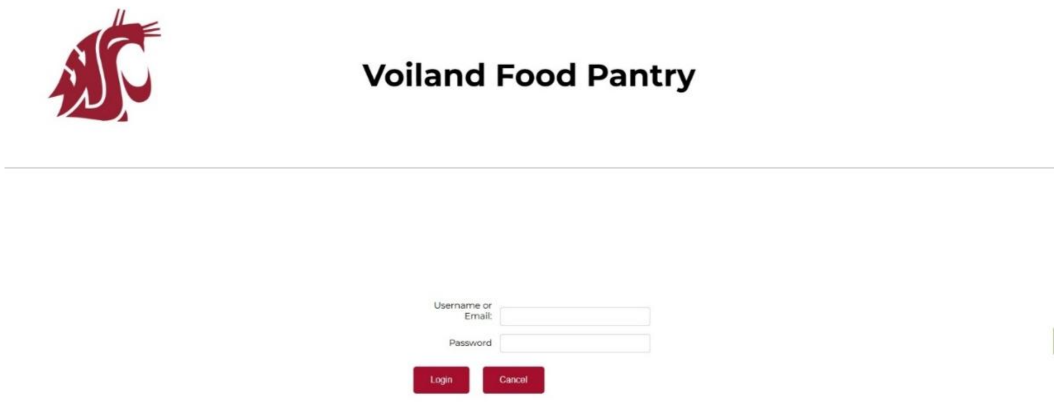

Voiland Food Pantry

Figure 5: Customer Login

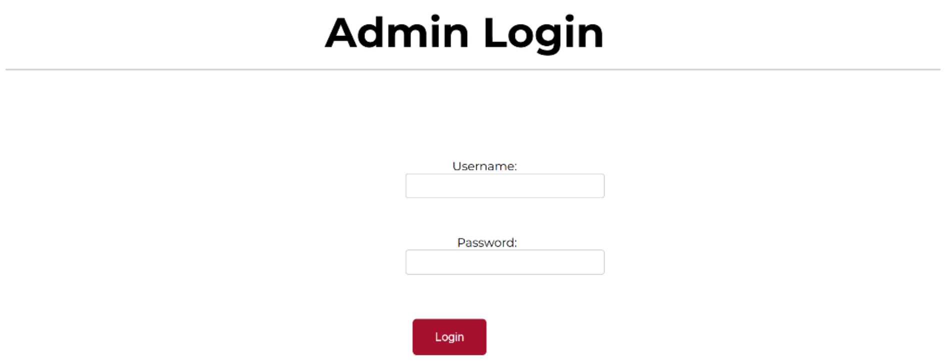
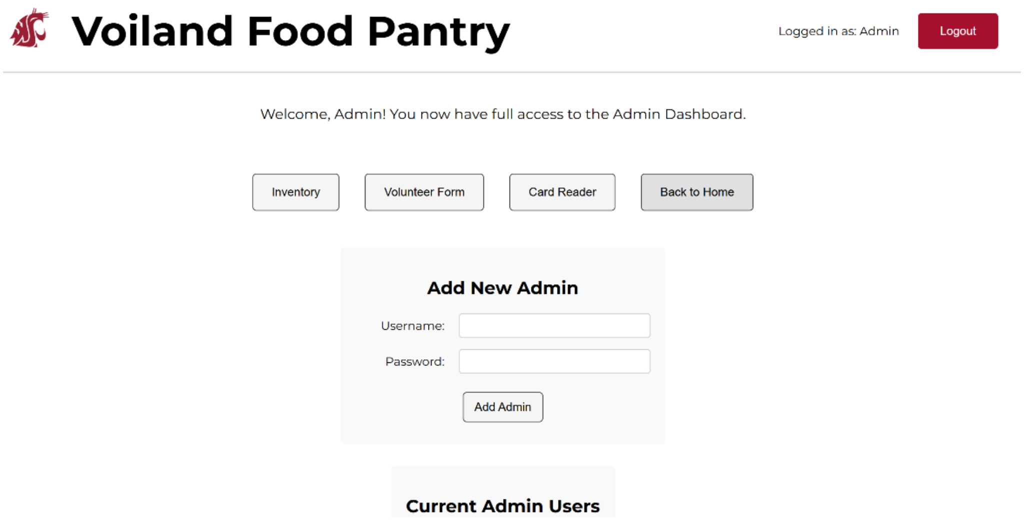

Admin Login

Figure 6: Admin Login


Voiland Food Pantry

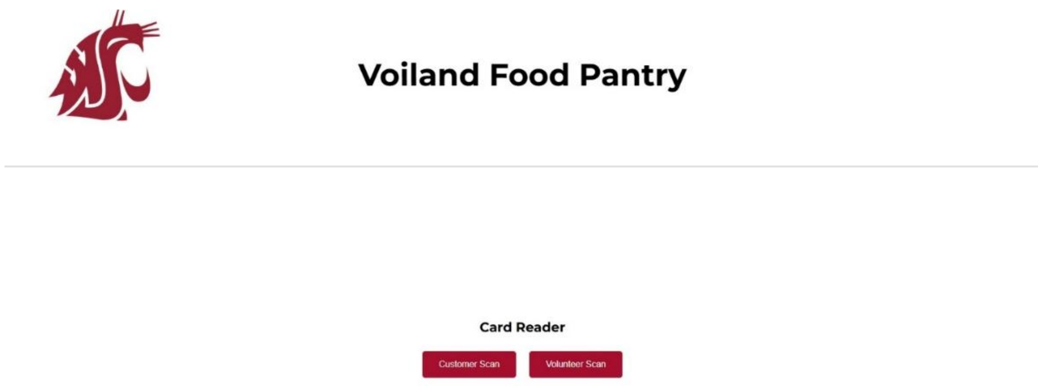
Logged in as: Admin

Welcome, Admin! You now have full access to the Admin Dashboard.

Add New Admin

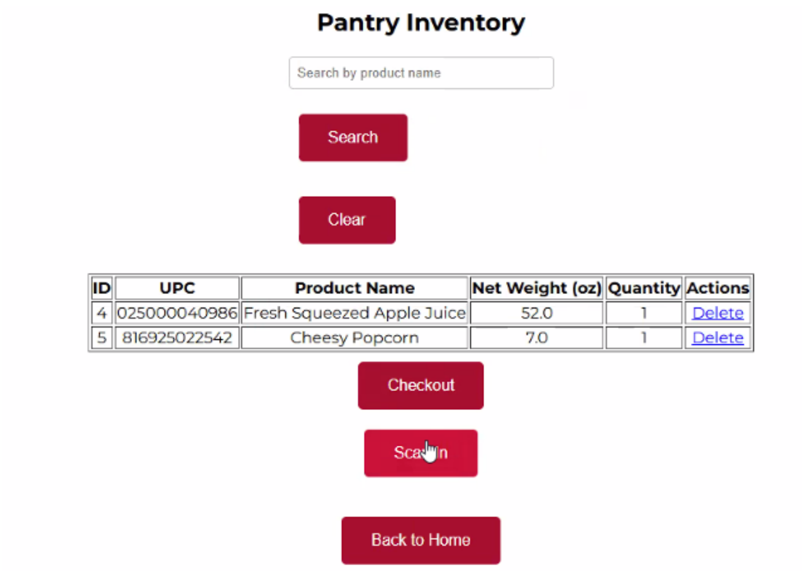
Current Admin Users

Figure 7: Admin Homepage


Voiland Food Pantry

Card Reader

Figure 8: Card Reader Input


Pantry Inventory

ID	UPC	Product Name	Net Weight (oz)	Quantity	Actions
4	025000040986	Fresh Squeezed Apple Juice	52.0	1	Delete
5	816925022542	Cheesy Popcorn	7.0	1	Delete

Figure 9: Pantry Inventory Database

VIII. 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.

IX. Project Testing and Acceptance Plan

IX.1. Overview

The software associated with this project includes a webpage consisting of a landing page, which leads to pages for student and staff/admin login and dashboards, inventory management, and volunteer hour tracking. Testing will be crucial for these major elements to ensure they function properly to keep the Voiland Food Pantry operating smoothly during its business hours.

IX.1.1. Test Objectives and Schedule

The objectives of our testing plan all work towards ensuring this application functions as expected of our stakeholders and future users. For this purpose, we will employ several kinds of testing methods, focusing our attention on integration testing due to the interconnected nature of this application. Manual testing will be utilized, allowing our team to personally test application operation and edge cases to ensure regular users do not encounter unintended errors with the system. This will be important for verifying the project's expected performance and ultimately, the final product we will deliver to our clients.

For this, both hardware and software resources will be required, which will be outlined later in greater detail. These resources will assist our team in our testing approach, which will facilitate our planned schedule of testing activities.

These activities include major work activities such as developing our testing strategy, setting up a testing environment, creating test cases for the system, executing tests, and recording the results of such tests. With this, we plan on delivering a functional system to our clients by the end of this semester so that they may interact with the application and its current features. Along the way, major milestones will be reached, such as approving our testing strategy, completing our test cases with results to highlight any improvements that must be made, and receiving feedback from our clients about said results.

IX.1.2. Scope

The purpose of this section is to outline our team's testing strategy, along with the approach we will take to employ it. This testing strategy will be specifically designed to validate the Voiland Food Pantry web application performance with both the FIZ and VCSS's requirements for the application in mind. Along with this, we will define both hardware and software requirements required for operation, as well as testing criteria that must be met before application deployment. Different types of testing, including unit, integration, functional, performance, and user acceptance, will be performed to guarantee this project's reliability in an active usage situation.

IX.2. Testing Strategy

The overall testing approach for the Voiland Food Pantry web application focuses on manual, exploratory testing of all currently implemented system components immediately after development. Testing is guided by the functional and non-functional requirements defined in the System Requirements Specification. The team aims to ensure that core features—such as user login, inventory management, and device integration—function correctly in realistic usage scenarios.

The testing plan for the Voiland Food Pantry web application focuses on verifying the correctness and robustness of the currently implemented features, as well as establishing procedures for testing future modules once implemented. Testing is conducted manually, following structured steps for each type of testing.

An example testing strategy is provided in Appendix B.

IX.3. Test Approach

Manual testing is performed for each module in isolation first (unit-level checks), then in combination with related modules (integration), and finally across the entire system (system testing). Each test session involves executing typical user workflows to verify that the system behaves as expected and handles invalid inputs or errors gracefully. Key aspects currently tested include:

- **User Login:** Verifying card swipe and website login for customers and administrators, including error handling for unregistered users.
- **Inventory Management:** Adding, updating, and removing items via barcode scanning and manual entry, ensuring accurate inventory counts.
- **Device Integration:** Testing barcode scanner and card reader input to ensure seamless interaction with the web interface.
- **Error Handling:** Ensuring appropriate messages appear for invalid operations, such as unregistered users attempting login or scanning invalid inventory codes.

Modules planned for future implementation next semester, such as volunteer hour logging and administrative reporting, are acknowledged but are not yet included in testing. Once implemented, these features will undergo the same manual testing process to ensure accurate data tracking and report generation.

The manual testing process follows these steps:

- Identify the component or feature to test based on the functional requirements.
- Prepare test scenarios reflecting common user workflows and potential edge cases, including valid and invalid inputs.
- Execute the test by performing actions through the UI or device input, observing system responses.
- Record results, noting success, failure, or unexpected behavior.
- Document issues for later review, including steps to reproduce, observed behavior, and potential cause.
- Verify fixes after bugs are resolved, repeating the tests to ensure the system works as intended.

The testing flow is organized as follows:

- **Unit Checks (Manual):** Each currently implemented function is exercised individually in the local development environment to confirm correct behavior.
- **Integration Testing (Manual):** Related modules are tested together to verify correct interactions.
- **System Testing (Manual):** End-to-end workflows are tested using realistic scenarios, ensuring the complete system meets requirements.

- **User Acceptance Testing (Manual):** Key stakeholders perform guided use cases on currently implemented features to confirm readiness for operational use.

IX.3.1. Unit Testing

We will follow traditional unit testing practices by identifying the smallest testable units within the application and testing them in isolation from the rest of the system.

Since our project currently relies heavily on manual testing, unit testing will be conducted by:

- Running individual endpoints, login verification and inventory CRUD operations, directly after implementation.
- Verifying that each endpoint returns the correct HTTP responses, error messages, and state changes.
- Testing edge cases, such as:
  - Invalid barcode scans
  - Nonexistent user IDs
  - Negative or zero inventory quantities
  - Duplicate item creation
- Observing how the backend behaves when the card reader or barcode scanner provides unexpected input.

Typical units under test include:

- **User Login Handler**
  Tests verification of registered vs. unregistered users, admin authentication, and card swipe input.
- **Inventory Service Functions**
  Tests adding, updating, removing, and querying inventory items individually.
- **Device Input Handlers**
  Tests how raw card reader and barcode scanner data is parsed and validated.

IX.3.2. Integration Testing

Integration testing focuses on verifying that multiple components work together correctly. After unit-level behavior is confirmed, we gradually combine related modules to ensure they interact without errors.

Integration testing for this project includes:

Component Groups Tested

- **Login + Inventory Access**
  Ensures only authenticated users with correct roles can access administrative inventory pages.
- **Scanner Input + Inventory Update Logic**
  Confirms that scanning a barcode directly updates the right item, quantity, and timestamp.
- **Card Reader + User Lookup Process**
  Tests whether scanned card data is correctly translated into a valid user login session.

Testing Approach

Integration testing is entirely manual at this stage. The sequence typically follows:

- **Select a pair of components to test together** (login + inventory editing).
- **Set up realistic data** such as sample users and sample inventory items.
- **Trigger the integrated workflow** using the UI, scanner, or card reader.
- **Check for correct interactions**, including:
  - Database updates
  - Correct routing to protected pages
  - Proper role restrictions
  - UI consistency
- **Add more components** (login → inventory → admin dashboard) once no new faults appear.

IX.3.3. System Testing

IX.3.3.1. Functional Testing

Functional system testing is currently performed manually by the development team. Each major requirement in the project specification is matched to a corresponding functional test scenario.

Current functional testing includes:

- Logging in with a customer card, admin account, or invalid credentials
- Scanning inventory items and verifying correct behavior
- Updating quantities using the UI
- Attempting restricted actions as a non-admin user
- Ensuring error messages appear when expected

Because volunteer tracking and reporting are not yet implemented, they will be added to functional testing during the next development cycle.

Failures are documented along with the steps to reproduce the issue, and the responsible developer applies fixes before retesting.

IX.3.3.2. Performance Testing

Although extensive performance testing is not required for the current scope of the project, we still evaluate basic performance characteristics during system-level testing to ensure the application remains responsive under typical usage. This includes:

- Login response time
- Inventory update latency when scanning or editing items
- UI responsiveness under typical administrative workflows

Because the current system is small and uses local databases, performance testing is largely qualitative—based on developer observations during usage. As the application grows to include analytics, reporting, and increased inventory volume, more formal performance testing will be introduced.

IX.3.3.3. Standards and Constraints Verification/Testing

This system must comply with the non-functional requirements, standards, and constraints outlined in previous sections III.3., III.5., and VIII. of this document. These verification activities will occur after all major features (including volunteer tracking and report generation) are implemented. This testing plan will focus on the following:

- Non-functional requirements (security, performance, uptime, accuracy, usability, learnability, reporting)
    - ≥ 99% testing item recognition, ≥ 80% of users complete tasks with ≤ 20 minutes of application familiarization, reports for datasets ≤10,000 records generated within 5 seconds, etc.
- WSU brand standards and style compliance
- Security requirements
    - RBAC enforcement and MySQL encryption
- W3C WCAG 2.2 web content accessibility standards
    - Expect an application accessibility score of ≥ 90
- ACM Code of Ethics
- Project constraints (budget, time, accessibility)
    - Development complete with open-source or university-provided resources, required features implemented by end of second semester, system requires no formal training for operation

IX.3.3.4. User Acceptance Testing

UAT involves testing the application from the perspective of pantry administrators and volunteers to ensure the system matches their real-world needs. Our User Acceptance Testing plan includes:

1. Preparing the Test Environment

- A working deployment of the web application
- Sample users, inventory items, and test data
- Feedback forms or a shared document for collecting comments

2. Testing Activities

- Administrators will test login, inventory updates, and barcode scanning.
- Pantry workers will test scanning workflows as they would during check-in/out.
- Users will attempt invalid actions (scanning an unregistered card) to verify error handling.

3. Feedback and Revision

- The development team reviews all feedback.
- Necessary improvements are prioritized and assigned.
- A final verification round takes place after changes are implemented.

User acceptance testing ensures the system meets user expectations and is ready for real-world operational use. As more features are added next semester, additional user acceptance testing cycles will be conducted.

IX.4. Environment Requirements

IX.4.1. Hardware Requirements

- Developer laptops/workstations with 8–16GB RAM and 4+ CPU cores for running the Spring Boot backend, MySQL server, and IDE tools simultaneously.
- Optional testing servers (university-hosted or local VM) with 16GB RAM and expandable storage for database snapshots and deployment testing.

IX.4.2. Software Requirements

- **Operating Systems:** Windows, macOS, or Linux — all team members used local environments compatible with Java and MySQL.
- **Java Development Kit (JDK 17+):** Required for Spring Boot 3.x features, annotation processing, and application compilation.
- **Spring Boot 3.x Framework:** Used to build REST controllers, services, dependency injection, and authentication logic.
- **Maven Build System:** Manages dependencies, plugins, and build automation for consistent project setup.
- **Thymeleaf Template Engine:** Enables server-side rendering of HTML pages and form binding for the pantry UI.
- **MySQL Server 8.x:** Used as the primary RDBMS for persistent data storage (inventory, volunteers, cards, logs).
- **MySQL Workbench / CL:** Used for schema validation, querying, and debugging database issues.
- **IDE Tooling:** IntelliJ IDEA, Eclipse, or VS Code
  All configured with Spring Boot, Java, and Maven extensions.

IX.4.3. Network and Database Configuration

- Local MySQL instance configured with a shared schema name.
- Database connection set in application.properties:
- Local backend accessible at:
  `http://localhost:8080/`
- Common development pages:
  - /login
  - /cardReader
  - /inventory
  - /volunteer

IX.4.4. Hardware Integration Setup

- **Magstripe Card Reader:** Tested and verified to behave as a keyboard device. Automatically injects card number data into input fields.
- **Barcode Scanner:** Reads UPC codes directly into HTML form inputs without additional drivers or software.

These devices required no additional middleware, ensuring smooth integration with the web application's input forms.

IX.4.5. CI/CD and Build Execution

- **GitHub:** Used for collaboration, version control, and branching strategy (feature branches for each milestone).
- **Application start commands**:
  - IDE Run Configuration, or
  - Command line: mvn spring-boot:run
- **Standardized setup steps:**
  - Clone project repository
  - Configure MySQL
  - Update credentials
  - Run backend locally

X. Alpha Prototype Description

Of the essential features required for this system, our team has implemented the basics of inventory management and customer and admin sign-in. We have since improved such features when reviewing client requirements and receiving client feedback. Further refinements on these features will be made in future development, which will also include the implementation of other essential features such as volunteer hour tracking and report generation.

User-facing pages, data persistence, and early hardware integration have been our focus for this semester’s development process. The current prototype validates our proposed architecture’s feasibility and confirms that our environmental requirements operate together as expected.

Current user interface design has been shown in Figures 4 – 9 above. Because this application is currently only a prototype for testing feature implementation, this user interface design is not final and will be subject to many changes in the near future. It is important to both our clients and our team that feature testing is verified with this prototype, while user interface design will be handled at a different time.

Testing of these features has been handled manually by our team thus far. This includes navigating through the application to ensure the pages are correctly linked, using the integrated hardware to verify expected operation, and navigating feature components in individual units.

X.1. Sign-In System

X.1.1. Functions and Interfaces Implemented

As shown in earlier figures, the sign-in system has been broken down into two basic sections based on our RBAC model: customer login and admin login. These two types of users have different access based on their roles, meaning that customers may not interact with admin-related features such as inventory management and volunteer forms.

An early version of user recollection has also been implemented in this subsystem. Pantry customers are prompted to enter their major and class standing during their first visit to the Voiland Food Pantry. This information will be remembered upon subsequent visits, allowing each customer to complete their visit with less hassle.

Remaining work includes further integration with the magstripe reader hardware for this portion of the project, as well as integration with other future components and further focus on what role-based operations each user may perform. Consulting with our clients will be key for a thorough sign-in system implementation.

X.1.2. Preliminary Tests

Our team performed both unit and integration testing within this subsystem and subsystems related to it. The prototype successfully performs the following:

- First-time users are prompted to provide non-identifying information for pantry analytics.
- Returning users are recalled and not prompted again for information related to clientele analytics.
- Customers and staff/admin users have separate sign-in pages, which leads to separate user homepage with RBAC. Customers do not have access to admin features.

X.2. Inventory Management System

X.2.1. Functions and Interfaces Implemented

Shown in Figure 9 above, the inventory management system is currently the feature with the most components implemented. The food inventory database has been created, allowing staff/admin roles to add, remove, and edit entries in this database. Hardware integration with the barcode scanner has also been implemented, meaning that these users can use the scanner on non-perishable food items to change the available stock stored in the database. Based on client feedback, a net weight tracker has also been added to keep count of how many ounces of food were donated in a given time period.

A variable quantity input feature was also added to the inventory to account for large donations or boxes of food items that come with many units. This way, staff adding or subtracting stock from the food inventory do not have to individually scan each item, and they can simply handle multiple items at once. With this, error handling has been added. For example, an error is shown if attempting to remove one unit from an item with a stock of zero.

Remaining work includes low stock notifications, UI confirmation messages, additional fields for the inventory database, and relations with other databases for future report generation. Our main focus will be storing food that is donated in another database for the eventual feature of monthly donation reports for pantry staff and admins.

X.2.2. Preliminary Tests

Our team performed both unit and integration testing within this subsystem and subsystems related to it. The prototype successfully performs the following:

- Add, remove, and edit information related to items and their stock in the database.
- Use the barcode scanner to add new items to the database and increase the stock of items that already exist in the database.
- Track net weight in ounces of donated food.
- Add or remove multiple items at once without having to rescan them.
- Search for food items in the database by name.
- Handles errors, such as subtracting from a stock of 0.

XI. Alpha Prototype Demonstration

This prototype demonstration showcased basic UI design for the sake of integration testing for navigation flow. The application homepage, sign-in pages, inventory management database, and hardware integrations showed with preliminary database operations based on information communicated to us by the database team not part of this course.

Based on our mentor's suggestions, we will be further improving the sign-in system portion of this project, which will allow our team to tweak our data persistence system accordingly. We will also be working on the front-end to work towards our goal of making an attractive user interface that is intuitive for all users of our system.

XII. Future Work

Major tasks in the upcoming second semester include implementation of final features such as volunteer tracking, report generation, better error handling, and further quality improvements to ensure our system remains intuitive for all users. Our team's plan for this will follow a similar format to that of the current semester's plan: improving the functionality of our prototype's interactable features, then later focusing on improving user interface design for our final product.

In particular, we will focus on the following key objectives:

- **Testing:** Our team will continue to perform thorough manual testing to ensure the system behaves as expected, guaranteeing that the Voiland Food Pantry will remain in smooth operation in the future.
- **Scalability:** Ensure the functionality of the Voiland Food Pantry by storing datasets with ≤10,000 records for visit tracking, volunteer tracking, and food inventory.
- **UI Design:** Improve the user interface design according to feedback from our clients. This will ensure the UI is intuitive and strikes a balance between attractiveness and simplicity. This will inspire trust in pantry clientele and make our system navigable by any type of user.
- **Volunteer Tracking:** Add volunteer features to staff pages to track the hours each volunteer spends assisting the operation of the Voiland Food Pantry.
- **Report Generation:** Add report generation features according to client requirements and feedback. Particularly, allow monthly stock reports to be sent to pantry administrators and allow volunteers to print hour tracking reports. Additional report generation requirements may be added in the future.

With these focal points in mind, our team aims to improve our current work to give our clients a satisfactory application to streamline the operation of the Voiland Food Pantry, which we hope will further assist in the scale of this WSU facility.

XIII. Glossary

CI/CD: Continuous Integration/Continuous Delivery. CI involves making small changes to improve code and merging them to a repository, while CD is the automatic release of validated changes to testing and development environments.

CRUD: Create, Read, Update, and Delete. These are operations for managing persistent data in a database.

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

JPA Repository: Access layer built on the Java Persistence API (JPA). This makes managing database operations much easier.

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.

Spring Boot: An open-source Java framework used to create standalone backend applications. Simplifies build configurations.

Thymeleaf: A Java XML/XHTML/HTML5 template engine used to generate HTML views. Integrates with Spring Boot to create web applications built on an MVC architecture.

XIV. References

[1] "Estimated Eligibility and Receipt among Food Insecure College Students," Government Accountability Office. [Online]. Available: https://www.gao.gov/assets/gao-24-107074.pdf. [Accessed: Sep. 12, 2025].

[2] "Cougar Food Pantry." Washington State University. [Online]. Available: https://cce.wsu.edu/resources/student-resources/cougar-food-pantry/. [Accessed: Sep. 12, 2025].

[3] "Go Cougs." Washington State University. [Online]. Available: https://brand.wsu.edu/. [Accessed: Sep. 12, 2025].

[4] "IEEE Recommended Practice for Software Requirements Specifications." Institute of Electrical and Electronics Engineers. [Online]. Available: http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf. [Accessed: Nov. 10, 2025]

[5] "IEEE Standard for Information Technology—Systems Design—Software Design Descriptions." Institute of Electrical and Electronics Engineers. [Online]. Available: https://cengproject.cankaya.edu.tr/wp-content/uploads/sites/10/2017/12/SDD-ieee-1016-2009.pdf. [Accessed: Nov. 10, 2025]

[6] "Web Content Accessibility Guidelines (WCAG) 2.2." World Wide Web Consortium. [Online]. Available: https://www.w3.org/TR/WCAG22/. [Accessed: Nov. 10, 2025].

[7] "ACM Code of Ethics and Professional Conduct." Association for Computing Machinery. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed: Nov. 10, 2025].

[8] "Thymeleaf Home." Thymeleaf.org. [Online]. Available: https://www.thymeleaf.org/. [Accessed: Nov. 27, 2025].

[9] "Spring Boot Overview," Spring.io. [Online]. Available: https://spring.io/projects/spring-boot [Accessed: Nov. 27, 2025].

XV. Appendix

XV.1. Appendix A – User Stories and Acceptance Scenarios

Additional User Stories
The following are additional user stories for reference.

User Story 4: Inventory Data Reporting
As a pantry administrator, I want to access inventory data so that I can print reports.
Feature: Access Inventory Data
Scenario: Normal access
• **Given:** I am an administrator for the pantry
• **When:** I access the inventory database
• **Then:** I can view current inventory data
• **And:** I can generate and print reports
Scenario: Database not functional
• **Given:** I am an administrator for the pantry
• **When:** The inventory database is down
• **Then:** I use paper records and past reports
• **And:** I generate reports manually
• **And:** The technical issue is reported for repair

User Story 5: Volunteer Data Reporting
As a pantry administrator, I want to access volunteer data so that I can print reports.
Feature: Access Volunteer Data
Scenario: Normal access
• **Given:** I am an administrator for the pantry
• **When:** I access the volunteer database
• **Then:** I can view current volunteer data
• **And:** I can generate and print reports
Scenario: Database not current
• **Given:** I am an administrator for the pantry
• **When:** The volunteer database is outdated
• **Then:** I refer to paper reports
• **And:** I update the database manually
• **And:** I generate and print reports

User Story 6: Donator Website Access
As a food pantry donator, I want to access the food pantry website so that I can contact supervisors.
Feature: Website Contact Info
Scenario: Normal website access
• **Given:** I am a donator
• **When:** I access the food pantry website
• **Then:** I see supervisor contact information
• **And:** I can coordinate my donation
Scenario: Website malfunction
• **Given:** I am a donator
• **When:** The website is not working
• **Then:** A custom error screen displays contact info
• **And:** I can still contact the supervisor.

User Story 7: Customer Website Access
As a customer, I want to access the food pantry website so that I can view the hours.
Feature: Website Hours Info
Scenario: Normal website access
• **Given:** I am a customer
• **When:** I access the food pantry website
• **Then:** I can see the hours of operation
Scenario: Website malfunction
• **Given:** I am a customer
• When: The website is not working
• Then: A custom error screen displays contact info
• And: I can contact the pantry to learn hours

User Story US-8: Low Stock Alert Notification
As the system, I want to notify administrators when stock is low so that inventory can be replenished in time.
Feature: Low Stock Alert
Scenario: Item below threshold

- **Given:** The system is monitoring inventory
- **When:** An item's stock falls below the threshold
- **Then:** An alert is displayed on the administrator dashboard
- **And:** An email notification is sent to the administrator

Scenario: Item above threshold

- **Given:** The system is monitoring inventory
- **When:** An item's stock is above the threshold
- **Then:** No alert is sent

XV.2. Appendix B – Example Testing Strategy

- Identify the requirements to be tested. All test cases shall be derived using the current Software Requirements Specification.

- Identify which particular test(s) will be used to test each module.

- Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.

- Identify the expected results for each test.

- Document the test case configuration, test data, and expected results.

- Perform the test(s).

- Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the revised Test Plan document.

- Successful unit testing is required before the unit is eligible for component integration/system testing.

- Unsuccessful testing requires a bug form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.

- Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.