**Software Requirements Specification  
PRJ566 – Fall 2024**

**PRJ566 – Team No:** Group 2

**Name of Project:** ByteBasket: A Food Bank Assistance Web Application  
 **Project Leader:** Davyd Kuleba

**Last updated:** 1/31/2025

**Team Members:**

1. Davyd Kuleba

2. Saeed Bafana

3. Claudia Suarez Socorro

4. Ahnaf Abrar Khan

5. Ulas Cagin Ondev

# TABLE OF CONTENTS

1. **Introduction/Overview - Document Information**
   1. Document Authors
   2. Revision History
   3. Document Conventions
   4. Document Purpose
   5. Intended Audience
   6. Group Agreement
2. **Project Overview**
   1. Project Proposal
   2. Stakeholders and Users
   3. Project Scope
   4. Functional Requirements
   5. Non-functional Requirements
   6. System Risks
   7. Operating Environment
   8. UI/UXD Interface Mockups
3. **Process & Data Modeling**
   1. UML Modeling: DFDs & Activity Diagrams
   2. Use Case Specification
      1. Business Rules
      2. System Use Case Diagrams
      3. Use Case Description Tables
4. **Domain Class Diagram**
5. **Database (Select either 5.1 or 5.2)**
   1. RDBMS Artifacts
      1. Scripts to create, populate, delete tables
      2. Data Dictionary
   2. NoSQL Artifacts
6. **Work Breakdown Structure (WBS)**
7. **Milestones & Acceptance Criteria**
8. **Implementation Schedule (Agile/Waterfall)**
9. **Client / Faculty Sign-off**

# 1 - Introduction/Overview - Document Information

## 1.1 Document Authors

1. Davyd Kuleba

2. Saeed Bafana

3. Claudia Suarez Socorro

4. Ahnaf Abrar Khan

5. Ulas Cagin Ondev

## 1.2 Revision History

|  |  |
| --- | --- |
| Week 03 | Sections of this document that were completed/updated this week.  1.0 Introduction/Overview  1.1 Document Authors  1.2 Revision History  1.4 Document Purpose  1.5 Intended Audience  1.6 Group Agreement  2.0 Project Overview  2.1 Project Proposal |
| Week 04 | Sections of this document that were completed/updated this week.  1.0 Introduction/Overview  1.4 Document Purpose  1.5 Intended Audience  2.0 Project Overview  2.1 Project Proposal  2.2 Stakeholders and Users  2.5 Nonfunctional Requirements  2.6 System Risks  2.7 Operating Environment |
| Week 05 | 2.0 Project Overview  2.3 Functional Requirements  2.4 Non-functional Requirements  3.1 UML Modeling: DFD |
| Week 06 | 3.1 Activity Diagram.  2.8 UI/UXD (Wireframes/Mock-ups) |
| Week 07 | * 1. Use Case Specification   3.2.1 Business Rules  3.2.2 System Use Case Diagrams 3.2.3 Use Case Descriptions |
| Week 08 | 2.8 Finalized Mockup |
| Week 09 | Recorded Presentation |
| Week 10 | 5. Database  5.1 Database Design  5.2 ERD Diagram |
| Week 11 | WBS  Implementation Plan |
| Week 12 | Product Backlogs |
| Week 13 | Completed SRS |

## 1.3 Document Conventions

Any text in red indicates an exception or error.

Any text in blue is in-progress.

Any text highlighted in yellow is an important point.

Any text in green was recently added.

Any text *italicized* represents definitions.

Any text with ~~strike-through~~ is deleted.

1.4 Document Purpose

## The Software Requirements Specification (SRS) for ByteBasket provides a comprehensive blueprint for developing a web application that revolutionizes food bank operations. This document serves multiple purposes:

## Details the technical and functional requirements for a system that streamlines food bank management, donation tracking, and distribution processes

## Establishes clear guidelines for inventory management and dietary requirement tracking

## Outlines the user interaction flows for both administrators and beneficiaries

## Defines the security and data handling protocols to protect user privacy

## Specifies the performance requirements to handle concurrent users and multiple food bank locations

## This SRS will guide development decisions, serve as a reference for stakeholders, and provide the foundation for testing and validation procedures.

## 1.5 Intended Audience

This document serves as a reference for multiple stakeholders involved in the ByteBasket project:

Food Bank Administrators

* Management staff responsible for overseeing food bank operations
* Inventory managers who will use the system daily
* Staff members involved in food distribution and donation acceptance
* Training coordinators who will teach others to use the system

Developers & System Architects

* Frontend and backend development teams
* Database administrators
* UI/UX designers
* Quality assurance testers
* System integration specialists

Donors & Volunteers

* Individual and corporate food donors
* Community volunteers
* Partner organizations
* Food delivery coordinators

Academic Advisors & Evaluators

* Project supervisors
* Technical advisors
* Academic assessment team
* External reviewers

Each audience member will find relevant information about system functionality, technical requirements, and implementation details pertinent to their role in the project.

## 1.6 Group Agreement

**TEAM AGREEMENT**

**Team #:** Group 2

**Project Title:** ByteBasket: A Food Bank Assistance Web Application

**Project Time Frame:** January 2025 – August 2025

**Team Members:**

1. Davyd Kuleba  
2. Saeed Bafana  
3. Claudia Suarez Socorro  
4. Ahnaf Abrar Khan  
5. Ulas Cagin Ondev

**Team Leadership:**Project Leader: Davyd Kuleba – Responsible for overall project direction, ensuring timely progress, and resolving conflicts.

Technical Lead: Saeed Bafana – Oversees software design, coding standards, and technical implementation.

Documentation Lead: Claudia Suarez Socorro – Responsible for maintaining and updating the SRS and all project documentation.

UI/UX Lead: Ahnaf Abrar Khan – Focuses on the design and usability of the web application, ensuring accessibility and user satisfaction.

Testing and Quality Assurance Lead: Ulas Cagin Ondev – Manages testing processes, identifies bugs, and ensures the system meets quality standards.

**Team Functions:**

* **Tools:** MS Teams (communication), OneDrive (file storage), WhatsApp (quick updates), Email (formal correspondence).
* **Meeting Schedule:** Weekly meetings every **Wednesday at 3 PM**, conducted virtually through MS Teams.
* **Issue Reporting:** Any problems will be reported via MS Teams and addressed during weekly meetings.

**Team Commitment**

The undersigned members agree to work together on the project until the end of the PRJ666 next Semester. They recognize that as a team and individually they are responsible for the quality of all deliverables.

|  |  |
| --- | --- |
| **Name:** | **Date:** |
| Davyd Kuleba | 1/24/2025 |
| Saeed Bafana | 1/24/2025 |
| Claudia Suarez Socorro | 1/24/2025 |
| Ahnaf Abrar Khan | 1/24/2025 |
| Ulas Cagin Ondev | 1/24/2025 |

**2 - Project Overview**

## 2.1 Project Proposal

**Project Background:**

Food insecurity remains a critical challenge in our communities, affecting countless individuals and families. Traditional food bank operations often face significant challenges in efficiently managing their resources and meeting diverse community needs. While food banks provide essential services, many struggle with:

* Manual inventory tracking leading to inefficiencies
* Difficulty matching donations with specific dietary needs
* Limited communication between donors, volunteers, and recipients
* Challenges in predicting and managing demand
* Lack of real-time visibility into available resources

ByteBasket emerges as a solution to modernize food bank operations through a web-based platform that connects all stakeholders in the food distribution ecosystem. By leveraging technology, we aim to transform how food banks operate and serve their communities.

**Problem Statement:**

|  |  |
| --- | --- |
| The Problem of: | - Inefficient food bank management impacting food distribution  - Manual inventory tracking leading to waste and shortages  - Poor visibility of real-time needs and available resources  - Limited ability to match dietary restrictions with available foods |
| Affects: | - Food banks struggling to manage resources efficiently- Donors unsure of current needs  - Volunteers lacking clear coordination  - Recipients facing uncertainty about available resources  - Community members with specific dietary requirements |
| The impact of which is: | - Reduced efficiency in resource allocation  - Food waste due to poor inventory management- Difficulty meeting specific dietary or cultural needs  - Limited transparency in available food supplies  - Inefficient volunteer resource utilization  - Increased operational costs for food banks |
| A successful solution would: | - Streamline inventory tracking and management- Enable targeted food distribution based on needs  - Enhance user engagement with food banks  - Provide real-time visibility of resources  - Facilitate better volunteer coordination  - Support diverse dietary requirements  - Reduce operational overhead |

**Product Vision:**

|  |  |
| --- | --- |
| For: | - Food banks seeking to modernize operations  - Donors wanting to make impactful contributions  - Individuals and families facing food insecurity  - Volunteers looking to serve their community effectively |
| Who: | - Need a more effective and transparent food donation system  - Require better coordination of resources  - Want to ensure their donations meet specific community needs  - Seek to efficiently manage and distribute food resources |
| The Product Name: | ByteBasket |
| That: | - Enables wishlist-based food requests  - Provides real-time inventory tracking  - Coordinates volunteer opportunities  - Matches donations with specific needs  - Supports dietary restriction management  - Offers analytics and reporting tools |
| Unlike: | - Traditional paper-based inventory systems  - Basic spreadsheet tracking methods  - Disconnected communication channels  - Systems lacking user-driven demand insights |
| Our product: | - Integrates user-friendly wishlist features  - Provides donor engagement tools  - Offers real-time inventory updates  - Supports multiple food bank locations  - Enables dietary restriction tracking  - Features mobile-responsive design  - Facilitates volunteer coordination  - Generates actionable insights through data analytics |

## 2.2 Stakeholders and Users

|  |  |
| --- | --- |
| **Stakeholder Name/Identifier** | **Category** |
| Food Bank Administrator | Administration, Primary User - Needs complete system access for managing inventory, users, and operations |
| Inventory Manager | Administration, User - Requires real-time tracking and update capabilities for food stock management |
| Donation Coordinator | User - Manages incoming donations and donor relationships |
| Food Bank Staff | User - Needs access to process requests and manage daily operations |
| Volunteer Coordinator | User - Manages volunteer schedules and assignments |
| System Administrator | Administration, Technical Support - Maintains system functionality and user support |
| Donors | External User - Requires donation tracking and impact visibility |
| Recipients | External User - Needs access to food availability and request submission |
| Development Team | Developers - Responsible for system implementation and maintenance |
| Project Sponsors | Administration, Sponsor - Oversees project success and resource allocation |

## 2.3 Functional Requirements

1. User Management
   1. System shall allow user registration with role-based access (Admin, Staff, Donor, Recipient, Volunteer)
   2. System shall support user authentication using email and password
   3. System shall enable profile management including dietary restrictions and preferences
   4. System shall allow administrators to manage user roles and permissions
2. Inventory Management
   1. System shall track real-time inventory levels across multiple food bank locations
   2. System shall categorize food items by type, expiration date, and dietary categories
   3. System shall generate alerts for low stock items and approaching expiration dates
   4. System shall support barcode/QR code scanning for quick item entry
3. Donation Management
   1. System shall allow donors to register and track their donations
   2. System shall generate donation receipts automatically
   3. System shall enable donors to view their donation history and impact
   4. System shall support scheduling of donation drop-offs
4. Request Processing
   1. System shall allow recipients to create and submit food requests
   2. System shall support wishlist creation based on available items
   3. System shall automatically match requests with dietary restrictions
   4. System shall enable status tracking of submitted requests
5. Volunteer Coordination
   1. System shall support volunteer registration and scheduling
   2. System shall allow volunteers to sign up for available shifts
   3. System shall track volunteer hours and activities
   4. System shall enable volunteer coordinators to manage assignments
6. Reporting and Analytics
   1. System shall generate reports on inventory levels and movement
   2. System shall track donation patterns and donor engagement
   3. System shall provide analytics on recipient needs and fulfillment
   4. System shall offer customizable dashboard views for different user roles

## 2.4 Nonfunctional Requirements

1. Performance
   1. System shall support up to 500 concurrent users
   2. System shall process transactions within 3 seconds
   3. System shall maintain 99.9% uptime during operating hours
   4. System shall support page load times under 2 seconds
2. Security
   1. System shall encrypt all sensitive data in transit and at rest
   2. System shall enforce password complexity requirements
   3. System shall implement role-based access control
   4. System shall maintain audit logs of all system activities
   5. System shall automatically log out inactive sessions after 30 minutes
3. Usability
   1. System shall be accessible on mobile devices through responsive design
   2. System shall support multiple languages (initially English, French, Spanish)
   3. System shall provide intuitive navigation requiring no more than 3 clicks to reach any feature
   4. System shall include help documentation and tooltips
4. Reliability
   1. System shall perform automated data backups every 24 hours
   2. System shall maintain data integrity during concurrent operations
   3. System shall provide error handling with user-friendly messages
   4. System shall support automatic recovery from system failures
5. Scalability
   1. System shall be capable of handling 50% annual growth in user base
   2. System shall support addition of new food bank locations without code changes
   3. System shall maintain performance levels as data volume increases
   4. System shall support horizontal scaling of server resources

## 2.5 Project Scope

**In Scope:**

1. Web Application Development

* User registration and authentication system
* Mobile-responsive interface
* Support for multiple food bank locations
* Inventory management system
* Request processing system

1. Core Functionality

* Real-time inventory tracking
* Dietary restriction management
* Donation management
* User request system
* Volunteer coordination
* Basic reporting and analytics

1. Initial Deployment

* Support for up to 10 food banks
* Capacity for 1000+ users
* Handling 500+ concurrent users
* Basic training materials
* System documentation

**Out of Scope:**

1. Hardware provisioning for food banks
2. Integration with external payment systems
3. Mobile app development (native apps)
4. Advanced analytics and predictive modeling
5. Automated delivery scheduling
6. Third-party logistics integration

## 2.6 System Risks

|  |  |
| --- | --- |
| **Risk** | **Response** |
| System Downtime | - Implement robust backup systems  - Establish regular maintenance schedule  - Provide offline operation procedures |
| Data Loss | - Regular automated backups  - Secure data storage solutions  - Redundant database systems |
| User Adoption | - Provide comprehensive training  - Create user-friendly interfaces  - Offer ongoing support |
| Performance Issues | - Regular performance monitoring  - Scalable infrastructure  - Load balancing implementation |
| Security Breaches | - Regular security audits  - Encrypted data transmission  - Secure authentication protocols |
| Internet Connectivity | - Offline data caching  - Synchronization capabilities  - Low-bandwidth operation mode |

## 2.7 Operating Environment

1. Hardware Requirements

* Server: Cloud-based hosting with scalable resources
* Client: Any device with web browser support
* Minimum 8GB RAM for server operations
* Adequate storage for database and file system

1. Software Requirements

* Modern web browsers (Chrome, Firefox, Safari, Edge)
* Operating System: Platform independent
* Database: SQL or NoSQL based on final architecture decision
* Web Server: Apache/Nginx

1. Network Requirements

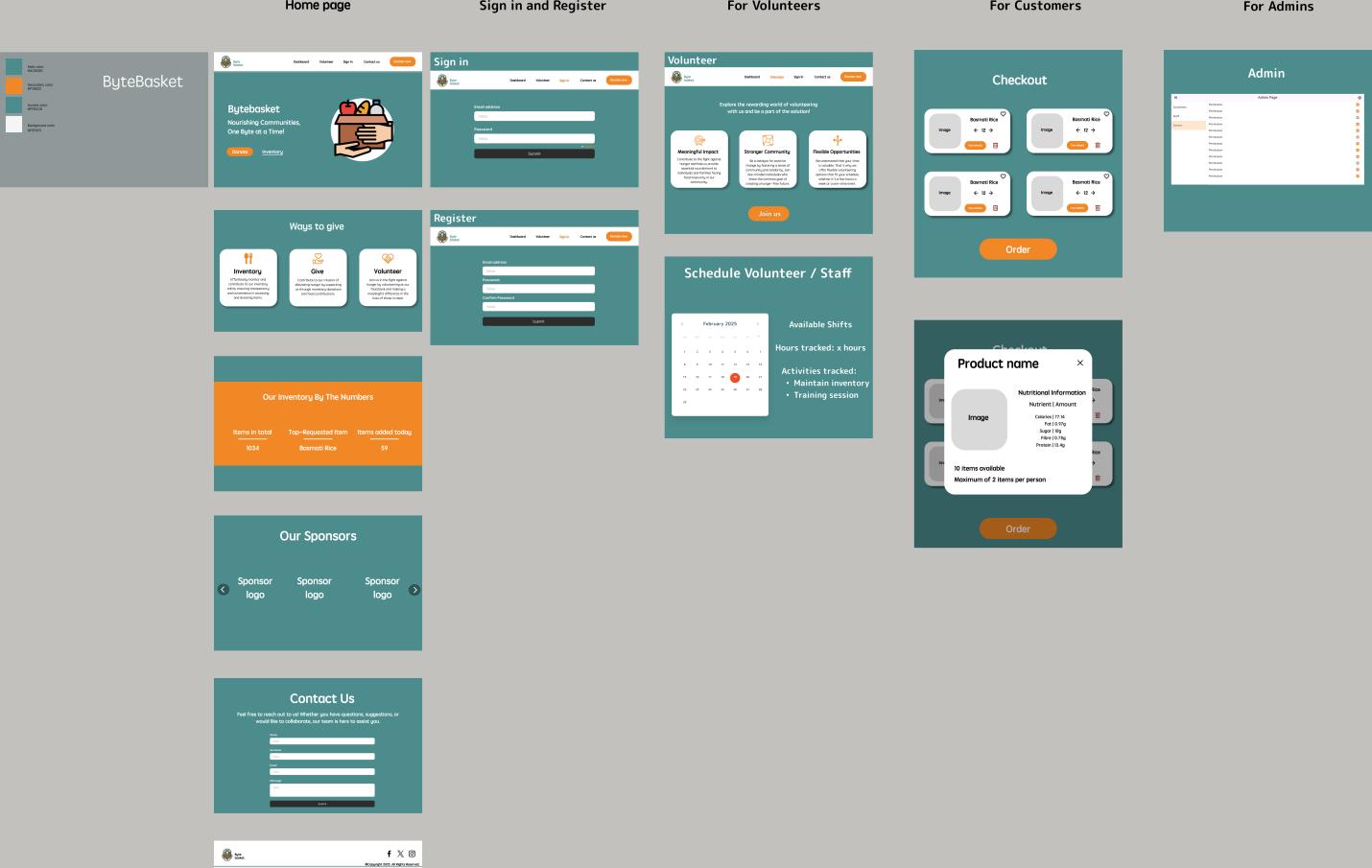
* Minimum 1Mbps internet connection
* Support for HTTPS protocol
* WebSocket support for real-time updates

1. Security Requirements

* SSL/TLS encryption
* Regular security updates
* Secure user authentication
* Data backup and recovery systems

1. User Interface Requirements
2. Responsive design for all screen sizes
3. Support for touch interfaces
4. Accessibility compliance
5. Multi-language support capability

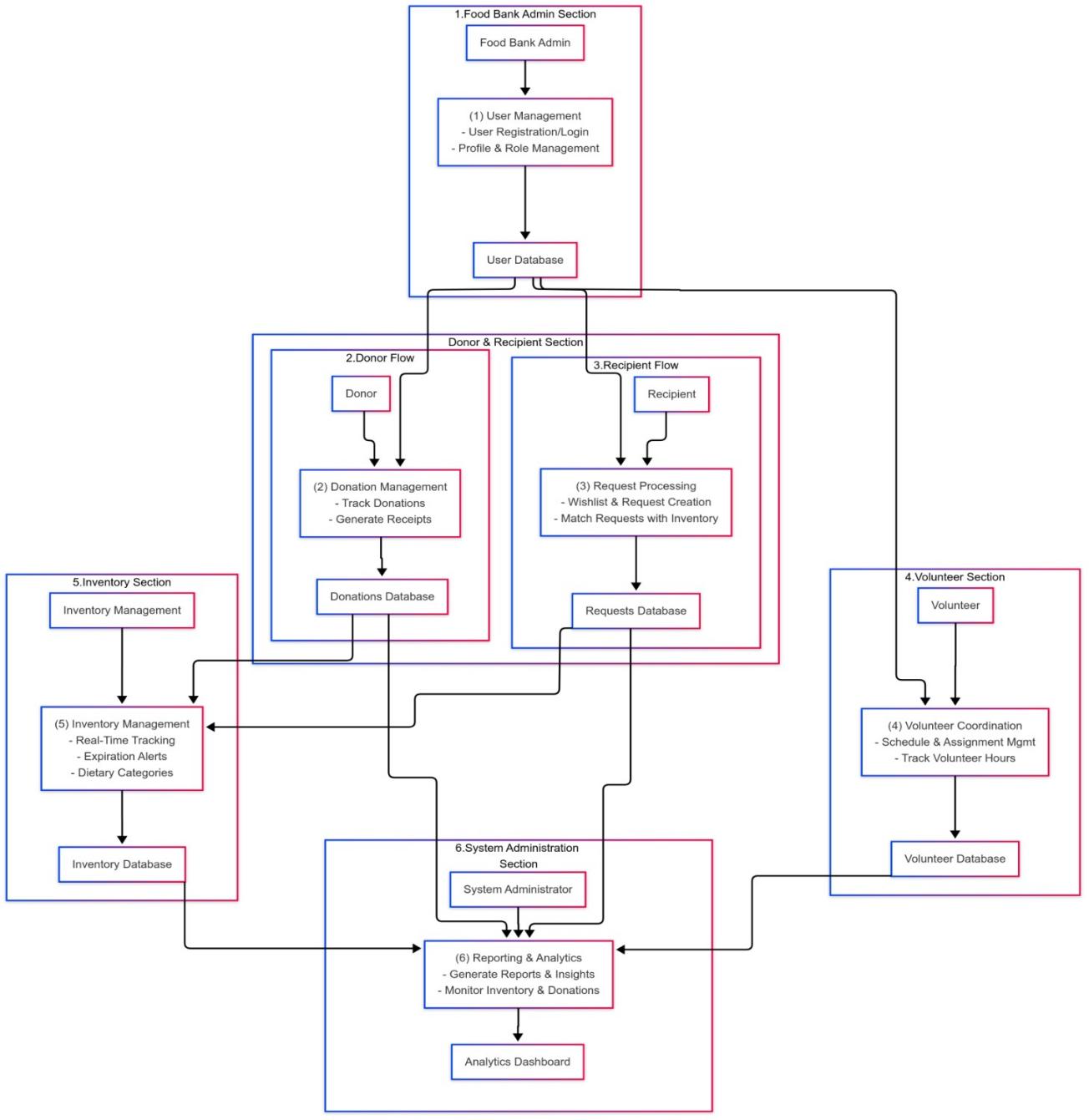
## 2.8 UI/UXD Interface Mock-ups

(Figma file uploaded on github repository)

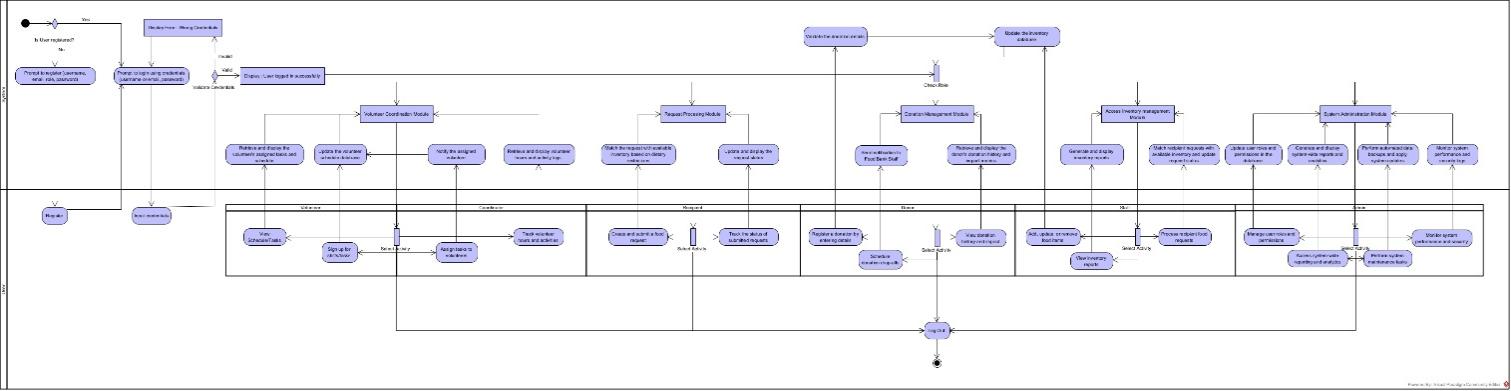
Figma link: https://www.figma.com/design/4oVEyc2QMZVtU5tpjwG2wK/BYTEBASKET?node-id=0-1&t=FVl2dsHA6CTv20GY-1

# Process and Data Modeling

## **3.1 UML/DFD Modeling and Data Modeling:** **DFD:**



**Activity Diagram:**

(Visual Paradigm file uploaded on github repository)

## **3.2 Business Rules**

|  |  |  |
| --- | --- | --- |
| Business Rule Number | Business Rule Description | Related UC |
| BR01 | User must provide a username, email, and password to register for the app | UC01 |
| BR02 | Each user must be assigned one of the following roles: Admin, Staff, Donor, Recipient, or Volunteer | UC01 |
| BR03 | Food inventory items must have categories, expiration dates, and dietary information | UC04 |
| BR04 | System must alert when inventory items are low or approaching expiration | UC06 |
| BR05 | Donors must receive an automated receipt for their contributions | UC07 |
| BR06 | Food requests must be processed based on availability and recipient dietary restrictions | UC08 |
| BR07 | Recipients can only request items that are currently in stock | UC08 |
| BR08 | Volunteer hours must be logged and tracked in the system | UC09 |
| BR09 | System must maintain at least 99.9% uptime during operating hours | UC12 |
| BR10 | All sensitive user data must be encrypted in transit and at rest | UC12 |
| BR11 | Users will be automatically logged out after 30 minutes of inactivity | UC01 |
| BR12 | System must support multiple food bank locations with separate inventory tracking | UC04 |
| BR13 | Reports must be generated based on actual inventory levels and transaction history | UC10 |
| BR14 | Donation drop-offs must be scheduled during food bank operating hours | UC07 |
| BR15 | Volunteer shifts must be approved by a coordinator before being confirmed | UC09 |
| BR16 | Recipients must verify their identity before receiving food assistance | UC08 |
| BR17 | Food items with dietary restrictions must be clearly labeled in the system | UC04 |
| BR18 | System must support barcode/QR code scanning for inventory management | UC04 |
| BR19 | Dashboards must be customized based on user roles | UC11 |
| BR20 | System must support multilingual interfaces (English, French, Spanish) | UC12 |

## **3.3 Use Case Specifications with corresponding interface mockups:**

**Each use case needs to have the following:**

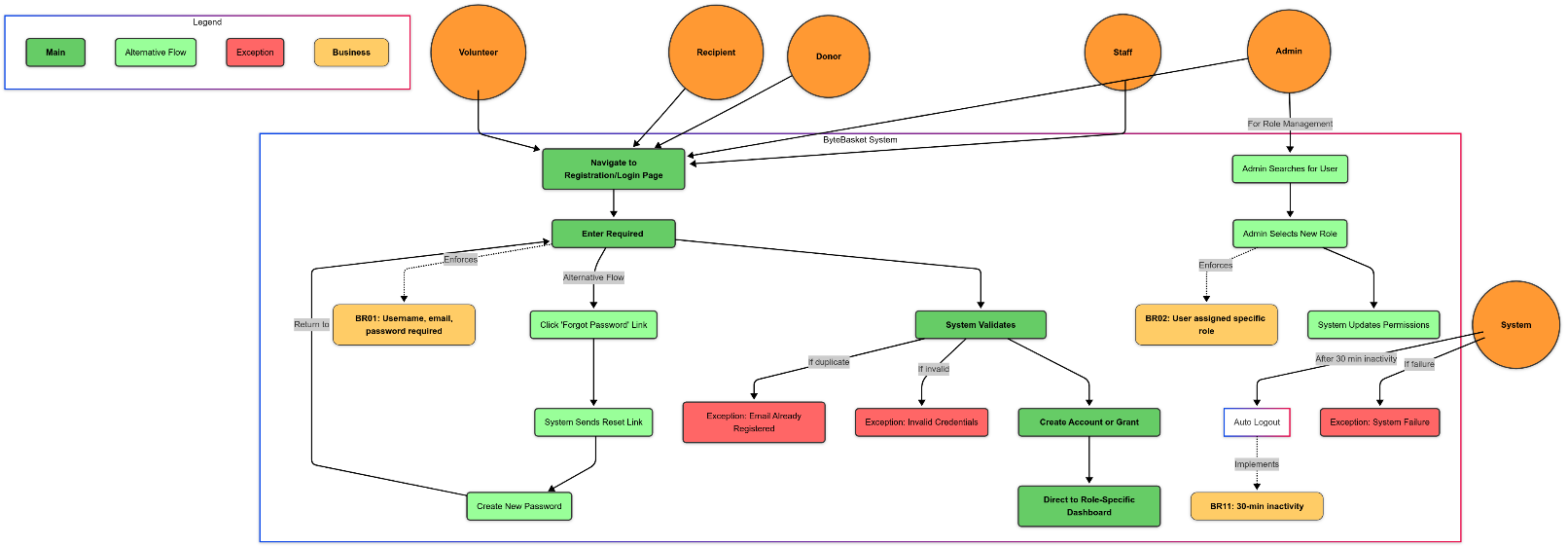
**1-** **Business Rules.**

**2- System Use Case Diagrams.**

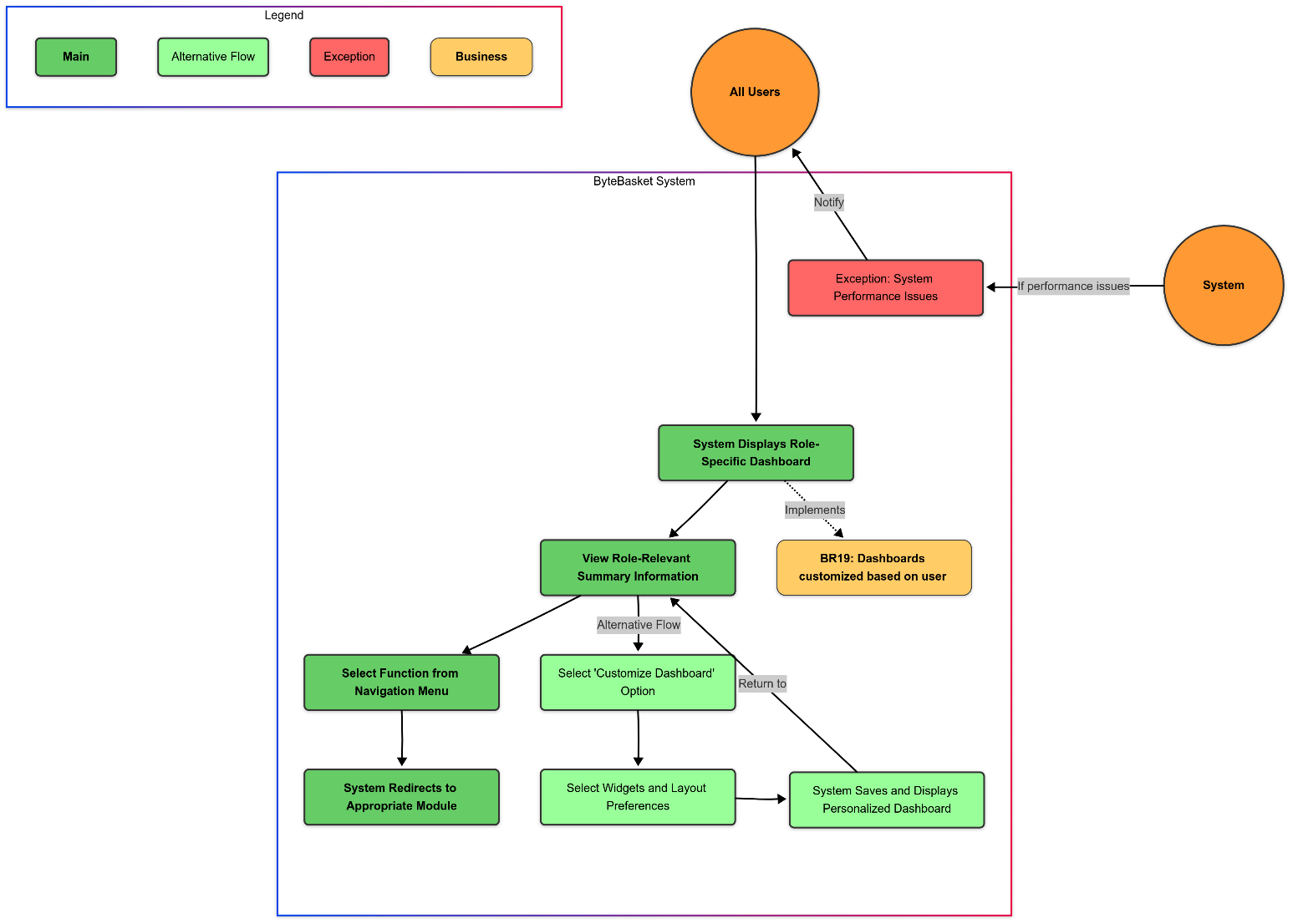
**3- Use Case Descriptions.**

**4- Corresponding Mockups**

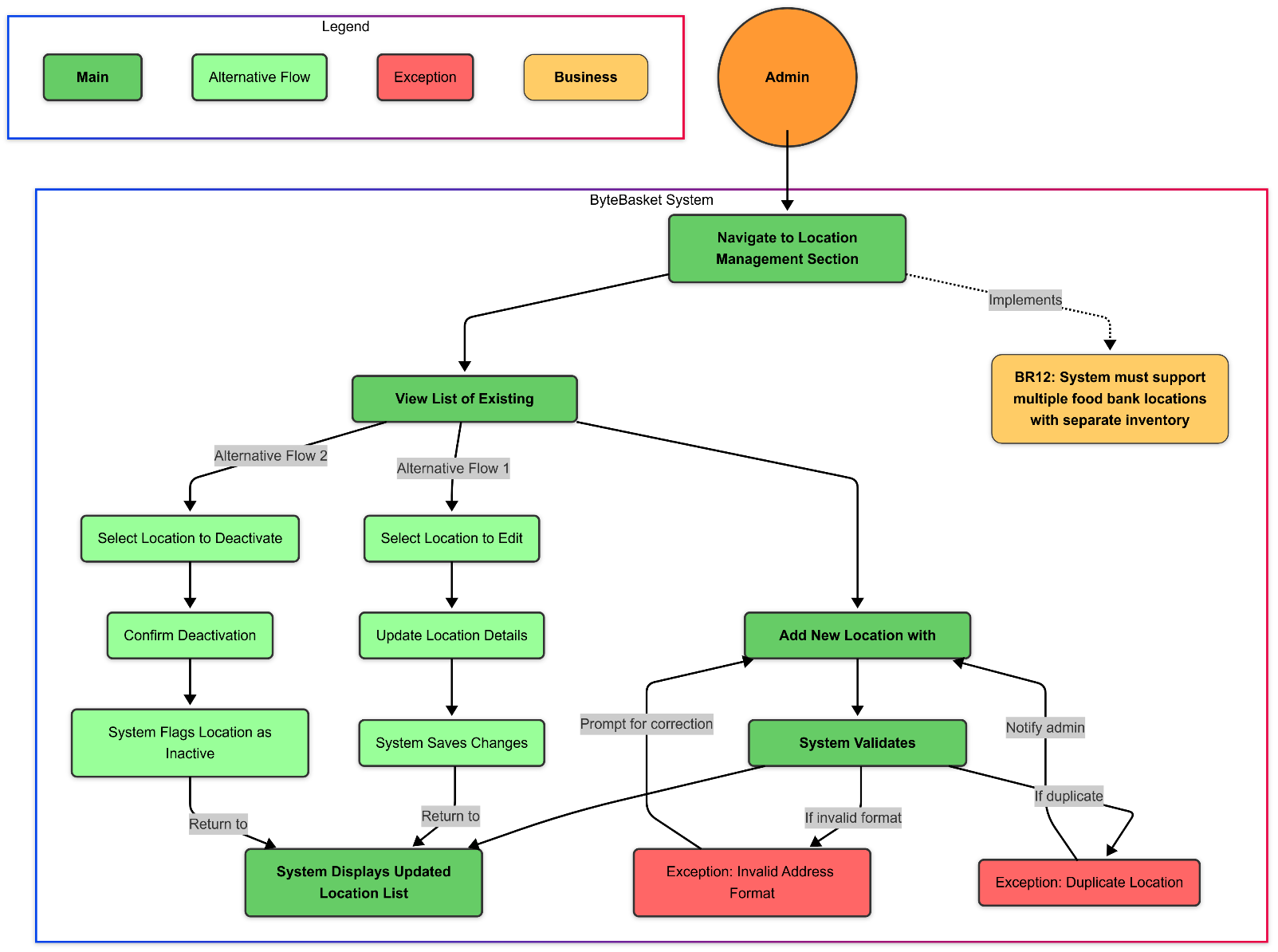
|  |  |
| --- | --- |
| Use Case Identifier | UC01 |
| Use Case Title | User Management |
| Actor(s) | Admin, Staff, Donor, Recipient, Volunteer |
| Description | This use case describes the processes for user registration, login, profile management, and role assignment in the ByteBasket system. |
| Trigger | User needs to create an account, login, update profile, or admin needs to manage user roles |
| Precondition(s) | 1. System is operational 2. For login: User has a registered account 3. For role management: Admin has logged in |
| Normal Flow of Events | 1. User navigates to the registration/login page 2. User enters required credentials 3. System validates the information 4. System creates user account or grants access 5. User is directed to their role-specific dashboard |
| Alternative Flow of Events | **If user forgets password:**   1. User clicks "Forgot Password" link 2. System sends password reset link to registered email 3. User follows link to create new password   **If admin is managing roles:**   1. Admin searches for user account 2. Admin selects new role for user 3. System updates user permissions |
| Postcondition(s) | 1. User account is created, updated, or accessed 2. User permissions reflect their assigned role |
| Exception(s) | **Invalid credentials:** System displays error message  **Email already registered:** System notifies user  **System failure:** User is notified to try again later |
| Bussines Rule(s) | BR01, BR02, BR11 |



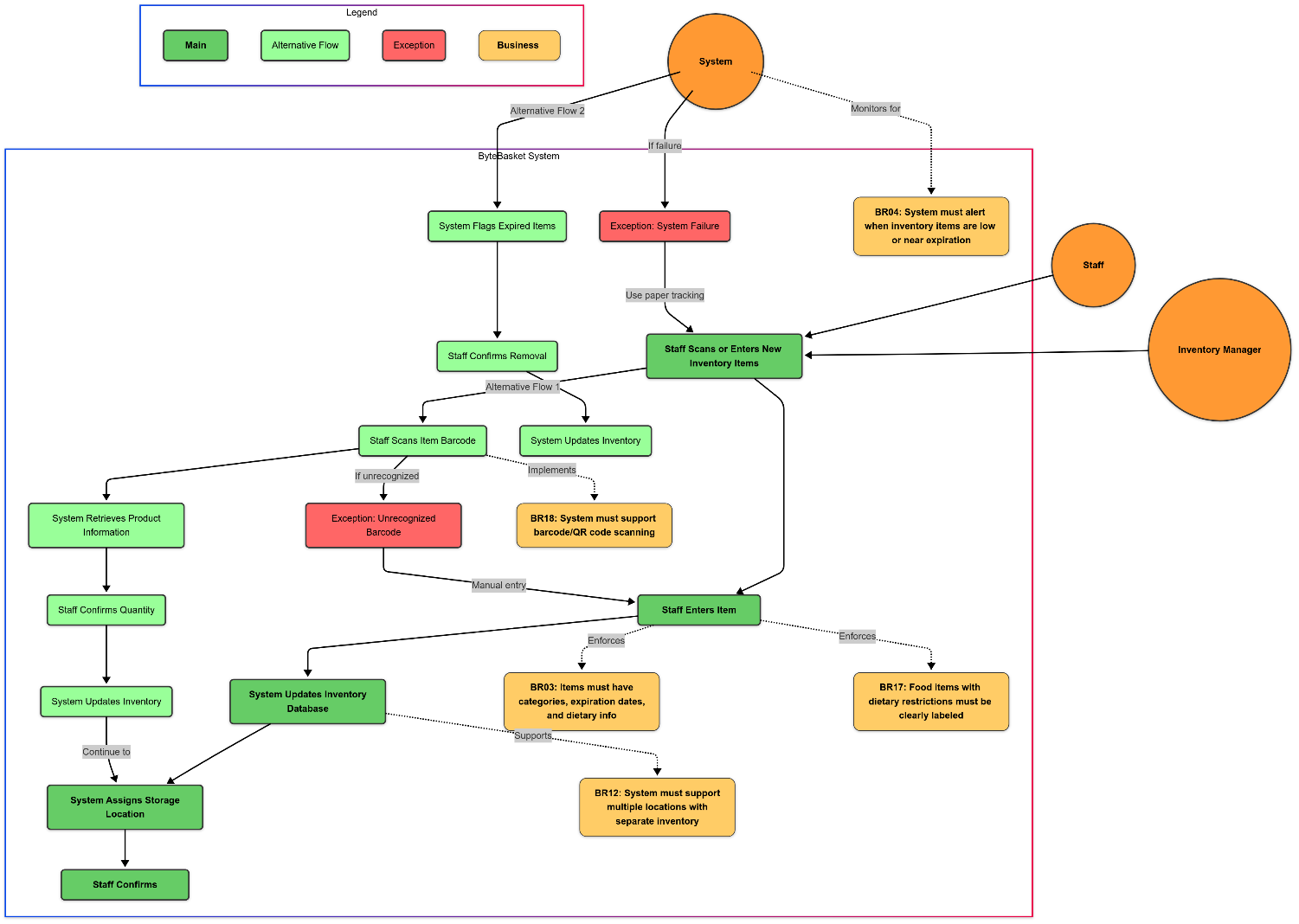
|  |  |
| --- | --- |
| Use Case Identifier | UC02 |
| Use Case Title | Dashboard Navigation |
| Actor(s) | All users |
| Description | This use case describes how users navigate their role-specific dashboards to access ByteBasket features. |
| Trigger | User logs into the system |
| Precondition(s) | User has successfully authenticated |
| Flow of Events | 1. System displays role-specific dashboard 2. User views summary information relevant to their role 3. User selects desired function from navigation menu 4. System redirects to appropriate module |
| Alternative Flow of Events | **User customizes dashboard:**   1. User selects "Customize Dashboard" option 2. User selects widgets and layout preferences 3. System saves and displays personalized dashboard |
| Exception(s) | **System performance issues:** User is notified to try again later |
| Postcondition(s) | User accesses desired system functionality |
| Bussines Rule(s) | BR19 |



|  |  |
| --- | --- |
| Use Case Identifier | UC03 |
| Use Case Title | Food Bank Location Management |
| Actor(s) | Admin |
| Description | This use case describes the process of adding, editing, and managing food bank locations |
| Trigger | Admin needs to add or modify food bank location information |
| Precondition(s) | Admin has logged into the system |
| Flow of Events | 1. Admin navigates to location management section 2. Admin views list of existing locations 3. Admin adds new location with details (name, address, operating hours, contact) 4. System validates and saves the information 5. System displays updated location list |
| Alternative Flow of Events | **Admin edits existing location:**   1. Admin selects location to edit 2. Admin updates location details 3. System saves changes   **Admin deactivates location:**   1. Admin selects location to deactivate 2. Admin confirms deactivation 3. System flags location as inactive |
| Exception(s) | **Invalid address format:** System prompts for correction  **Duplicate location:** System notifies admin |
| Postcondition(s) | 1. Food bank location is added or updated in the system 2. Users can select this location for related operations |
| Bussines Rule(s) | BR12 |



|  |  |
| --- | --- |
| Use Case Identifier | UC04 |
| Use Case Title | Inventory Management |
| Actor(s) | Inventory Manager, Staff |
| Description | This use case describes the tracking, categorizing, and management of food inventory. |
| Trigger | New items arrive or inventory needs to be updated |
| Precondition(s) | 1. User has inventory management permissions |
| Flow of Events | 1. Staff scans or enters new inventory items 2. Staff enters item details (category, expiration, dietary info) 3. System updates inventory database 4. System assigns storage location 5. Staff confirms placement |
| Alternative Flow of Events | **Using barcode scanner:**   1. Staff scans item barcode 2. System retrieves product information 3. Staff confirms quantity 4. System updates inventory   **Removing expired items:**   1. System flags expired items 2. Staff confirms removal 3. System updates inventory |
| Exception(s) | **Unrecognized barcode:** Staff manually enters information  **System failure:** Temporary paper tracking used |
| Postcondition(s) | 1. Inventory is accurately tracked 2. Items are categorized and labeled with dietary information |
| Bussines Rule(s) | BR03, BR04, BR12, BR17, BR18 |



|  |  |
| --- | --- |
| Use Case Identifier | UC05 |
| Use Case Title | Volunteer Coordination |
| Actor(s) | Volunteer Coordinator, Volunteer |
| Description | This use case describes volunteer registration, scheduling, and hour tracking. |
| Trigger | Volunteer signs up or coordinator manages volunteer schedule |
| Precondition(s) | 1. User has volunteer role permissions   Coordinator has appropriate permissions |
| Flow of Events | 1. Volunteer views available shifts 2. Volunteer selects desired shift 3. System records volunteer interest 4. Coordinator approves volunteer request 5. System confirms shift assignment 6. Volunteer logs hours after completing shift 7. Coordinator verifies logged hours |
| Alternative Flow of Events | **Coordinator creates new shifts:**   1. Coordinator adds shift details (date, time, role, requirements) 2. System adds shift to available opportunities   **Volunteer cancels commitment:**   1. Volunteer selects shift to cancel 2. System notifies coordinator 3. Shift becomes available for other volunteers |
| Exception(s) | **Shift conflicts:** System notifies volunteer  **Late cancellation:** System flags according to policy |
| Postcondition(s) | 1. Volunteer shifts are assigned and tracked 2. Volunteer hours are recorded in the system |
| Bussines Rule(s) | BR08, BR15 |

|  |  |
| --- | --- |
| Use Case Identifier | UC06 |
| Use Case Title | Inventory Alerts |
| Actor(s) | Inventory Manager, Admin, Staff |
| Description | This use case describes how the system generates and manages alerts for low stock or expiring items. |
| Trigger | Inventory levels fall below threshold or items approach expiration |
| Precondition(s) | Inventory data is up-to-date in the system |
| Flow of Events | 1. System continuously monitors inventory levels 2. System identifies items below threshold or nearing expiration 3. System generates alerts on dashboard 4. System sends notification to responsible staff 5. Staff acknowledges alert 6. Staff takes appropriate action |
| Alternative Flow of Events | **Adjusting alert thresholds:**   1. Admin navigates to alert settings 2. Admin modifies thresholds for specific items or categories 3. System applies new alert criteria |
| Exception(s) | **Notification failure:** Dashboard still displays alerts  **False positive:** Staff can mark as resolved |
| Postcondition(s) | 1. Staff is aware of inventory issues 2. Actions are taken to address low stock or handle expiring items |
| Bussines Rule(s) | BR04 |

|  |  |
| --- | --- |
| Use Case Identifier | UC07 |
| Use Case Title | Donation Management |
| Actor(s) | Donor, Donation Coordinator |
| Description | This use case describes how donors contribute and track their donations. |
| Trigger | Donor wishes to make a contribution |
| Precondition(s) | Donor has registered account |
| Flow of Events | Donor logs into system  Donor selects "Make Donation" option  Donor enters donation details (items, quantities)  Donor schedules drop-off time  System generates confirmation  At drop-off, staff verifies donation  System generates tax receipt  Donor receives notification |
| Alternative Flow of Events | **Recurring donations:**   1. Donor sets up recurring donation schedule 2. System generates regular reminders 3. Donation process follows normal flow   **Viewing donation history:**   1. Donor selects "Donation History" 2. System displays list of past donations and impact |
| Exception(s) | **Scheduling conflict:** System suggests alternative times  **Rejected items:** Staff documents reason and donor is notified |
| Postcondition(s) | 1. Donation is recorded in the system 2. Inventory is updated 3. Donor receives receipt 4. Impact tracking is updated |
| Bussines Rule(s) | BR05, BR14 |

|  |  |
| --- | --- |
| Use Case Identifier | UC08 |
| Use Case Title | Request Processing |
| Actor(s) | Recipient, Staff |
| Description | This use case describes how recipients request food assistance and how requests are processed. |
| Trigger | Recipient needs food assistance |
| Precondition(s) | 1. Recipient has registered account 2. Recipient has verified identity |
| Flow of Events | 1. Recipient logs into system 2. Recipient views available items 3. Recipient creates wishlist 4. System verifies availability and matches dietary needs 5. Recipient submits request 6. Staff receives notification 7. Staff prepares request 8. Recipient receives notification 9. Recipient picks up food package |
| Alternative Flow of Events | **Items unavailable:**   1. System suggests alternatives 2. Recipient accepts or rejects alternatives 3. Request is updated accordingly   **Recurring requests:**   1. Recipient sets up recurring schedule 2. System processes regular requests |
| Exception(s) | **Verification issues:** Staff assists with verification process  **Request exceeds limits:** System notifies recipient of policy |
| Postcondition(s) | 1. Request is fulfilled 2. Inventory is updated 3. Recipient receives assistance |
| Bussines Rule(s) | BR06, BR07, BR16 |

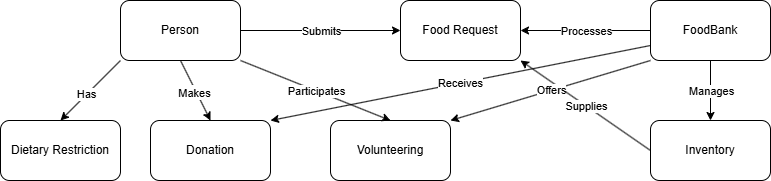
|  |  |
| --- | --- |
| Use Case Identifier | UC09 |
| Use Case Title | Volunteer Hour Tracking |
| Actor(s) | Volunteer, Volunteer Coordinator |
| Description | This use case describes how volunteer hours are logged and verified. |
| Trigger | Volunteer completes shift |
| Precondition(s) | Volunteer has completed assigned shift |
| Flow of Events | 1. Volunteer logs into system 2. Volunteer selects "Log Hours" option 3. Volunteer enters shift details (date, time, duration, activities) 4. System records volunteer input 5. Coordinator receives notification 6. Coordinator reviews and verifies hours 7. System updates volunteer record |
| Alternative Flow of Events | **Hour correction:**   1. Coordinator identifies discrepancy 2. Coordinator adjusts recorded hours 3. Volunteer receives notification 4. Volunteer acknowledges or disputes change |
| Exception(s) | **Log submission after deadline:** Requires special coordinator approval  **System failure:** Hours recorded on paper form |
| Postcondition(s) | 1. Volunteer hours are recorded 2. Volunteer profile is updated with service history |
| Bussines Rule(s) | BR08, BR15 |

|  |  |
| --- | --- |
| Use Case Identifier | UC10 |
| Use Case Title | Reporting |
| Actor(s) | Admin, Staff |
| Description | This use case describes how the system generates reports on inventory, donations, and distribution. |
| Trigger | User needs to generate a report |
| Precondition(s) | User has reporting permissions |
| Flow of Events | 1. User navigates to reporting module 2. User selects report type 3. User defines parameters (date range, categories, locations) 4. System processes request 5. System generates report 6. User views or exports report |
| Alternative Flow of Events | **Scheduled reports:**   1. User sets up recurring report 2. User defines recipients and schedule 3. System automatically generates and distributes reports   **Custom reports:**   1. User selects "Custom Report" option 2. User defines specific metrics and layout 3. System saves report template for future use |
| Exception(s) | **Data processing error:** System notifies user to try again  **Large reports:** System offers background processing option |
| Postcondition(s) | Report is generated and available for viewing or download  Report data accurately reflects system information |
| Bussines Rule(s) | BR13 |

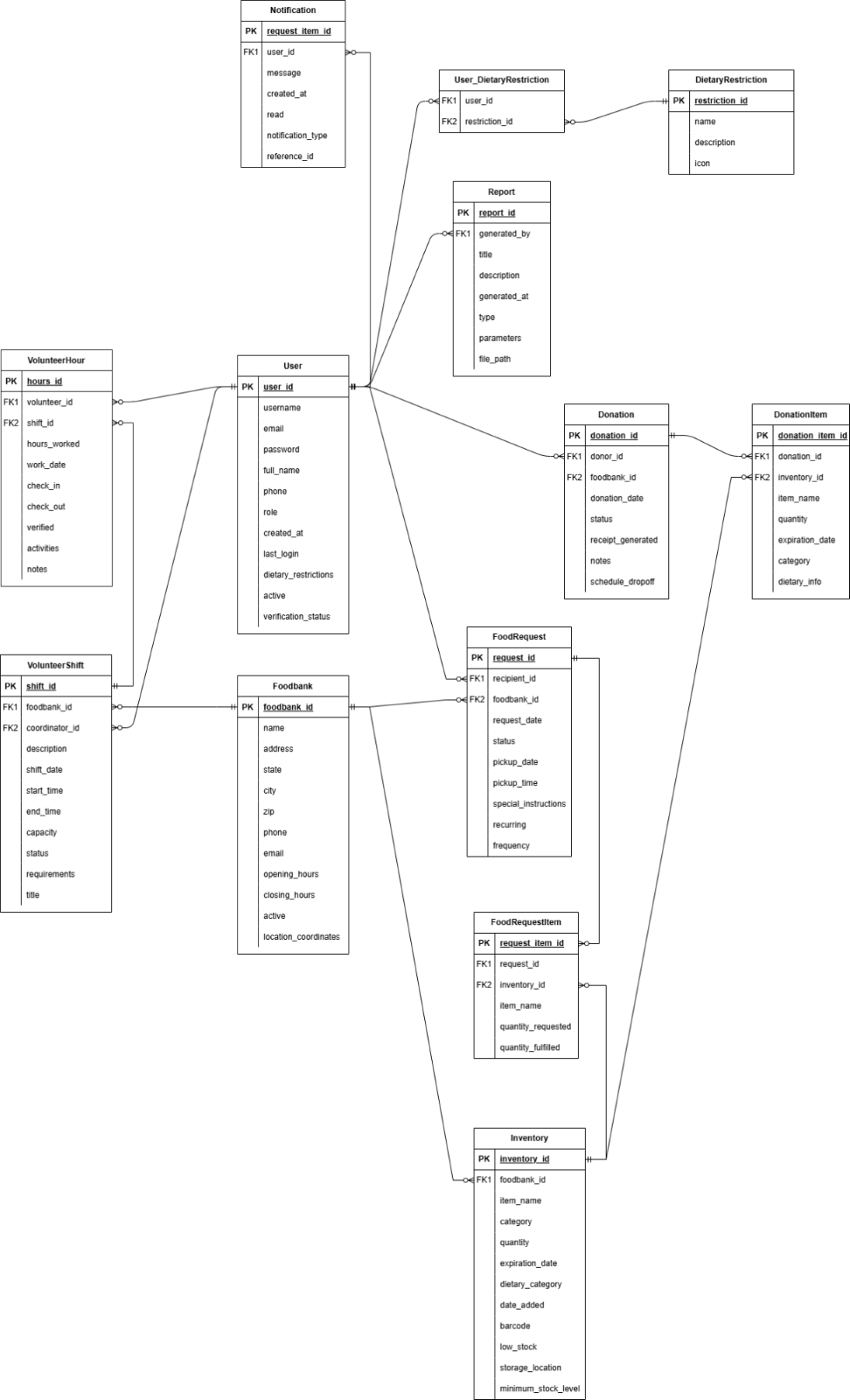
|  |  |
| --- | --- |
| Use Case Identifier | UC11 |
| Use Case Title | Dashboard Customization |
| Actor(s) | All users |
| Description | This use case describes how users can customize their dashboards. |
| Trigger | User wants to customize dashboard view |
| Precondition(s) | User has logged into the system |
| Flow of Events | 1. User selects "Customize Dashboard" option 2. User views available widgets 3. User selects desired widgets 4. User arranges widget layout 5. User saves preferences 6. System displays customized dashboard |
| Alternative Flow of Events | **Resetting to default:**   1. User selects "Reset Dashboard" option 2. System restores default configuration   **Template selection:**   1. User selects from pre-configured dashboard templates 2. System applies template |
| Exception(s) | **Save failure:** System notifies user  **Widget compatibility issues:** System suggests alternatives |
| Postcondition(s) | 1. User dashboard displays customized information 2. Preferences are saved for future sessions |
| Bussines Rule(s) | BR19 |

|  |  |
| --- | --- |
| Use Case Identifier | UC12 |
| Use Case Title | System Administration |
| Actor(s) | System Administrator |
| Description | This use case describes system maintenance, security, and configuration tasks. |
| Trigger | Admin needs to perform system maintenance or configuration |
| Precondition(s) | User has system administrator permissions |
| Flow of Events | Admin logs into administrative console  Admin selects maintenance or configuration task  Admin makes necessary changes  System validates changes  System implements changes  Admin receives confirmation |
| Alternative Flow of Events | **Security monitoring:**   1. Admin reviews security logs 2. Admin investigates suspicious activities 3. Admin takes appropriate action   **Backup management:**   1. Admin schedules backup 2. System performs backup 3. Admin receives confirmation |
| Exception(s) | **Critical system error:** System reverts to previous configuration  **Security breach:** System initiates lockdown protocols |
| Postcondition(s) | 1. System configuration is updated 2. System operates according to specifications |
| Bussines Rule(s) | BR09, BR10, BR20 |

# Domain Class Diagram

(Domain Class Diagram)

# Database

  
(Database design)

1. User Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Constraints | Description |
| user\_id | INT | PK, AUTO\_INCREMENT, NOT NULL | Unique identifier for each user |
| username | VARCHAR(50) | UNIQUE, NOT NULL | Username for login |
| email | VARCHAR(255) | UNIQUE, NOT NULL | User's email address |
| password | VARCHAR(255) | NOT NULL | Hashed password |
| full\_name | VARCHAR(255) | NOT NULL | User's full name |
| phone | VARCHAR(20) | NULL | User's contact number |
| role | ENUM | NOT NULL | Role: 'Admin', 'Staff', 'Donor', 'Recipient', 'Volunteer' |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Account creation date/time |
| last\_login | TIMESTAMP | NULL | Last login date/time |
| dietary\_restrictions | TEXT | NULL | Text description of dietary needs |
| active | BOOLEAN | DEFAULT TRUE | Account status |
| verification\_status | VARCHAR(50) | DEFAULT 'Pending' | User verification status |

2. FoodBank Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Constraints | Description |
| foodbank\_id | INT | PK, AUTO\_INCREMENT, NOT NULL | Unique identifier for each food bank location |
| name | VARCHAR(255) | NOT NULL | Food bank name |
| address | VARCHAR(255) | NOT NULL | Street address |
| city | VARCHAR(100) | NOT NULL | City |
| state | VARCHAR(50) | NOT NULL | State/Province |
| zip | VARCHAR(20) | NOT NULL | Postal/ZIP code |
| phone | VARCHAR(20) | NOT NULL | Contact phone number |
| email | VARCHAR(255) | NULL | Contact email address |
| opening\_hours | TIME | NOT NULL | Opening time |
| closing\_hours | TIME | NOT NULL | Closing time |
| active | BOOLEAN | DEFAULT TRUE | Operational status |
| location\_coordinates | VARCHAR(50) | NULL | Geo-coordinates for mapping |

3. Inventory Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Constraints | Description |
| inventory\_id | INT | PK, AUTO\_INCREMENT, NOT NULL | Unique identifier for inventory item |
| foodbank\_id | INT | FK REFERENCES FoodBank(foodbank\_id), NOT NULL | Associated food bank |
| item\_name | VARCHAR(255) | NOT NULL | Name of food/non-food item |
| category | VARCHAR(100) | NOT NULL | Category (e.g., canned, fresh, dry) |
| quantity | INT | NOT NULL | Current quantity in stock |
| expiration\_date | DATE | NULL | Expiration date if applicable |
| storage\_location | VARCHAR(100) | NULL | Storage location within food bank |
| dietary\_category | ENUM | NULL | Special dietary categories |
| date\_added | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Date item was added to inventory |
| barcode | VARCHAR(50) | NULL | Barcode/UPC if available |
| low\_stock | BOOLEAN | DEFAULT FALSE | Indicates if item is below minimum stock level |
| minimum\_stock\_level | INT | DEFAULT 10 | Threshold for low stock alert |

4. Donation Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Constraints | Description |
| donation\_id | INT | PK, AUTO\_INCREMENT, NOT NULL | Unique identifier for donation |
| donor\_id | INT | FK REFERENCES User(user\_id), NOT NULL | Donor who made contribution |
| foodbank\_id | INT | FK REFERENCES FoodBank(foodbank\_id), NOT NULL | Receiving food bank |
| donation\_date | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Date donation was made/received |
| status | ENUM | NOT NULL | Status: 'Scheduled', 'Received', 'Processed' |
| receipt\_generated | BOOLEAN | DEFAULT FALSE | Whether tax receipt was generated |
| notes | TEXT | NULL | Additional information about donation |
| scheduled\_dropoff | DATETIME | NULL | Scheduled date/time for dropoff |

5. DonationItem Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Constraints | Description |
| donation\_item\_id | INT | PK, AUTO\_INCREMENT, NOT NULL | Unique identifier for donation item |
| donation\_id | INT | FK REFERENCES Donation(donation\_id), NOT NULL | Associated donation |
| inventory\_id | INT | FK REFERENCES Inventory(inventory\_id), NULL | Linked inventory item if applicable |
| item\_name | VARCHAR(255) | NOT NULL | Name of donated item |
| quantity | INT | NOT NULL | Quantity donated |
| expiration\_date | DATE | NULL | Expiration date if applicable |
| category | ENUM | NOT NULL | Category: 'Canned', 'Dry', 'Fresh', 'Personal' |
| dietary\_info | VARCHAR(255) | NULL | Dietary information |

6. FoodRequest Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Constraints | Description |
| request\_id | INT | PK, AUTO\_INCREMENT, NOT NULL | Unique identifier for food request |
| recipient\_id | INT | FK REFERENCES User(user\_id), NOT NULL | Recipient requesting food |
| foodbank\_id | INT | FK REFERENCES FoodBank(foodbank\_id), NOT NULL | Food bank processing request |
| request\_date | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Date request was submitted |
| status | ENUM | DEFAULT 'Pending' | Status: 'Pending', 'Approved', 'Ready', 'Fulfilled' |
| pickup\_date | DATE | NULL | Scheduled pickup date |
| pickup\_time | VARCHAR(50) | NULL | Scheduled pickup time |
| special\_instructions | TEXT | NULL | Special instructions for preparation |
| recurring | BOOLEAN | DEFAULT FALSE | Whether this is a recurring request |
| frequency | VARCHAR(50) | NULL | Frequency of recurring requests |

7. FoodRequestItem Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Constraints | Description |
| request\_item\_id | INT | PK, AUTO\_INCREMENT, NOT NULL | Unique identifier for requested item |
| request\_id | INT | FK REFERENCES FoodRequest(request\_id), NOT NULL | Associated food request |
| inventory\_id | INT | FK REFERENCES Inventory(inventory\_id), NOT NULL | Requested inventory item |
| item\_name | VARCHAR(255) | NOT NULL | Name of requested item |
| quantity\_requested | INT | NOT NULL | Quantity requested |
| quantity\_fulfilled | INT | DEFAULT 0 | Quantity actually provided |

8. VolunteerShift Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Constraints | Description |
| shift\_id | INT | PK, AUTO\_INCREMENT, NOT NULL | Unique identifier for volunteer shift |
| foodbank\_id | INT | FK REFERENCES FoodBank(foodbank\_id), NOT NULL | Food bank location for shift |
| title | VARCHAR(255) | NOT NULL | Title/name of shift |
| description | TEXT | NULL | Description of tasks/responsibilities |
| shift\_date | DATE | NOT NULL | Date of shift |
| start\_time | TIME | NOT NULL | Start time for shift |
| end\_time | TIME | NOT NULL | End time for shift |
| capacity | INT | DEFAULT 1 | Maximum number of volunteers needed |
| status | ENUM | DEFAULT 'Open' | Status: 'Open', 'Filled', 'Completed' |
| requirements | TEXT | NULL | Special requirements or skills needed |
| coordinator\_id | INT | FK REFERENCES User(user\_id), NOT NULL | Staff coordinating this shift |

9. VolunteerHour Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Constraints | Description |
| hours\_id | INT | PK, AUTO\_INCREMENT, NOT NULL | Unique identifier for volunteer hours record |
| volunteer\_id | INT | FK REFERENCES User(user\_id), NOT NULL | Volunteer who worked |
| shift\_id | INT | FK REFERENCES VolunteerShift(shift\_id), NOT NULL | Related volunteer shift |
| hours\_worked | DECIMAL(5,2) | NOT NULL | Number of hours worked |
| work\_date | DATE | NOT NULL | Date hours were worked |
| check\_in | TIME | NULL | Time volunteer checked in |
| check\_out | TIME | NULL | Time volunteer checked out |
| verified | BOOLEAN | DEFAULT FALSE | Whether hours have been verified |
| activities | TEXT | NULL | Description of activities performed |
| notes | TEXT | NULL | Additional notes |

10. Notification Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Constraints | Description |
| notification\_id | INT | PK, AUTO\_INCREMENT, NOT NULL | Unique identifier for notification |
| user\_id | INT | FK REFERENCES User(user\_id), NOT NULL | User receiving notification |
| message | TEXT | NOT NULL | Notification content |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Creation date/time |
| read | BOOLEAN | DEFAULT FALSE | Whether notification has been read |
| notification\_type | VARCHAR(50) | NOT NULL | Type of notification |
| reference\_id | VARCHAR(50) | NULL | Reference to related record |

11. Report Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Constraints | Description |
| report\_id | INT | PK, AUTO\_INCREMENT, NOT NULL | Unique identifier for report |
| title | VARCHAR(255) | NOT NULL | Report title |
| description | TEXT | NULL | Report description |
| generated\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Generation date/time |
| generated\_by | INT | FK REFERENCES User(user\_id), NOT NULL | User who generated report |
| type | ENUM | NOT NULL | Type: 'Inventory', 'Donation', 'Distribution', 'Volunteer' |
| parameters | TEXT | NULL | Parameters used for report generation |
| file\_path | VARCHAR(255) | NULL | Path to saved report file |

12. DietaryRestriction Table

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Constraints | Description |
| restriction\_id | INT | PK, AUTO\_INCREMENT, NOT NULL | Unique identifier for dietary restriction |
| name | VARCHAR(100) | UNIQUE, NOT NULL | Name of restriction (e.g., 'Gluten-Free') |
| description | TEXT | NULL | Detailed description |
| icon | VARCHAR(255) | NULL | Icon/symbol to represent restriction |

13. User\_DietaryRestriction Table (Junction Table)

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Constraints | Description |
| user\_id | INT | FK REFERENCES User(user\_id), NOT NULL | User with dietary restriction |
| restriction\_id | INT | FK REFERENCES DietaryRestriction(restriction\_id), NOT NULL | Associated restriction |
| PRIMARY KEY |  | (user\_id, restriction\_id) | Composite primary key |

Work Breakdown Structure (WBS)

## Work Breakdown Structure

Sample WBS:

Diagram

Description automatically generated

# Implementation Schedule

## Product Backlogs:

**User Authentication & Access Control**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **User Story** | **Priority** | **Acceptance Criteria** | **Dependencies** | **Effort (SP)** |
| **PB-001** | As a user, I want to register and log in so that  I can access the system. | High | Valid email/password. Role assignment.  Verification email sent. Password hashed. | Email system | 5 |
| **PB-002** | As an admin, I want to manage user  roles and permissions. | High | Assign/update/revoke roles. Role-based access to system features. | PB-001 | 5 |
| **PB-003** | As a user, I want to manage my profile including dietary  preferences. | Medium | Edit profile details. Set dietary restrictions. | PB-001 | 3 |
| **PB-004** | As a user, I want to reset my  password if I forget it. | High | “Forgot Password” option. Email link to reset securely. | Email system | 5 |

**Inventory Management**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **User Story** | **Priority** | **Acceptance**  **Criteria** | **Dependencies** | **Effort (SP)** |
| **PB-005** | As a staff member, I want to add new food items to the inventory. | High | Item type, expiration, dietary info recorded. Barcode scanning  supported. | PB-001 | 8 |
| **PB-006** | As a system, I want to generate alerts for low or  expiring | High | Dashboard alerts and notifications for threshold  breaches. | PB-005 | 5 |
| PB-007 | As a user, I want to view inventory  status per location. | Medium | Filter by location. Real-time updates. | PB-005 | 5 |

**Donation Management**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **User Story** | **Priority** | **Acceptance**  **Criteria** | **Dependencies** | **Effort (SP)** |
| **PB-008** | As a donor, I want to make and track donations. | High | Enter item details.  Schedule drop-off. Get receipt.  View donation  history. | PB-001 | 8 |
| **PB-009** | As a system, I want to generate automated donation  receipts. | High | Receipt generated post verification. Sent via  email. | PB-008 | 3 |

**Request Processing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **User Story** | **Priority** | **Acceptance Criteria** | **Dependencies** | **Effort (SP)** |
| **PB-010** | As a recipient, I want to request food based on my  dietary needs. | High | Wishlist feature. Matched to dietary data. Submit &  track request. | PB-003, PB- 005 | 8 |
| **PB-011** | As a staff member, I want to view and fulfill recipient requests. | High | Notifications for new requests.  Inventory auto- updated  after fulfillment. | PB-010 | 5 |

**Volunteer Coordination**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **User Story** | **Priority** | **Acceptance**  **Criteria** | **Dependencies** | **Effort (SP)** |
| **PB-012** | As a volunteer, I want to register and sign up for  shifts. | Medium | Available shifts visible. Auto- confirm or coordinator  approval. | PB-001 | 5 |
| **PB-013** | As a coordinator, I want to manage volunteer schedules and verify  hours. | High | Add/edit shifts. View logs.  Approve or reject hours. | PB-012 | 8 |

**Reporting & Analytics**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **User Story** | **Priority** | **Acceptance Criteria** | **Dependencies** | **Effort (SP)** |
| **PB-014** | As an admin, I want to view inventory and request fulfillment  reports. | Medium | Generate by category/date/location. Export to PDF/Excel. | PB-005, PB- 010 | 8 |
| **PB-015** | As a donor, I want to see the impact of  my donations. | Medium | Donation-to-request match rate. Summary view. | PB-008, PB- 014 | 3 |

**System Administration**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **User Story** | **Priority** | **Acceptance Criteria** | **Dependencies** | **Effort (SP)** |
| **PB-016** | As a system admin, I want to configure roles,  security, and backups. | High | Role-based access. Daily backups.  Audit logs. Password policies  enforced. | PB-002 | 8 |
| **PB-017** | As a system admin, I want to view system uptime and  performance logs. | Medium | 99.9% uptime ensured. Logs downloadable. | PB-016 | 5 |

**UI/UX and Accessibility**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **User Story** | **Priority** | **Acceptance**  **Criteria** | **Dependencies** | **Effort (SP)** |
| **PB-018** | As a user, I want a mobile- responsive and intuitive  interface. | High | Interface works across devices. Max 3-click  navigation. | All | 8 |
| **PB-019** | As a user, I want support for multiple  languages. | Medium | English, French, and Spanish supported. | PB-018 | 5 |

**Additional Features**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **User Story** | **Priority** | **Acceptance**  **Criteria** | **Dependencies** | **Effort (SP)** |
| **PB-020** | As an admin, I want to add/edit/manage food bank locations. | Medium | Add/edit/delete location info.  Select location during ops.  Mark as inactive if needed. | PB-002 | 5 |
| **PB-021** | As a user, I want to receive notifications for actions like requests, shifts, or donations. | Medium | Notifications for request updates, expiring items, volunteer shifts,  donation confirmation. | PB-006, PB- 008, PB-011, PB-013 | 8 |

Project Phases

**Phase 1:** Project Setup and Core Infrastructure (Weeks 1-4)

* Set up development environment
* Configure CI/CD pipeline
* Establish Git workflow
* Create database schema
* Set up cloud hosting environment
* Implement authentication system
* Create basic UI framework with responsive design

**Phase 2:** Core Functionality (Weeks 5-12)

* User management system
* Food bank location management
* Basic inventory management
* Simple donation tracking
* Basic food request system
* Dashboard views for different user roles

**Phase 3:** Advanced Features (Weeks 13-20)

* Advanced inventory management with barcode scanning
* Dietary restriction tracking
* Volunteer coordination
* Donation receipt generation
* Wishlist functionality
* Notification system

**Phase 4:** Reporting, Analytics & Optimization (Weeks 21-28)

* Reporting system implementation
* Data analytics dashboard
* Performance optimization
* Security hardening
* Multi-language support
* Mobile responsiveness refinement

**Phase 5:** Testing, Documentation & Launch (Weeks 29-32)

* Comprehensive testing
* User documentation creation
* Staff training materials
* Final bug fixes
* Production deployment
* Post-launch support

## Sprint Planning:

**Sprint 1** (Weeks 1-2): Project Setup

* Environment setup
* Database creation
* Repository structure
* Authentication framework selection
* Initial UI wireframe implementation

**Sprint 2** (Weeks 3-4): User Management

* User registration & login
* Role-based access control
* Profile management
* Admin user management interface
* Initial dashboard views

**Sprint 3** (Weeks 5-6): Food Bank & Inventory Basics

* Food bank location management
* Basic inventory CRUD operations
* Inventory categorization system
* Storage location tracking
* Inventory search functionality

**Sprint 4** (Weeks 7-8): Basic Donation Management

* Donor interface
* Donation recording system
* Donation history tracking
* Linking donations to inventory

Basic receipt generation

**Sprint 5** (Weeks 9-10): Basic Request Processing

* Recipient interface
* Food request creation
* Request status tracking
* Request history view
* Basic request fulfillment workflow

**Sprint 6** (Weeks 11-12): Dashboard Enhancements

* Role-specific dashboards
* Real-time inventory levels
* Recent activity tracking
* Basic notifications
* Dashboard customization options

**Sprint 7** (Weeks 13-14): Advanced Inventory Features

* Barcode/QR scanning implementation
* Expiration date tracking
* Low stock alerts
* Inventory movement history
* Inventory reports

**Sprint 8** (Weeks 15-16): Dietary Restrictions & Allergies

* Dietary restriction management
* Allergen tracking
* Item labeling system
* Matching algorithm for requests
* Dietary preference profiles

**Sprint 9** (Weeks 17-18): Volunteer System

* Volunteer registration
* Shift management
* Hour tracking
* Volunteer coordinator interface
* Volunteer reports

**Sprint 10** (Weeks 19-20): Enhanced Donation Features

* Scheduled donations
* Recurring donations
* Advanced receipt options
* Donor impact dashboard
* Corporate donor features

**Sprint 11** (Weeks 21-22): Wishlist & Advanced Requests

* Wishlist creation
* Preferred items tracking
* Recurring requests
* Special instructions handling
* Request prioritization

**Sprint 12** (Weeks 23-24): Notification System

* Email notifications
* In-app notifications
* SMS integration (optional)
* Notification preferences
* Automated alerts

**Sprint 13** (Weeks 25-26): Reporting System

* Custom report generation
* Standard reports implementation
* Data export functionality
* Report scheduling
* Visual data presentation

**Sprint 14** (Weeks 27-28): Optimization & Security

* Performance optimization
* Security audit & hardening
* Scalability testing
* Load testing
* Database optimization

**Sprint 15** (Weeks 29-30): Testing & Documentation

* User testing
* Comprehensive QA
* User documentation
* Admin documentation
* Training materials creation

**Sprint 16** (Weeks 31-32): Launch Preparation

* Final bug fixes
* Production environment setup
* Launch strategy
* Post-launch support plan

# Client / Faculty Sign-off

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

X .

Name of Client/Rep/Professor