Acme Food Bank Inventory Tracking System

Project Requirements and Specifications

Acme Corporation

Yaru Gao;

Dante Meyer;

May 24, 2024

TABLE OF CONTENTS

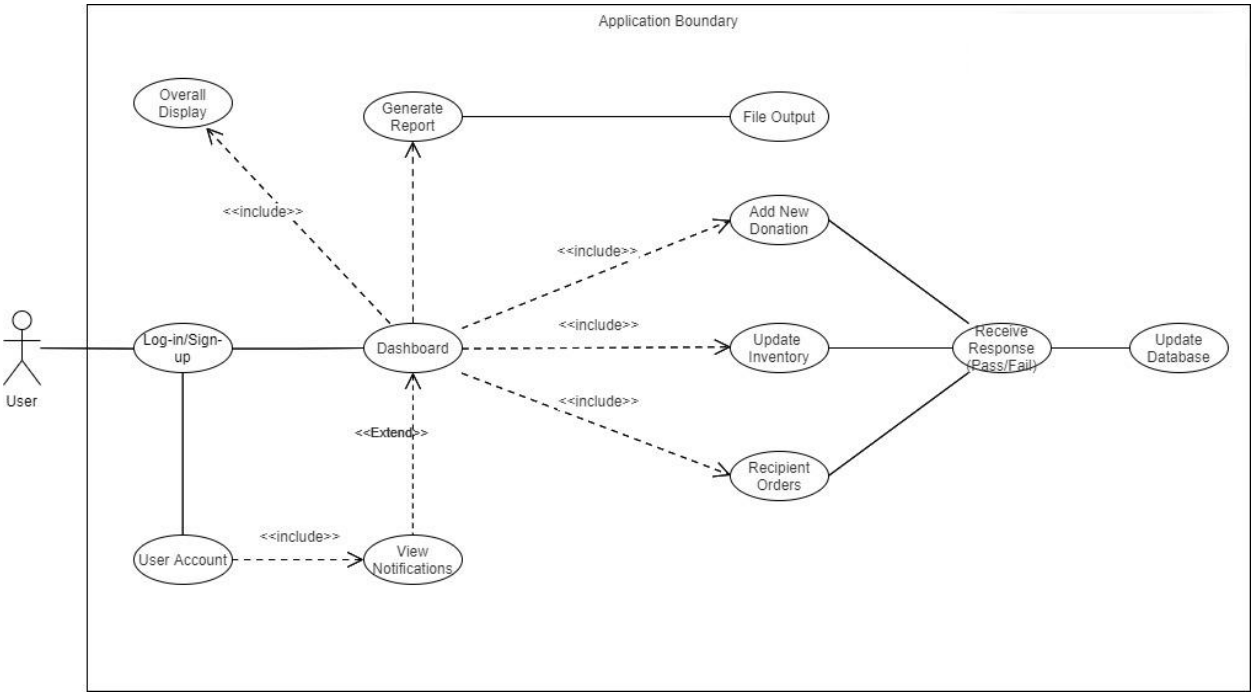
I. Introduction

The Acme Food Bank faces significant challenges in managing frequent and varied donations of food and hygiene products. Manual data management and entry are difficult due to the volume of the product entries. This process is less efficient, not always timely, and prone to errors, requiring significant effort to ensure accurate tracking of quantities and expiration dates.

Our project aims to develop a reliable inventory tracking system to address these challenges. By utilizing a combination of Python, Flask, HTML, and SQLite, we will create a web application that allows users to easily interact with a centralized database. This system will facilitate real-time updates, provide secure user authentication, and generate detailed reports to enhance the operational efficiency of the Acme Food Bank.

II. System Requirements Specification

II.1. Use Cases



[View Dashboard]
Pre-condition: User is logged in.
Post-condition: The user can see an overview of inventory levels, recent donations, and upcoming expirations.
Basic Path:

1. User logs into the system.
2. User will be automatically navigated to the dashboard page.

Alternative Path: The user can also access the dashboard page by clicking on the “Dashboard” button on taskbar.
Related Requirements:

- Overview Display
- Key Metrics

- Inventory Status

[Add New Donation]
Pre-condition: User is logged in and has appropriate permissions.
Post-condition: The new donation is added to the inventory.
Basic Path:
1. User logs into the system.
2. User navigates to the "Add Donation" page.
3. User enters details of the donation.
4. User submits the donation and the data is saved.

System updates the inventory with the new donation.
Alternative Path:
- At the 4th step, if the user failed to submit the donation or refreshed the page, the user returns to the 2rd step.

Related Requirements:
- Inventory Management
- Real-time Updates

[Update Inventory]
Pre-condition: User is logged in and has appropriate permissions.
Post-condition: The inventory is updated with new information.
Basic Path:
1. User logs into the system.
2. User navigates to the "Update Inventory" page.
3. User selects the item to update.
4. User modifies the necessary details.
5. User submits the changes.
6. System updates the inventory.

Alternative Path:
- At the 5th step, if the user failed to submit or refreshed the page, the user returns to the 2rd step.

Related Requirements:
- Inventory Management
- Real-time Updates

[Generate Report]
Pre-condition: User is logged in and has appropriate permissions.
Post-condition: A customized report is generated and available for export.
Basic Path:
1. User logs into the system.
2. User navigates to the "Generate Report" page.
3. User selects the type of report and desired parameters.
4. User submits the request for report generation.
5. System generates the report and provides options to export.

Alternative Path:
- At the 4th step, if the user failed to submit or refreshed the page, the user returns to the 2rd step.

Related Requirements:
- Reporting
- Monthly Analysis

[Receive Notifications]
Pre-condition: Items in inventory are nearing expiration.
Post-condition: User is notified of items nearing expiration.
Basic Path:
1. System monitors the inventory for items nearing expiration.
2. System sends alerts/notifications to the user about upcoming expirations.
3. User views the notifications.

Alternative Path: None
Related Requirements:
- Notifications
- Expiration Alerts
- Highlighting Expirations

[Recipient Orders]
Pre-condition: Recipient has access to the system.
Post-condition: Recipient's demand request is recorded in the system.
Basic Path:
1. User logs into the system with the recipient identity.
2. User navigates to the "Place Order" page.
3. User enters the details of their demand request.
4. User submits the request.
5. System receives the request and outputs the report file.

Alternative Path:
- At the 4th step, if the user failed to submit or refreshed the page, the user returns to the 3rd step.

Related Requirements:
- Recipient Orders
- Demand Requests
- Transaction Processing

II.2. Functional Requirements

II.2.1. [Dashboard]

[Overview Display]: [The dashboard must provide an overview of inventory levels, recent donations, and upcoming expirations.]
Source: [Food bank staff need to monitor and manage inventory efficiently.]
Priority: Priority Level 0: Essential and required functionality

[Inventory Status]: [Users must be able to see the status of the inventory, including total items in stock, categories of items, item location, brand name, purchase date, recipient's info, and demand, etc..]
Source: [Food bank staff and recipients require detailed inventory tracking.]
Priority: Priority Level 0: Essential and required functionality

[Search Functionality]: [The dashboard must feature a search functionality to allow users to easily find specific items or categories.]
Source: [Food bank staff, donors, and recipients need efficient item retrieval.]
Priority: Priority Level 1: Desirable functionality

[Recent Donations]: [The dashboard must highlight recent donations, showing donor names, donation dates, and items donated.]
Source: [Donor tracking requirement.]
Priority: Priority Level 2: Extra features or stretch goals

[Key Metrics]: [The dashboard must display key metrics using graphs, charts, and tables.]
Source: [Food bank staff need visual data representation for effective management.]
Priority: Priority Level 2: Extra features or stretch goals

II.2.2. [Account Data Storage and User Authentication]

[Secure Data Storage]: [Account data must be stored in the SQLite database with TLS encryption to ensure secure data transmission and other sensitive information.]
Source: [IT support staff require data protection.]
Priority: Priority Level 0: Essential and required functionality

[Secure Login]: [The system must provide secure login with two-factor authentication (2FA) to protect user accounts and sensitive information.]
Source: [IT support staff require user data security.]
Priority: Priority Level 0: Essential and required functionality

[Registration]: [The system must allow users to register for new accounts through a registration process.]
Source: [IT support staff require secure user management.]
Priority: Priority Level 0: Essential and required functionality

II.2.3. [Inventory Management]

[CRUD Operations]: [The system must support CRUD operations for adding new donations, updating existing inventory, and marking or removing items that have been distributed or expired.]
Source: [Food bank staff need efficient inventory management.]
Priority: Priority Level 0: Essential and required functionality

[Real-time Updates]: [Using Python, the system must support real-time updates to maintain accurate and current inventory records.]
Source: [Food bank staff need up-to-date inventory data.]
Priority: Priority Level 0: Essential and required functionality

II.2.4. [Reporting]

[Customized Reports]: [The system must generate customized reports based on data from multiple entities, with options to export reports in various formats (e.g., PDF, CSV).]
Source: [Food bank management needs comprehensive reporting.]
Priority: Priority Level 1: Desirable functionality

[Monthly Analysis]: [The system must record and integrate related data about inventory from the database to generate reports required for monthly analysis.]
Source: [Management requirement for periodic data analysis.]
Priority: Priority Level 2: Extra features or stretch goals

II.2.5. [Recipient Orders]

[Demand Requests]: [The system must allow recipients to send their demand requests similar to purchase orders.]
Source: [Recipients' demand of specific products.]
Priority: Priority Level 1: Desirable functionality

[Transaction Processing]: [This information must be stored in the database for transaction processing and data analysis.]
Source: [Food bank staff need to process recipient requests.]
Priority: Priority Level 1: Desirable functionality

II.2.6. [Notifications]

[Expiration Alerts]: [The system must provide alerts and notifications for items nearing their expiration dates to ensure timely distribution.]
Source: [Recipients rely on timely and unexpired products.]
Priority: Priority Level 1: Desirable functionality

[Highlighting Expirations]: [Upcoming expirations must be highlighted with marks (e.g., color-coded alerts) to draw attention to items that need to be distributed soon.]
Source: [Food bank staff's operational need of visual alerts for expiring items.]
Priority: Priority Level 2: Extra features or stretch goals

II.3. Non-Functional Requirements

[Testing]:
Functions that handle basic operations (Writing to the database, login systems, etc.) should have a number of test cases to ensure stability.

[Response time]:
The web interface should load within a reasonable timeframe for convenience. Database operations also should not take long.

[Secure file permissions]:
Certain files on the server (such as the databases, all python scripts, HTML templates, and environment files) should not be accessible via the client, by using read/write protection and not allowing for routing to them via Flask.

[Storage space]:
Not much storage space is necessary, but there should be enough room for a few database files and Python scripts, in addition to python, flask, and other required packages.

[Open Source]:
The code for the project is open sourced on Github, freely available to download, view, and modify. This provides an extra layer of transparency and security, as it assures the program does not contain malicious code.

[Privacy]:
Without explicit port-forwarding, the inventory system should be not accessible outside the food bank's local network, and connections shouldn't be interceptable due to TLS

encryption. Account compromise is also more difficult thanks to 2-factor authentication and password encryption.

[Usability]:

Measures should be taken to create a more usable and intuitive interface and experience for both the frontend program and backend management, such as concise menus and verbose terminal output.

III. System Evolution

Due to the program running on Python and using a web interface, the software stack is very portable, as there are releases of Python for Windows, Linux, and MacOS for multiple types of processors. Because of this, the only requirements are that it runs on any basic computer with at least a modest amount of storage, memory, processing power, and an internet connection. SQLite database files can range from a few kilobytes for most use cases to terabytes [1] for extremely large-scale projects, but as this is a food bank, the need for a storage expansion or to split the data into multiple files is very unlikely.

In the future, it may be more convenient to transition to a different codebase, using Flask to power a REST API that is accessed by a more frontend-oriented framework such as Vue, Angular, Laravel, or React. This would allow for easier user-interface development, independent from the more data-centered backend, but as a downside it would require running two different server backends in separate languages, which can be more complex to troubleshoot. A good way to prepare for this would be to compartmentalize the data-modifying code in Flask to different functions, which are then utilized by the HTML frontend routes so that the code can be easily severed from them in the future and recycled for a JSON-based API.

In case of abrupt hardware and software issues, a useful countermeasure would be to log all the program operations to a text file, so troubleshooters can more easily figure out what in the program went wrong and how to fix it. Database backups should also be made at a regular schedule or per change to easily roll back catastrophic or accidental data modifications, or to restore it in case of accidental deletion. Furthermore, users should also have the option to reset their password if forgotten.

IV. Glossary

2FA (Two-Factor Authentication): An extra layer of security used to ensure that people trying to gain access to an online account are who they say they are.

TLS (Transport Layer Security): A cryptographic protocol designed to provide communications security over a computer network.

Web/Mobile App Development: The process of creating applications for mobile devices and web platforms.

User Interface (UI): The space where interactions between humans and machines occur, aimed at effective operation and control of the machine.

CRUD Operations: CRUD: An acronym for Create, Read, Update, Delete, representing the four basic operations performed on database records or data in a software application.

V. References

[1] SQLite Consortium, "Implementation Limits for SQLite," 3 January 2024. [Online]. Available: https://www.sqlite.org/limits.html [Accessed 24 May 2024].