Acme Food Bank Inventory Tracking System

Project Testing and Acceptance Plan

Acme Corporation



760bf1b0-aa9c-4c1e-acfb-06eb2cac77ca**T-Nine**

Yaru Gao;

Dante Meyer;

**TABLE OF CONTENTS**

**[I.](#_gjdgxs)****Introduction 3**

[I.1.](#_30j0zll) Project Overview 3

[I.2.](#_1fob9te) Test Objectives and Schedule 3

[I.3.](#_3znysh7) Scope 3

**[II.](#_2et92p0)****Testing Strategy 4**

**[III.](#_tyjcwt)****Test Plans 4**

[III.1.](#_3dy6vkm) Unit Testing 4

[III.2.](#_1t3h5sf) Integration Testing 4

[III.3.](#_4d34og8) System Testing 5

[III.3.1.](#_2s8eyo1) Functional testing: 5

[III.3.2.](#_3rdcrjn) Performance testing: 5

[III.3.3.](#_26in1rg) User Acceptance Testing 5

**[IV.](#_lnxbz9)****Environment Requirements 5**

**[V.](#_1ksv4uv)****Glossary 5**

**[VI.](#_44sinio)****References 6**

# Introduction

## Project Overview

The Acme Food Bank Inventory Tracking System is a web application designed to streamline the management of food and hygiene product donations. This system addresses the inefficiencies and errors of manual data entry by providing a centralized database for real-time inventory management. It supports essential functions such as CRUD operations on inventory items, donation entries, transaction orders, and user account management. Additionally, it offers features for generating detailed inventory reports and secure user authentication.

## Test Objectives and Schedule

The main objective will focus on validating the core functionality of the Acme Food Bank Inventory Tracking System. The approach involves conducting the three main software testing stages, including unit testing, integration testing, and system testing [1]. Unit testing will verify individual components for correct functionality. Integration testing will ensure that these components work together. System testing will validate the complete system against the specified requirements.

The testing process requires access to the code and any usable testing tool for manual testing, with all necessary dependencies installed. Specific environmental requirements, including detailed configurations and dependencies, will be discussed in later sections of this document.

The testing schedule for the application includes several key activities. Necessary work activities for the upcoming weeks involve setting up the test environment, developing test cases, executing functional and non-functional tests, identifying and documenting bugs, and retesting to verify fixes. The products to be delivered by the final sprint deadline include tested and debugged code, updated documentation, and a container with the application setup. Major milestones for the final sprint include the completion of test cases, test execution, and final verification of fixes.

## Scope

This document outlines the test plan for the application. It defines the objectives, required resources, and schedule of activities for the testing phase, ensuring an effective approach to verify the application's functionality. The purpose of this document is to guide the testing process to guarantee the usability, reliability and effectiveness of the system in managing inventory.

# Testing Strategy

The overall testing approach includes unit testing, integration testing, and system testing, as mentioned in the previous section. Integration/Continuous Delivery (CI/CD) will also be used, ensuring early issue detection and quick deployment of updates.

Flow of Testing Process:

1. Identify testable requirements from the Software Requirements Specification.

2. Create detailed test cases specifying inputs, expected outputs, and conditions.

3. Prepare necessary hardware, software, and network configurations.

4. Run the test cases using CI/CD pipelines and the aforementioned testing approach, document the results, and note any deviations from expected outcomes.

5. Report bugs, track their resolution, and perform retests to ensure that fixes are effective.

6. Conduct a final round of testing to ensure readiness for deployment.

# Test Plans

## Unit Testing

The primary goal of unit testing is to take the smallest unit of testable software in the application, isolate it from the remainder of the code, and test it for bugs and unexpected behavior.

Different groups of tests will take place in their own Python script files, likely managed by a single central testing file that can be run at any time.

* Login system: Simulate a user logging in via internal code or an HTTP request, verify that only the correct credentials work, there are no exceptions, and make sure restricted pages and features are inaccessible.
* Registration: Simulate a user signing up via internal code or an HTTP request, verify that only valid data types can be used for registration, and that missing inputs are handled properly. Make sure the new user exists on the database, and that logging them in works properly.
* Inventory management: Simulate adding an item to the inventory, verify that data types are restricted, oversized files don’t result upload, and that only real image files can be used for a product image. Check the DB for the correct changes.
* 2FA: Make sure only the correct 2FA code works, and that it can be disabled properly.
* Orders: Make sure filing orders correctly modifies the inventory, and that the order form only accepts proper inputs.

## Integration Testing

Integration testing detects faults that have not been detected during unit testing by focusing on small groups of components. Two or more components are integrated and tested, and when no new faults are revealed, additional components are added to the group.

* Try making orders on a variety of inventory items, and monitor the inventory closely
* Try adding a variety of items to the inventory under different conditions, be sure it affects orders properly
* Modify user data and be sure that the website is still intact (to prevent code injection)

## System Testing

System testing is a type of black box testing that tests all the components together, seen as a single system to identify faults with respect to the scenarios from the overall requirements specifications. Entire system is tested as per the requirements.

During system testing, several activities are performed:

## Functional testing:

Test of functional requirements (from requirements specification). The goal is to select those tests that are relevant to the user and have a high probability of uncovering a failure.

* Inventory/Order modifications, covering different types of edge cases along with normal use.
* Account modification, logging in and out.

## Performance testing:

Performance tests check whether the nonfunctional requirements and additional design goals from the design document are satisfied. In stress testing, system is stressed beyond its specifications to check how and when it fails.

* Making a large amount of requests and modifications to the database to check the programs capacity and speed.
* Making requests with large amounts of data at a time.
* Increasing the amount of items in the database to see how much that impacts performance.

## User Acceptance Testing:

Acceptance testing and installation testing check the system against the project agreement. The purpose is to confirm that the system is ready for operational use. During acceptance test, end-users (customers) of the system compare the system to its initial requirements (if necessary) with help by the developers.

* Check that all files are in the correct directories
* Make sure the server is able to run and function properly on any device or environment.

## III.4. Test Result

Navigation Bar

**Description:**

This is a series of manual test cases for selecting the navigation bar.

**Assumptions:**

- The user is currently logged in.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Expected Inputs | Expected Output | Actual Output | Pass/Fail |
| 01 | Click on the link to other HTML pages | The system redirects user to the respective page. | The system redirects user to the respective page. | P |
| 02 | Click on login button | The login modal opens at the center of the page and allows the user to login or signup. | The login modal opens at the center of the page and allows the user to login or signup. | P |
| 03 | Click on signup link in the log-in modal | The signup modal opens and allows user to register for an account. | The signup modal opens and allows user to register for an account. | P |
| 04 | Click on Avatar after logged in. | The system redirects user to account center. | The system redirects user to account center. | P |

Homepage

**Description:**

This is a series of manual test cases for selecting the Homepage.

**Assumptions:**

- N/a.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Expected Inputs | Expected Output | Actual Output | Pass/Fail |
| 01 | Click on slideshow dots | The slideshow container displays the respecetive page. | The slideshow container displays the respecetive page. | P |
| 02 | Click on right/left arrow of slideshow container | The slideshow container moves to the next/previous page. | The slideshow container moves to the next/previous page. | P |
| 03 | Click on different menu button. | The menu container displays the respective content. | The menu container displays the respective content. | P |

Inventory Page

**Description:**

This is a series of manual test cases for selecting the Inventory page.

**Assumptions:**

- The user is currently logged in.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Expected Inputs | Expected Output | Actual Output | Pass/Fail |
| 01 | Click on add button | The system prompts the user to enter item info. Then the new item is added to database and displayed on this page. | The system prompts the user to enter item info. Then the new item is added to database and displayed on this page. | P |
| 02 | Click on update button | The system prompts the user to edit info of selected item. Then the data of relevant item is updated in database. | The system prompts the user to edit info of selected item. Then the data of relevant item is updated in database. | P |
| 03 | Click on delete button | The selected item is removed from both page and database. | The selected item is removed from both page and database. | P |
| 04 | Click on category buttons of the item list. | The item list displays the sorted items by category | The item list displays the sorted items by category | P |
| 05 | Enter a keyword into search bar and click on search button | The item list displays the items that contain the keyword. | The item list displays the items that contain the keyword. | P |

Create Order Page

**Description:**

This is a series of manual test cases for selecting the ordering page.

**Assumptions:**

- The user is currently logged in.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Expected Inputs | Expected Output | Actual Output | Pass/Fail |
| 01 | Click on add order button with all info entered | The new order is added to the orders section of database file. | The new order is added to the orders section of database file. | P |

Donation Page

**Description:**

This is a series of manual test cases for selecting the donation page.

**Assumptions:**

- The user is currently logged in.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Expected Inputs | Expected Output | Actual Output | Pass/Fail |
| 01 | Click on add donation button with all info entered | The new order is added to the donation section of database file. | The new order is added to the onation section of database file. | P |

Report Generating Page

**Description:**

This is a series of manual test cases for selecting the report generating page.

**Assumptions:**

- The user is currently logged in.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Expected Inputs | Expected Output | Actual Output | Pass/Fail |
| 01 | Click on generate report with the selected data and file type. | The system export the selected data as a file of desired file type. | The system export the selected data as a file of desired file type. | P |
| 02 | Click on category buttons of the item list. | The item list displays the sorted items by category | The item list displays the sorted items by category | P |
| 03 | Enter a keyword into search bar and click on search button | The item list displays the items that contain the keyword. | The item list displays the items that contain the keyword. | P |

# Environment Requirements

The Acme Food Bank Inventory Tracking System requires a test environment that includes a server running either Windows, Linux, or macOS. Essential software includes Python 3.8 or higher, SQLAlchemy, Flask, Jinja, and Visual Studio or VS Code. The server should be connected to an usable network, set to private for secure access.

The system should operate in a web-based application mode during testing, with all dependencies installed in a virtual environment. For secure communication, configuring TLS encryption and setting the server to use HTTPS is recommended. Additional tools like a browser for accessing the web interface is also necessary.

# Glossary

Flask: A lightweight web framework in Python used to build web applications.

HTTP (HyperText Transfer Protocol): The protocol used for transmitting hypertext requests and information on the internet.

SQL (Structured Query Language): A standard programming language for managing and manipulating databases.

Jinja: A templating engine for Python, used to generate HTML dynamically by combining templates with data sources.

HTML (HyperText Markup Language): The standard markup language used for creating web pages.

JavaScript: A programming language commonly used to create interactive effects within web browsers.

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.

# References

[1] Segue Technologies. "The Four Levels of Software Testing." Segue Technologies. Accessed July 5, 2024. https://www.seguetech.com/the-four-levels-of-software-testing/.