**Test Plan Documentation**

**UrbanMart - E-commerce website**

Table of Contents

[Introduction 2](#_Toc175481284)

[Scope of Testing 3](#_Toc175481285)

[Functional Testing 3](#_Toc175481286)

[Integration Testing 3](#_Toc175481287)

[System Testing 3](#_Toc175481288)

[Regression Testing 3](#_Toc175481289)

[User Interface (UI) Testing 3](#_Toc175481290)

[Performance Testing 3](#_Toc175481291)

[Features to be Tested 4](#_Toc175481292)

[Testing Approach and Strategy 4](#_Toc175481293)

[Unit Testing 4](#_Toc175481294)

[Integration Testing 5](#_Toc175481295)

[System Testing 5](#_Toc175481296)

[Regression Testing 5](#_Toc175481297)

[User Interface (UI) Testing 5](#_Toc175481298)

[Testing Tools to be Used 5](#_Toc175481299)

[Roles and Responsibilities 5](#_Toc175481300)

[Timeline for Testing Activities 6](#_Toc175481301)

[Test Deliverables 7](#_Toc175481302)

# Introduction

This document outlines the detailed test plan for the Simple E-Commerce Platform developed by our group. The application includes essential functionalities such as User Registration, User Profile Management, Product Management (Add, Edit, View, Delete Products), and Order Management. The objective of this test plan is to ensure that the application is fully functional, reliable, and meets the requirements specified in the project documentation.

# Scope of Testing

The scope of testing for this project includes the following testing types:

## Functional Testing

* Testing each feature of the application to ensure that it performs its intended function.
* Verifying that the business logic and workflows (e.g., order processing, user authentication) work correctly.

Integration Testing

* Ensuring that different modules of the application (e.g., User Management, Product Management) interact with each other correctly.
* Validating data flow between modules, such as the user placing an order and the product inventory being updated accordingly.

## System Testing

* Testing the entire application as a whole to ensure that it meets the specified requirements.
* Verifying that all integrated components of the system work together seamlessly.

## Regression Testing

* Ensuring that any code changes or bug fixes do not introduce new defects.
* Re-running previously executed test cases to confirm that the application still works as expected.

## User Interface (UI) Testing

* Testing the application’s interface to ensure it is user-friendly, responsive, and consistent across different devices and browsers.
* Verifying that UI elements (buttons, forms, navigation) work as intended and provide a good user experience.

## Performance Testing

* Evaluating the application's performance under various conditions, such as load testing to simulate multiple users accessing the system simultaneously.
* Identifying performance bottlenecks and ensuring the application can handle expected user loads.

# Features to be Tested

**User Registration:**

* Create new user accounts
* Validation of user inputs (e.g., email, password)

**User Profile Management:**

* View, edit, and update user profile details

**Product Management:**

* Add new products
* Edit existing product details
* View product listings
* Delete products from the catalog

**Order Management:**

* Create and manage customer orders
* View order history

# Testing Approach and Strategy

Our testing approach is designed to cover all aspects of the application, ensuring comprehensive coverage and early defect detection. The strategy involves both manual and automated testing to balance thoroughness with efficiency.

## Unit Testing

Approach: Developers will write unit tests for their code using Jest. Each method or function will be tested in isolation to ensure it performs as expected.

Tools: Jest

## Integration Testing

Approach: Once unit testing is complete, integration tests will be performed to ensure that the modules interact correctly. For instance, ensuring that user login successfully adds the user to the database and allows them to place orders.

Tools: JUnit, Mockito.

## System Testing

Approach: The entire system will be tested as a whole. End-to-end test cases will be written to cover the full workflow of the application, from user registration to order completion.

Tools: JUnit, Selenium (for UI tests).

## Regression Testing

Approach: A suite of automated regression tests will be created. These tests will be run after every major change or update to ensure that existing functionality is not broken.

Tools: JUnit, Selenium.

## User Interface (UI) Testing

Approach: UI testing will be performed manually and through automation to ensure that the application’s interface is intuitive and works across different devices and browsers.

Tools: Selenium, BrowserStack (for cross-browser testing).

# Testing Tools to be Used

JUnit: For unit testing individual components of the application.

Mockito: For mocking dependencies in unit and integration tests.

Selenium: For automating UI tests to ensure that the front-end behaves as expected.

BrowserStack: For testing the application across different browsers and devices to ensure

# Roles and Responsibilities

Project Manager (Member A):

* Oversees the entire project.
* Coordinates tasks among group members.
* Ensures the project is completed on time and within scope.

Lead QA Engineer (Member B):

* Develops the test plan.
* Leads the testing process and ensures quality standards are met.
* Coordinates regression testing and continuous integration efforts.

Developers (Member C and Member D):

* Write unit tests for their respective modules.
* Fix any bugs identified during testing.
* Work closely with QA to ensure their code meets quality standards.

Test Automation Engineer (Member E):

* Develops and maintains automated test scripts using Selenium.
* Ensures that the automated tests cover all critical paths of the application.
* Collaborates with developers to integrate automated tests into the CI pipeline.

Manual Tester (Member F):

* Executes manual test cases, particularly for areas not covered by automation.
* Performs exploratory testing to identify potential issues that scripted tests might miss.
* Documents any defects found and works with developers to resolve them.

Timeline for Testing Activities

|  |  |  |
| --- | --- | --- |
| Week | Activity | Responsible Member(s) |
| Week 1 | Finalize test plan and create initial test cases | Lead QA Engineer, Developers |
| Week 2 | Begin unit testing of individual modules | Developers |
| Week 3 | Integration testing of connected modules | Lead QA Engineer, Developers |
| Week 4 | Initial UI testing and system testing | Test Automation Engineer, Manual Tester |
| Week 5 | Regression testing and bug fixes | All Members |
| Week 6 | Performance testing and final system validation | Test Automation Engineer, Lead QA Engineer |
| Week 7 | Final regression testing and test summary report | Lead QA Engineer, Project Manager |

# Test Deliverables

Test Plan Document: This document.

Test Cases: Detailed test cases for each feature.

Test Scripts: Automated test scripts for regression and UI testing.

Test Summary Report: A report summarizing the test results, including any defects found and resolved.