Car Purchasing App

Test Plan Document

5/9/2023

Issued By:

1. Nader Abdallah

Reviewed By:

Contents

Contents

Introduction:

This test plan outlines the strategies, process, workflow, and methodologies used for testing the food web app website. The website has been developed using HTML, CSS, JS, and php. The testing will be carried out using excel sheets for test cases and bug reports.

1.1) Scope:

1.1.1) In Scope:

Testing of all functional requirements such as user registration, login, menu display, ordering, and payment processing.

.

1.1.2) Out of Scope:

Testing of all non-functional requirements such as usability, performance, security, and compatibility

Testing of any features not explicitly mentioned in the functional and non-functional requirements.

1.2) Quality Objective:

The objective of the testing project is to ensure that the application under test conforms to functional and non-functional requirements. The quality specifications defined by the client must be met, and any bugs or issues identified and fixed before go-live.

1.3) Roles and Responsibilities:

QA Analyst: responsible for designing, executing, and maintaining the test cases, as well as reporting and verifying defects.

Test Manager: responsible for planning, monitoring, and controlling the testing process and ensuring that the project meets its objectives.

Configuration Manager: responsible for managing the test environment and ensuring that it is consistent with the production environment.

Developers: responsible for fixing the bugs identified in the application under test.

Testing Team: responsible for creating and executing manual and automated test cases and bug reports.

Test Methodology:

2.1) Overview:

We will be adopting the agile test methodology for this project. Agile methodology provides flexibility and encourages continuous testing, which is suitable for web applications.

2.2) Test Levels:

The following test levels will be executed:

- System Testing: carried out to test the application as a whole.

2.3) Test Cases Section:

This section provides an overview of the test cases to be executed during the testing phase. Test cases are essential for systematically verifying the functionality, performance, and usability of the software application. The following information should be included for each test case:

1. Test Case ID: A unique identifier for each test case.
2. Test Case Name: A descriptive name or title for the test case.
3. Description: A brief description of the purpose and objective of the test case.
4. Test Steps: A step-by-step sequence of actions to be performed to execute the test case.
5. Expected Results: The expected outcome or behavior of the software application when the test case is executed successfully.
6. Actual Results: The actual outcome or behavior observed when the test case is executed.

7. Test Data: Any specific data or inputs required for executing the test case.
8. Preconditions: Any specific conditions or prerequisites that need to be met before executing the test case.
9. Test Environment: Details about the test environment, including hardware, software, and network configurations.
10. Test Execution Status: The status of the test case execution (pass/fail/pending).
11. Test Priority: The priority assigned to the test case (e.g., high, medium, low).
12. Test Dependencies: Any dependencies or relationships with other test cases or modules.
13. Test Case Owner: The person responsible for designing, executing, and maintaining the test case.
14. Test Case Review: Information about the review process, including reviewers and review dates.
15. Test Case Version: The version or revision number of the test case.

2.4) Bug Report section:

The bug triage process will be carried out to determine the priority of each bug and schedule the "To Be Fixed Bugs."

1-Bug Life Cycle:

New: When a bug is reported for the first time.
Assigned: When the bug is assigned to a developer or tester.
In Progress: When the developer or tester starts working on the bug.
Fixed: When the bug has been fixed by the developer.
Retest: When the fixed bug is ready for retesting.
Verified: When the bug fix has been verified and confirmed.
Closed: When the bug is considered resolved and closed.

2-Criteria for Opening a Bug:

Clear description: A bug report should have a concise and detailed description of the issue, including steps to reproduce and expected results.
Reproducibility: The bug should be reproducible consistently to help identify and resolve the issue.
Severity and priority: The bug should be assigned appropriate severity and priority levels based on its impact on the system and business requirements.
Supporting materials: Include any relevant screenshots, logs, or additional information that can help the development team understand and reproduce the bug.

3-Bug Report Contents:

Title: A brief and descriptive title for the bug report.
Description: Detailed information about the bug, including steps to reproduce, observed behavior, and expected behavior.
Environment details: Mention the platform, operating system, device, and software versions on which the bug occurred.
Severity and priority: Assign appropriate severity and priority levels to the bug based on its impact and urgency.
Attachments: Include any relevant screenshots, logs, or files that can aid in understanding and resolving the bug.
Bug assignment: Assign the bug to the respective developer or team responsible for its resolution.
Status updates: Track the bug's life cycle, including changes in status, comments, and progress updates.

4-Bug Fixing and Verification:

Bug Fixing and Verification:

Once a bug has been reported and assigned to a developer, the bug fixing and verification process begins. This section outlines the steps involved in resolving the reported bugs and ensuring their successful closure.

Bug Fixing Process:

1-Developer Analysis: The assigned developer analyzes the bug report, reproduces the issue, and identifies the root cause of the bug.
2-Bug Fixing: The developer makes the necessary code changes or fixes to address the reported bug.
3-Code Review: The fixed code undergoes a code review process to ensure quality and adherence to coding standards.
4-Compilation and Build: The fixed code is compiled, and a new build or version of the software is created.

5-Confirmation Testing:
The developer performs confirmation testing on the fixed bug to verify that the issue has been resolved.

Bug Verification Process:

1-Regression Testing:
The fixed bug undergoes regression testing, which involves retesting the affected functionality to ensure that the fix did not introduce any new issues.

2-Confirmation Testing
by Testers: Testers also perform confirmation testing on the fixed bug to validate that the issue has indeed been resolved.

3-Verification Criteria:
The bug is considered successfully fixed and can be closed if the following criteria are met:

- The bug no longer exists or behaves as expected.

- The fix does not introduce new bugs or adversely affect other parts of the system.
- The bug fix has been confirmed and verified by both developers and testers.
- The fix has been tested in different environments (if applicable) to ensure compatibility.

During the confirmation and regression testing stages, it is crucial to document and track the test cases used, any issues encountered, and their resolutions. This helps ensure thorough testing and provides a comprehensive overview of the bug fixing and verification process.

2.4) Test Summery Report

1. Introduction
   - Provide an overview of the test summary report, including the purpose and scope.
   - Mention the project name, version, and release details.

2. Executive Summary
   - Summarize the overall testing activities and results.
   - Highlight key findings, achievements, and challenges.
   - Include any critical issues or risks identified during testing.
   - Provide an overview of the test coverage and test execution status.
   - Provide recommendations or next steps based on the test results.

3. Test Objectives
   - Specify the objectives of the testing effort.
   - Align the objectives with the project goals and requirements.

4. Test Scope
   - Define the scope of the testing activities.
   - Mention the in-scope and out-of-scope items, features, or modules.
   - Specify any specific platforms, environments, or configurations considered during testing.

5. Test Approach
   - Describe the overall testing approach or strategy adopted.
   - Mention the types of testing performed (e.g., functional, performance, security).
   - Explain any test techniques or methodologies used (e.g., black-box, white-box, exploratory).
   - Discuss any automation or tooling used for testing purposes.
   - Highlight any specific test design techniques or models applied (e.g., equivalence partitioning, boundary value analysis).

6. Test Coverage
   - Provide details on the test coverage achieved.
   - List the functional areas, features, or modules covered by testing.
   - Specify the percentage or metrics of test coverage achieved.

7. Test Execution
   - Describe the test execution process.
   - Provide details on the test environment, configurations, and setups used.

- Mention the test data and test scenarios used during execution.
- Report the number of test cases executed, passed, failed, and pending.
- Highlight any issues or defects encountered during execution.

8. Test Results
   - Present the overall test results and metrics.
   - Provide a summary of the pass/fail percentages.
   - Include any significant defects or issues found during testing.
   - Discuss any performance, security, or usability concerns identified.
   - Report any deviations from expected behavior or requirements.

9. Test Summary
   - Summarize the overall test effort.
   - Assess the quality of the software application based on the test results.
   - Provide an evaluation of the testing process, including strengths and areas for improvement.
   - Identify any risks or challenges related to testing and their impact on the project.

10. Conclusion
    - Conclude the test summary report.
    - Provide any final recommendations or actions to be taken.
    - Acknowledge the team members and stakeholders involved in the testing effort.
    - Express gratitude for their contributions and support.

11. Appendices
    - Include any supporting documents, test artifacts, or additional information.

12. References
    - Provide references to any documents or sources used in preparing the test summary report.

2.5) Suspension Criteria and Resumption Requirements:

The testing procedure will be suspended if any of the following criteria are met:

- Defects found in the application cannot be fixed before release delivery.(see section 2.3)
- The testing environment (See Section 4.2) is not available.

-The testing can resume once the above criteria are resolved. Refer to risk strategy document (please refer to the following link: https://github.com/beshoysamehsamaan/Car-Purchasing-App/blob/main/1-%20PM/project%20risk%20management.docx)

2.6) Test Completeness:

Testing will be deemed complete when the following criteria are met:

- All functional requirements have been tested and passed.
- All manual and automated test cases have been executed and passed.
- All open sever bugs have been closed, or will be closed in the next release.(See section 2.3)

3) Test Deliverables:

The following test artifacts will be delivered during different phases of the testing lifecycle:

- Test Plan: detailing the testing strategy, process, and methodology.
- Test Cases: outlining the test cases designed for each test level. *(to refer to the test cases sheet please consider the following link: https://github.com/beshoysamehsamaan/Car-Purchasing-App/blob/main/5-%20Testing/Test%20Cases.xlsx)*
- Requirement Traceability Matrix: outlining the relationship between the requirements and test cases.
- Bug Reports: documenting defects found in the application.
- Test summery report ()

- Customer Sign-Off: sign-off by the client to indicate that the application has been tested and meets the required quality standards.

Resource & Environment Needs

4.1) Testing Tools

The following tools will be used for testing:

Excel sheets for test cases and bug reports writing

- Git and GitHub version control system
- TFS for project management

4.2) Test Environment

The following software and hardware requirements are needed for testing:

- Windows 10 or above operating system
- Latest version of IntelliJ IDEA IDE with Selenium WebDriver plugin
- Latest version of JDK
- Latest version of Google Chrome browser

Internet connectivity

- Sufficient hardware resources including RAM, processor, and hard disk space
- Access to test database
- Additional software specific to the client's application may also be required.

Note: It is recommended that the test environment is identical to the production environment to ensure accurate and reliable results.

Test Execution Schedule

The test execution will be carried out in two phases, each phase consisting of functional and non-functional testing. The estimated time for each phase is:

- Phase 1: 1 week
- Phase 2: 1 week

Test Reporting

Will be used to generate detailed reports for each test run. The reports will include the following information:

- Test case status (Pass/Fail)
- Test case execution time
- Screenshots for failed tests
- Logs for failed tests
- Test case steps
- Test case description
- Test case owner
- Risks and Mitigation Strategies

Risk strategy

The following risks and their mitigation strategies have been identified:

Integration issues with client's existing systems - Regular communication with the client's development team and frequent testing of integrations during the development cycle will help mitigate this risk.

Insufficient time for testing - Proper planning and prioritization of test cases will help ensure that critical tests are completed within the given timeframe.

Inadequate test coverage - Continuous review of the test plan and test cases with the client's team will help identify areas where additional testing is required.

Conclusion

This test plan outlines the testing strategies, process, workflow, and methodologies that will be used to test the food web app website. By following this plan, we aim to ensure that the application meets the

Functional and non-functional requirements defined by the client and is free of critical bugs and issues.