SWT27(8)1Project

Jaden van der Lely
Johannes Theunissen
Yanga Mazibuko
Johannes Cornelius Muller

Test Plan

1. Introduction:

Software testing is a crucial step in the software development lifecycle, ensuring that the developed solution meets the required standards of functionality, performance, and quality. In this project, the solution for Michael & Themba Technologies will be tested using both Whitebox and Blackbox techniques to identify defects, ensure proper functionality, and improve the overall quality of the software.

The goal is to evaluate the software solution using manual and automated testing to ensure it works as intended and aligns with the client's expectations. xUnit will automate test execution, ensuring a faster and more efficient testing process.

2. Scope:

This test plan focuses on evaluating key functionalities of the software applications developed in Visual Studio 2022 using C#. The testing will encompass three main applications: a Banking Application, an Inventory Management System (IMS), and a Flight Booking System.

2.1 Functions to be Tested:

2.1.1 Banking App:

(**Windows forms?)** User Interface (GUI): Ensuring that the graphical interface is intuitive and functional for banking operations like balance inquiries, transfers, and account management.

- **Transaction Processing**: Verifying the accuracy and reliability of banking transactions such as deposits, withdrawals, and transfers.

- **Database Operations**: Testing CRUD operations (Create, Read, Update, Delete) related to user accounts and transaction history to ensure data integrity.

- **Security Features**: Ensuring the implementation of secure authentication and authorization mechanisms.

2.1.2 Inventory Management System (IMS):

- **Product Management**: Testing the addition, updating, and removal of products from the inventory.

- **Stock Level Monitoring**: Verifying the system correctly tracks stock levels and triggers reorder alerts when stock is low.

- **Report Generation:** Testing the generation of inventory reports, ensuring they are accurate and comprehensive.

3. Quality Objectives

3.1 Primary Objectives

The primary objective of testing the system is to ensure that it meets all functional and non-functional requirements as outlined in the project specifications.

The key focus areas include:

- **Functionality:** Ensuring that all features, such as product management, stock level monitoring, and report generation, work as intended.
- **Reliability:** The system should perform consistently under normal and peak conditions.
- **Performance:** Critical actions such as adding products, updating inventory, and generating reports should be executed efficiently without undue delays.
- **Security:** All data should be securely handled, especially during operations involving sensitive information like stock levels or financial transactions.
- **Compliance:** The system should meet regulatory and client-specific requirements.
- **Usability:** The system should be easy to use, with intuitive user interfaces that enable users to interact with the application efficiently.

3.2 Secondary Objectives

Secondary objectives aim at continuous improvement of the system's quality, including:

- **Bug Detection:** Exposing as many defects as possible during the testing phases to reduce the number of bugs in the production environment.
- **Risk Mitigation:** Identifying and addressing risks associated with system crashes, data inaccuracies, or performance bottlenecks before release.
- **Maintainability:** Ensuring the system is structured and documented in a way that future changes, enhancements, and maintenance are manageable and cost-effective.
- **Scalability:** Testing how the system handles increased workloads, ensuring the system can grow in terms of both the number of users and data volumes without degradation in performance.

4. Test approach

The test approach will combine white-box testing, black-box testing, and automation techniques across all systems. White-box testing will focus on internal code structures, including unit testing for transaction processing, database operations, security mechanisms in the Banking App, and stock-level monitoring in the IMS using techniques such as decision, statement, and conditional coverage.

Black-box testing will validate the user interface, transaction flows, and system behaviors without regard to the internal code, ensuring end-to-end functionality in flight booking, inventory management, and banking transactions by using techniques such as the 3-point BVA, equivalence partitioning, and decision table. Automation will be integrated into both white-box and black-box tests, enabling efficient, repeatable testing of core functionalities like booking flows, CRUD operations, and report generation, ensuring scalability and coverage across various user scenarios whilst creating test cases with the use of xUnit.

4.1 **Test Automation**

The test automation strategy will apply both white-box and black-box testing techniques to streamline validation processes. White-box automation will focus on internal components such as transaction processing logic, database CRUD operations, and API responses in the Banking App, IMS, and Flight Booking System, ensuring data accuracy and flow integrity. Black-box automation will simulate user interactions with the UI, automating tasks like flight bookings, inventory updates, and banking operations, while validating expected system outputs.

5. Roles And Responsibilities

Role	Staff Member	Responsibilities
Group Member	Jaden van der Lely	The Introduction, Scope, and Quality objectives of the test plan including both primary and secondary objectives.
Group Member	Yanga Mazibuko	Focusing on the approach that will be used for the test as well as the roles and responsibilities of each member.
Group Member	Johannes Corneilus Muller	Looking at the entry & exit criteria, suspension criteria, resumption requirements, and the test schedule involving the approval, and the terms used in the plan.
Group Member	Johannes Theunissen	Taking on the strategy that will used for the test including the bug life cycle and testing types. The resource and environment needs should also be mentioned involving topics like testing tools and the test environment.

6. Entry and Exit Criteria

6.1 Entry Criteria

- The race system should be installed, configured, and functioning properly.
- All race and jockey information should be available to allow testers to verify the correct behavior.
- All necessary testing tools, including data visualization and countdown timer, must be successfully installed and functional.
- Proper race data, such as horse details, jockeys, and race timings, should be available.
- The test environment, such as browsers and relevant APIs for interaction, should be prepared.
- QA resources have a complete understanding of the race rules and race details.
- QA resources are knowledgeable about the race simulation functionality.
- All test scenarios, test cases, and traceability matrices (RTM) have been reviewed.

6.2 Exit Criteria

- A certain level of test case coverage for all race-related scenarios has been achieved.
- No high-priority bugs related to race timings, data display, or performance are left outstanding.
- All high-risk areas, such as countdown functionality and race result display, have been fully tested, with only minor issues remaining.
- The test phase is complete when the testing budget has been spent or the schedule has been reached.

7. Suspension Criteria and Resumption Requirements

7.1 Suspension Criteria

- The race simulation contains severe defects that block further testing progress.
- Significant changes in race requirements, such as alterations in the race distance or horse details, suggested by stakeholders.
- Issues related to hardware or software occur.
- Assigned resources are unavailable when needed by the test team.

7.2 Resumption Criteria

- Testing will only resume when the issues that caused the suspension have been resolved.

8. TEST STRATEGY

8.1 QA's role in the test process

Understanding Requirements:

- Requirement specifications for the Banking Application, Inventory Management System, and Flight Booking System were provided by the client in the project for PRG27(8)1.
- The QA team will conduct a comprehensive review of these documents to fully understand the expected functionalities and features of each application.

Preparing Test Cases:

- **Banking Application:** Test cases will cover transaction processing, user interface, and security features.
- **Inventory Management System:** Test cases will include product management, stock level monitoring, and report generation.
- **Flight Booking System:** Test cases will focus on flight search, booking, cancellations, and rescheduling.

Preparing Test Matrix:

- The QA team will create a test matrix linking each test case to its corresponding requirement from the provided specifications. This ensures comprehensive coverage for all functionalities across the three applications.

Reviewing test cases and matrix:
The QA Lead will conduct a peer review to validate the test cases and test matrix for all three systems.

- The test case author will address the review feedback and submit updated test cases for final approval Creating Test Data:
  - Test data will be created by the QA team on the client's development/test site.

- This data will simulate real-world scenarios for each application, covering:
  - **Banking Application:** Sample user accounts, transactions, and authorization credentials.

  - **Inventory Management System:** Product inventory lists, stock thresholds, and sample reports.

  - **Flight Booking System:** Flight schedules, seat availability, and customer booking details.

Executing Test Cases:

- The QA team will execute test cases on the client's development/test site for each application. Results such as actual vs. expected outcomes, along with pass/fail status, will be documented in the test case documents.

Defect Logging and Reporting:

- The QA team will document any defects identified during testing in a Word file. The details of these defects will then be communicated to the responsible developer for each system.

Retesting and Regression Testing:

- After the developer fixes a defect, retesting will be performed to ensure it has been resolved.

- Regression testing will be conducted to verify that the fix neither introduces new issues nor impacts other parts of the application.

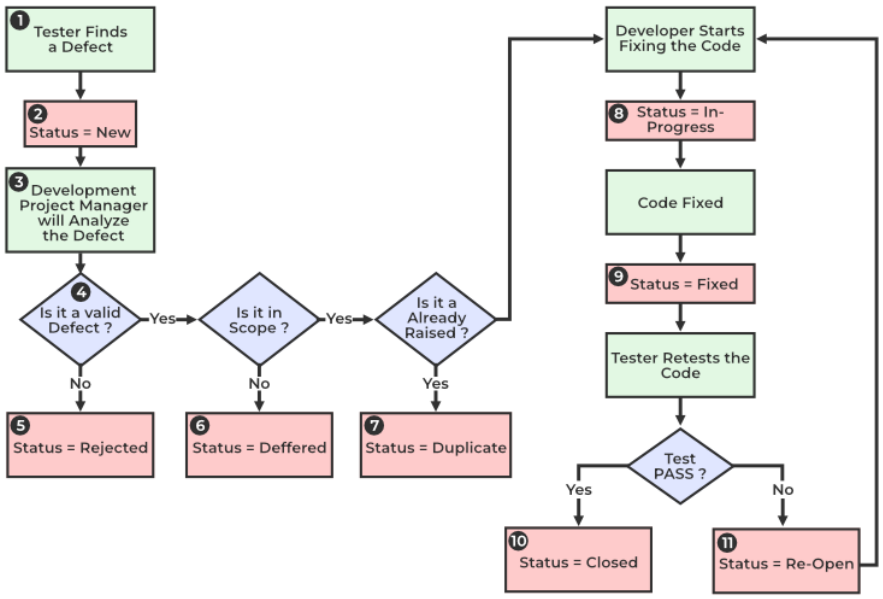
Deployment/Delivery:

- Once all reported bugs are resolved and no new issues are identified, the Project Manager will deploy the report to the client's test site.

- If needed, the QA team will perform one final round of testing on the client's test site before delivery.

- A final report, along with sample outputs, will be sent to the lead and reporting group. Additionally, a hard copy of the delivery slip will be signed by both the QA and developer.

8.2 Bug life cycle:



8.3 Testing Types

Black Box Testing:

- Focuses on testing the functional requirements of the Banking App, Inventory Management System, and Flight Booking System without any knowledge of internal code structure.

- Ensures that the applications perform as expected from the user's perspective, focusing on input and output behaviors.

White Box Testing:

- Concentrates on testing internal structures or workings of the Banking App, Inventory Management System, and Flight Booking System.

- Verifies internal code logic, code coverage, and paths to ensure that all conditions and loops work correctly.

9. RESOURCE AND ENVIRONMENT NEEDS

9.1 Testing Tools

Process	**Tool**
Test case creation	Microsoft Word
Test case tracking	Microsoft Excel
Test case execution	Manual testing, Visual Code
Test case management	Microsoft Excel
Defect management	Microsoft Word
Test reporting	Microsoft Word
Checklist creation	Microsoft Excel
Project structure	Microsoft Word

- **Test Case Creation & Tracking:** Test cases for the Banking Application, Inventory management system, and Flight Booking System will be written and tracked using Microsoft Excel and Word

- **Test Case Execution:** Manual testing and Visual Code will be used for automation where applicable and will be used for test execution across all applications.

- **Defect Management:** Defects will be tracked and managed using Microsoft Word, with regular updates shared with the development team.

- **Test Reporting:** Reports summarizing test results and metrics will be written in Microsoft Word.

- **Checklist Creation:** Microsoft Excel will be used to create checklists to track testing progress and coverage.

- **Project Structure:** Microsoft Word will be used to visualize and organize the project's testing process and overall structure.

9.2 Configuration Management

- **Documents CM:** SVN will be used for managing test-related documents.

- **Code CM:** Git will be used for version control and management of the test scripts and the application source code.

9.3 Test Environment

- **Support Level 1 (Browsers):**
  - **Windows 10/11:** Edge (latest), Chrome (latest), Firefox (latest), Safari (latest).
  - **Mac OS X:** Chrome (latest), Firefox (latest), Safari (latest).
  - **Linux Ubuntu:** Chrome (latest), Firefox (latest).

- **Support Level 1 (Devices):**
  - **iOS Devices:** iPhone 14/15, iPad Air, iPad Pro.
  - **Android Devices:** Google Pixel 5/6, Samsung Galaxy S22/S23.
  - **Other:** Microsoft Surface Pro, Lenovo ThinkPad.

- **Support Level 2:**
  - **Windows 7:** IE 11, Chrome (latest), Firefox (latest).
  - **Windows XP (Legacy):** IE 8, Chrome (latest), Firefox (latest).

- **Support Level 3:**
  - Any other platforms or devices not explicitly listed will receive minimal support and will only be tested if deemed necessary by the client or project lead.

10 Test Schedule

Task Name	Start	Finish	Effort	Comments
Test Planning	09/26/2024	09/27/2024	Support!	Planning the race test strategy and resources.
Review Race Data and Requirements	09/26/2024	09/27/2024		Verifying horse and jockey data.
Create Test Basis	09/27/2024	09/28/2024		Test scenarios for race details and countdown timer.
Staff and Train New Test Resources	-	-		
First Deploy to Race Test Environment	09/28/2024	09/29/2024		Race system setup for testing.
Functional Testing - Iteration 1	09/28/2024	09/29/2024		Focus on countdown timer and race data display.
Iteration 2 Deploy to Race Test Environment	09/29/2024	09/30/2024		Second iteration for bug fixes and enhancements.
Functional Testing - Iteration 2	09/29/2024	09/30/2024		Testing for stability and user interface functionality.
System Testing				
Regression Testing				
UAT (User Acceptance Testing)				Testing with end-users for race scenario validation.
Resolution of Final Defects				Final bug fixes before staging environment release.
Deploy to Staging Environment				
Performance Testing				Load testing the race system.
Release to Production				Go live for actual race management.