Test Plan Template

1. **Introduction**

**Project Overview**

**The software we are testing is a delivery program that will be used by a local delivery company. The company has three different trucks which deliver on three different routes in the city. Our program objective is to design a program that can find the most suitable truck and calculate the shortest delivery route for the truck.**

Test Plan Objectives

**The test plan will illustrate the objectives, strategies, processes, and tools used when testing this truck delivery software. Our test plan objectives are as follows:**

* **Testing the software to verify it fulfills all the functional system requirements specified in the SFT221 project instruction.**
* **Testing the software to identify any issues and bugs in the program.**
* **Recording and sharing the testing outcomes with the team to ensure effective communication.**

1. **Scope**

**The test plan will encompass the testing of all functions in this software program, specifying the approach of unit, integration, acceptance testing. The plan will include testing of all system use case requirements as started in the project instruction, along with performance and security testing to ensure the software is bug- free.**

1. **Test Strategy**

**Approaches to Perform the Tests:**

* **Blackbox Unit Testing: where we sperate the whole testing into small units. Through comparing expected results and actual results to find if any issues in each unit. It may take 1~2 hour to design tests, and < 10 minutes to execute the tests.**
* **Whitebox Unit Testing: Where we will analyze the internal structure of the code to design tests. Whitebox testing produces additional test that test every path through the code. It may take 1~2 hour to analyze and design tests, and < 10 minutes to execute the tests.**
* **Integration Testing: Once the functions are integrated into the program, we can apply integration test to ensure the group of functions work together correctly. It may take 1~2 hours to design tests and < 10 minutes to execute these tests.**
* **Acceptance Testing: Where we run examples of how the user would use the system to verify this truck delivery program can meet all the requirements outlined in the instruction. It may take 1 hour to design and < 10 minutes to execute them.**

Test Design Processes:

* 1. **Understand Requirements: It is important to understand what the software is expected to do and how it should behave. We will review the project documentation, specifications, and other materials.**
  2. **Build a Traceability Matrix: Building traceability matrix can help us track the progress of testing easily and make sure all requirements are fulfilled.**
  3. **Prepare Test Cases: After we understand the requirements, we will prepare test cases, which outline specific inputs and expected outputs for different tests of our truck delivery software.**
  4. **Reviewing Test Cases: another group member of our quality assurance team will review the test cases to check if any issues or gaps and provide suggestions for improvement.**

1. **Environment Requirements**

**This project's testing environment requires the following hardware and software:**

**• Hardware Requirements:**

**Test computer(s) having sufficient specifications to efficiently run the software under test.**

**Enough storage space to hold the test data and artifacts.**

**Network connectivity is required to access resources and execute remote testing.**

**• Software Requirements:**

**Operating System: Windows 11, macOS 13.0**

**Development Environment: Visual Studio, Visual Studio Code**

**Test Management Tools: JIRA for issue tracking and test case management. GitHub provides a centralized repository for storing and sharing test-related artifacts. TotoiseGit for git clone, commit, pull, and push.**

**Before beginning the testing phase, ensure that the necessary hardware and software requirements are present and correctly configured.**

1. **Execution Strategy**
   1. Entry and exit criteria

* Entry Criteria

Test environment is set up and configured correctly.

Test cases and test data are prepared and reviewed.

Development of the software under test is completed.

The software build or version to be tested is available.

* Exit Criteria:

Test coverage reaches a predetermined level,95% of test scripts executed.

Pass 100% of the tests.

All critical defects are resolved and retested.

Testing objectives and goals are achieved.

Test environment and data are cleaned up and restored.

Test deliverables are completed and reviewed.

* 1. Severity of defects
     1. **Critical:** Defects that cause the system to crash or produce anomalous results, rendering it unusable.
     2. **High:** Defects that cause a lack of program functionality and may require a workaround to achieve desired functionality.
     3. **Medium:** Defects that degrade the quality of the system but often have a workaround to provide the desired functionality.
     4. **Low:** Minor errors with minimal impact on functionality, such as unclear error messages or cosmetic issues.
     5. **Feature Request:** Suggestions for improving the user interface but do not affect the system's core functionality.
  2. **Test Reporting**
     1. Report to be produced:

Test execution report (Daily) to project manager, development team, QA team: A summary of the test execution progress, including the number of tests run, passed, and failed.

Defect Report (Whenever new defect is found and resolved) to project manager, development team, QA team: Lists all identified defects along with their severity, description, and reproducibility procedures.

Test Coverage Report(weekly) to Project manager, QA team: A report that summarizes the coverage obtained in terms of requirements, functionality, or code coverage.

Test review Report (at the end of each testing cycle) to Project manager, stakeholders, management: A thorough review of the complete testing effort, including major findings, metrics, and recommendations.

* + 1. Testers will give project managers with defect information, including defect specifics, procedures to reproduce, and supporting documentation.

Project managers will assign and prioritize developers to solve reported defects based on their severity.

To facilitate communication between management, development, and quality assurance teams, regular communication channels such as meetings and collaboration tools will be established.

* 1. **Interaction with the developers**

QA team will interact with developers in the following ways to resolve defects found in the software:

Defect Reporting: QA team will log and report defects to the development team, providing detailed information about each defect.

Defect Triage: QA and development teams will collaborate to prioritize and categorize defects based on severity and impact.

Defect Resolution: Developers will fix the reported defects, and QA team will verify the fixes and retest the affected areas.

Collaboration and Communication: QA and development teams will have the meeting, discuss on the Teams and work together to discuss and resolve any issues or questions related to defects and testing.

1. **Test Schedule**

|  |  |  |  |
| --- | --- | --- | --- |
| **Due date** | **Milestone** | **Task** | **Deliverables** |
| July 3rd, 2023 | Milestone 1 | Setup teams, GIT, and Jira accounts | Team contract, Initialized Git repository, Jira project |
| July 10th, 2023 | Milestone 2 | Study the problem, analyze existing code, create new data structures, Draft a test plan | Problem analysis, Data structures in repository, Test plan |
| July 17th, 2023 | Milestone 3 | Design and spec functions, Start writing blackbox tests and implementation of functions, update Jina project Add a function-test repository matrix | Function specifications, Blackbox tests, Function implementation, Function-test matrix |
| July 24th, 2023 | Milestone 4 | Complete blackbox tests, run blackbox tests, Debug as needed, Write and implement whitebox tests, Create test automation script | Completed blackbox and whitebox tests, Test automation script |
| July 31st, 2023 | Milestone 5 | Write, implement, and execute integration tests, Write acceptance tests | Completed integration and acceptance test executions, Debugging |
| August 7th, 2023 | Milestone 6 | Run acceptance tests, Debug as needed, produce a testing report, Review the test matrix | Acceptance test executions, Testing report, Reviewed test matrix |

1. **Control Procedures**

|  |  |
| --- | --- |
| **Process / Practice** | **Details** |
| Reviews will be done on: | Test Plan  To ensure the progress align with the project goals |
|  | Test Cases  To ensure the test cases are well designed with suitable coverage |
|  | Test Results  To ensure the correctness of the test executed |
| Test Environment | The test environment will be set up according to the environment requirements and will remain available throughout the testing process. |
| Bug Review Meetings and Progress Monitoring | Bug Review Meetings will be held biweekly to review the reported bugs. The progress of bug resolution will be monitored to ensure that it stays on track and is completed within the designated time frame. |
| Change Request | Understanding the change request and assessing its impact on the existing test plan, amendments will be made, and regression testing will be conducted. |
| Defect Reporting | Defect details, including their nature and severity, will be logged and reported immediately. |
| Test Execution Schedule | The Test Execution Schedule will be planned based on the project schedule and the availability of resources to execute the tests. |
| Documentation | The Test Plan will be kept and updated throughout the process. The Traceability Matrix Report will be maintained, The Test Cases and test data will be documented and the Test Report will be generated. |
| Communication | here will be a weekly testing team meeting where the testing progress will be shared, and any issues that require immediate response will be discussed through MS Teams. |

1. **Functions To Be Tested**

* struct Map populateMap();
* int getNumRows(const struct Map\* map);
* int getNumCols(const struct Map\* map);
* void printMap(const struct Map\* map, const int base1, const int alphaCols);
* struct Map addRoute(const struct Map\* map, const struct Route\* route);
* void addPtToRoute(struct Route\* route, struct Point pt);
* void addPointToRouteIfNot(struct Route\* route, const int row, const int col, const struct Point notThis);
* void addPointToRoute(struct Route\* route, const int row, const int col);
* struct Route getBlueRoute();
* struct Route getGreenRoute();
* double distance (const struct Point\* p1, const struct Point\* p2);
* struct Route shortestPath(const struct Map\* map, const struct Point start, const struct Point dest);
* struct Route getPossibleMoves(const struct Map\* map, const struct Point p1, const struct Point backpath);
* int getClosestPoint(const struct Route\* route, const struct Point pt);
* int eqPt(const struct Point p1, const struct Point p2);

1. **Resources and Responsibilities**

|  |  |  |
| --- | --- | --- |
| Name | Roles | Responsibilities |
| Siu Man Cheng | Team Leader (Developer) | Team management (tasks assigning, facilitate team communication), monitor the project progress and efficiency.  Function specifications, implementation and debugging (bugs fixing and documenting) |
| Ye Tian | Developer | Function specifications, implementation and debugging (bugs fixing and documenting) |
| Lok Yin Tai | Developer | Function specifications, implementation and debugging (bugs fixing and documenting) |
| Xinyang Wu | Quality Assurance | Test case and data design, including blackbox tests, whitebox tests, integration tests and acceptance tests.  Tests execution (perform, results record) |
| Xiaofei Xu | Quality Assurance | Test case and data design, including blackbox tests, whitebox tests, integration tests and acceptance tests.  Tests execution (perform, results record) |
| Chang Cui | Quality Assurance | Test case and data design, including blackbox tests, whitebox tests, integration tests and acceptance tests.  Tests execution (perform, results record) |

1. **Deliverables**

The following documents will be produced as part of the testing process:

* Test Plan
* Traceability Matrix Report
* Test Cases (and test data)
* Test Report
* Actual source Code for the final project

1. **Suspension / Exit Criteria**

Testing activities may be paused or terminated due to specific circumstances or events, such as:

* Critical issues identified during testing that require immediate resolution.
* Unavailability of essential dependencies required for testing.
* Unforeseen technical or environmental issues impacting the testing process.
* Changes in project priorities or scope that require a pause in testing.
* If the test has successfully achieved its intended objective and the desired outcome has been obtained.

1. **Resumption Criteria**

The authority to make the decision to suspend testing rests with the designer of this specific test case (to be assigned during each meeting), whom should have the necessary knowledge and authority to assess the situation and make informed decisions regarding the suspension of testing activities.

Testing activities can resume once the following conditions or criteria have been met:

* Critical issues identified during testing have been resolved, and the fixes have been verified.
* Essential resources or dependencies required for testing are available and properly configured.
* Technical or environmental issues that impacted the testing process have been addressed and resolved.
* Changes in project priorities or scope have been adequately communicated and accounted for in the testing plan.

If the resumption criteria cannot be met within a specified timeframe, the following escalation path and responsibilities should be established:

* If the decision authority is unable to reach a resolution within the specified timeframe, the issue should be escalated to the group.
* All team members should have the authority and responsibility to assess the situation, collaborate with relevant tester, and make decisions regarding further actions, such as seeking additional resources or adjusting the testing plan.
* The communication channels and responsibilities for escalating and resolving the issue should be clearly defined to ensure timely and effective resolution.

1. **Dependencies**

13.1 **Personnel Dependencies**: During team meetings, it is important to consider the workload of each team member and assign testing tasks accordingly. By distributing the workload efficiently, the aim is to ensure that each team member can focus on their assigned testing responsibilities and perform their tasks effectively.

13.2 **Software Dependencies**: GitHub is used for version control, allowing for seamless integration of testing files and facilitating collaboration among team members. JIRA is employed to allocate tasks and ensure effective communication within the team. Additionally, Visual Studio is utilized as the testing tool, providing the necessary features and capabilities for executing and managing the testing process.

13.3 **Hardware Dependencies**: Each team member relies on their respective computers to carry out testing activities. The availability and proper functioning of individual computer systems are essential to maintain communication and ensure uninterrupted testing.

13.4 **Test Data & Database Dependencies**: Testing requires relevant and appropriate test data to simulate real-world scenarios and cover different test scenarios. The availability and proper management of test data are crucial for effective testing.

1. **Risks**

14.1. **Schedule Risk**: Insufficient time allocated for testing activities may result in rushed or incomplete testing, leading to potential defects in the software. It is crucial to accurately estimate the required testing effort and allocate sufficient time to ensure comprehensive testing coverage.

14.2. **Technical Risk**: While we have taken steps to minimize technical risks by utilizing tools like GitHub and JIRA, it is important to remain cautious when pushing changes to the repository on GitHub.

14.3. **Management Risk**: Lack of clear communication and coordination between team members may lead to misalignment of expectations and delays in decision-making. To mitigate this risk, regular and effective communication channels should be established, including team meetings, progress updates, and documentation of decisions made.

14.4. **Personnel Risk**: If team members lack motivation or fail to engage in effective communication, it can result in reduced efficiency and compromised software quality.

14.5. **Requirements Risk**: Incomplete or conflicting requirements can lead to misunderstandings in the delivered software and user expectations. Clear documentation of requirements, regular reviews, and a systematic change management process will help identify and address any inconsistencies or gaps in the requirements.

1. **Tools**

Several software tools and technologies are utilized during the testing process:

|  |  |
| --- | --- |
| Tool | Objective |
| Jira | To perform team collaboration by communicating project progress and prioritizing tasks using Kanban boards. It can also manage tasks, testing progress, bugs, and issues tracking in a visualized way. |
| GitHub | To manage project, documents, source code version control with the ability to share and contribute among team members together. It improves team productivity and minimizes version conflicts. |
| Excel | To create test plan, test cases, traceability matrix and reports. It supports test data and result analysis. |
| Visual Studio | To perform coding, testing, and debugging project source code. |
| MS Teams | To provide instant communication and remote meeting channel among team members. |

1. **Documentation**

Below documentation are required throughout the testing process:

|  |  |
| --- | --- |
| Document | Description |
| Project Requirement | To understand the project requirements with desired functionality and results. It works as the foundation of testing process and test case creation. |
| Scrum Report | To understand testing tasks progress and share knowledge between team members with effective communication. |
| Test Plan | To outline the scope, resources, testing approach, testing process and schedule required as a foundation to implement software testing in the project. |
| Test Case Report | To include testing scenarios and details instructions such as input, expected output to ensure program is performed correctly in quality standard and look for potential issue. |
| Traceability Matrix Report | To ensure project requirements are covered, and testing are conducted with expected result by tracking in a structured format. |

1. **Approvals**

All team members are responsible to prepare, review and approve all documentations throughout the testing process to ensure deliverables are aligned with project objectives and requirements and maintain high quality standard of the project.  
  
By signing below, I agree the content and approaches of the test plan stated in this document.  
DATE: 10 July, 2023Print Name: Chang Cui Signature: Chang CuiPrint Name: Lok Yin Tai Signature: Lok Yin TaiPrint Name: Siu Man Cheng Signature: Siu Man ChengPrint Name: Xiaofei Xu Signature: Xiaofei XuPrint Name: Xinyang Wu Signature: Xinyang WuPrint Name: Ye Tian Signature: Ye Tian