Group 3 Test Plan

1. Introduction

This project aims to optimize the delivery process by efficiently allocating shipments to trucks based on distance, capacity, and route constraints for a local delivery company. Our team may perform unit testing, integration testing, functional testing, and acceptance testing to make sure the project meets client requirements.

2. Scope

a. What will be tested:
  - User input
  - Shipment assignment algorithm
  - Algorithm for finding the shortest route
  - Error handling

b. What will not be tested:
  - Integration with other systems, such as dispatch system or tracking system.

3. Test Strategy

a. Testing Types
  - Unit Test
  - Integration Test
  - Blackbox Test
  - Whitebox Test
  - User Acceptance Test

b. Testing Strategy Details
  - Understand requirements by reviewing requirement documents and client stories. After understanding the requirements, we create a traceability matrix to link the test cases and requirements.
  - List all the requirements in the traceability matrix and make sure each requirement has at least one corresponding test case.

- Prepare test cases with all the scenarios of black box and withe box. All the test cases will be reviewed by other teammates to make sure the test cases are accurate and related to the requirements.

4. Environment Requirements

a.	Hardware:	Windows PCs and MacBook
b.	Software:	Visual Studio 2022
c.	Programming Language:	C and C++

5. Execution Strategy

a. Entry Criteria:
   - Completion of a specific development phase,
   - the availability of the test environment,
   - the approval of test cases and test data.

b. Exit Criteria:
   - Passing a specified percentage of test scripts from 95% or 100%.
   - Completing all test cases with certain priorities.
   - No open or unresolved defects.

c. Types of Tests Conducted
   - Unit Testing: Verifies the functionality of individual components or units of code.
   - Black box Testing: Test the functionality of each function.
   - White box Testing: Test the logic of each function.
   - Integration Testing: Ensures that different components or systems work together.
   - Acceptance Testing: Confirms that the software meets the client's requirements and is ready for deployment.

d. Defect Reports
   - Contents: Detailed information about each defect, including severity level, affected functionality, steps to reproduce, and any workarounds if available.
   - Generation Conditions: Created immediately upon discovery of a defect.

- Recipients: Assigned directly to relevant developers via a defect tracking system, with summaries provided to project managers and team leads.

6. Test Schedule

a. Milestone 2 (By March 12, 2024): finish the general test plan.
b. Milestone 3 (By March 22, 2024): Create and execute Blackbox tests for some functions.
c. Milestone 4 (By March 27, 2024): Complete the Blackbox tests. Create and execute Whitebox tests for some functions, debugging the program.
d. Milestone 5 (By April 10, 2024): Complete the Whitebox tests. Create and execute integration tests and acceptance tests for the program. And debug the program .
e. Milestone 6 (By April 18, 2024): Complete the acceptance tests and final tests. Test all bugs are fixed. Complete test report

7. Control Procedures

a. Reviews

Review testing progress regularly to ensure the method and outcomes align with the project's goals. Report and discuss problems or difficulty during the testing progress.

b. Bug Review Meetings

Discuss importance of the discovered bugs. Team members can provide their insights of how to address the bugs.

c. Change Request

When changes to the project or testing plan are needed, inform other team members as soon as possible. Schedule meeting to discuss and adjust test plan or test cases based on the new changes. Ensure every member understand the impact and adjust their work accordingly.

d. Defect Reporting

Report discovered defects by creating a spreadsheet and a bug report Jira ticket. List all key information about the bug, and updates of any changes of the bug.

8. Functions To Be Tested

a. struct Map populateMap();
b. int getNumRows(const struct Map* map);
c. int getNumCols(const struct Map* map);
d. void printMap(const struct Map* map, const int base1, const int alphaCols);
e. struct Map addRoute(const struct Map* map, const struct Route* route);
f. void addPtToRoute(struct Route* route, struct Point pt);
g. void addPointToRouteIfNot(struct Route* route, const int row, const int col, const struct Point notThis);
h. void addPointToRoute(struct Route* route, const int row, const int col);
i. struct Route getBlueRoute();
j. struct Route getGreenRoute();
k. struct Route getYellowRoute();
l. double distance(const struct Point* p1, const struct Point* p2);
m. struct Route shortestPath(const struct Map* map, const struct Point start, const struct Point dest);
n. struct Route getPossibleMoves(const struct Map* map, const struct Point p1, const struct Point backpath);
o. int eqPt(const struct Point p1, const struct Point p2);
p. int getClosestPoint(const struct Route* route, const struct Point pt);
q. New created: int inputWeightValidation(const int weight);
r. New created: int inputDestinationValidation(const struct Point point, const struct Map* map);
s. New created: int inputSizeValidation(const double size);
t. New created: int getUserInput(struct UserInput* input, const struct Map* map);
u. New created: void printMessage(int selection, const struct Route* route);
v. New created: void convertStringToPoint(struct Point* destination, const char* source);

9. Resources and Responsibilities

a. Resources
   - Team members: 1 developer, 2 QA analysts, 1 flexible role (coordinator)

    - Hardware: Windows PCs and MacBook
    - Tools: GitHub, Jira, Visual Studio 2022, Microsoft Office
b. Responsibilities
    - QA analysts to develop and execute test cases.
    - Developers: complete code implementation, fix bugs
    - Flexible role: help other members' work or edits the document.

10. Deliverables

A comprehensive Final Test Report summarizing all testing activities. And the program is performed as expected.

11. Suspension / Exit Criteria

a. The project passes the deadline.
b. Cannot achieve the client requirements after processing the testing strategies.
c. Other teams complete the project and accepted by the client.

12. Resumption Criteria

a. Client extends the deadline of this project.
b. Once the necessary adjustments have been made and verified through subsequent testing strategies, ensuring that all client requirements are fully satisfied.

13. Dependencies

a. Personnel Dependencies

    The testing phase requires two members as QA analysts, one as developer and one as a coordinator. Quick coordination with the development team is essential for addressing any issues.

b. Software Dependencies

    Testing is contingent upon the availability of the Visual Studio Community version 2022. Test automation relies on git Hooks, which must create a script to project repository.

c. Hardware Dependencies

Hardware Dependencies: The testing activities can be performed on both Windows PCs and MacBooks available within the team, ensuring compatibility and functionality across different operating systems.

d. Test Data & Database

The team needs anonymized, representative sample data for testing on both Windows PCs and MacBooks to ensure comprehensive coverage and functionality validation.

14. Risks

a. Schedule

Each stage has a tight timeline with numerous subtasks, posing a risk of incomplete tasks due to time constraints.

b. Technical

The source code is in C, whereas testing cases are in C++, which might cause compatibility or integration issues. Different IDEs for Windows and MacOS could lead to inconsistencies, and VPN dependency for GitHub access might introduce delays.

c. Management

Inefficient communication among team members and scheduling conflicts for meetings can delay decision-making and progress. Tracking members' work could be challenging, affecting project visibility.

d. Personnel

There's a risk of inadequate manpower or expertise, especially if the project requires specific knowledge in C or C++ that team members might lack. Absences or turnover could further strain resources.

e. Requirements

Incremental requirements introduced at each milestone with minimal descriptions can lead to misunderstandings or misaligned deliverables. Need us to start working on a new milestone in advance the class.

15. Tools

a. Visual Studio 2022

16. Documentation

a. A comprehensive Final Test Report
b. A Traceability matrix
c. Well documented function description
d. Completed code of program
e. Scrum Reports for each milestone.

17. Approvals

a. Tests except user acceptance test: approved by other members.
b. Program can get the same output as sample output.
c. Acceptance test: approved by the professor.