**Test Plan**

=======================================================================

|  |
| --- |
| Faaz Sherwani [Team Lead + Tech Lead] |
| Jubril Akolade |
| Iraklis Tsanachtsidis |
| Tarun Thomas |
| Frank Perez |
| Aum Patel |
| Rutarj Shah |

Table of Contents

Table of Contents ..........................................................................................................................................2

Introduction ..................................................................................................................................................3

Test Objectives .......................................................................................................................................... 3

Scope ......................................................................................................................................................... 3

Overview ................................................................................................................................................... 3

Glossary ..................................................................................................................................................... 3

Test Strategy ............................................................................................................................................. 3

Approach .......................................................................................................................................................4

Paths Constraint ........................................................................................................................................ 4

Path Selection/Selection Criteria .............................................................................................................. 4

Truck Specs ............................................................................................................................................... 4

Box Specs .................................................................................................................................................. 4

Test Coverage ............................................................................................................................................ 4

Software Component Validation .............................................................................................................. 5

Business Process Validation ...................................................................................................................... 5

Test Tools .................................................................................................................................................. 5

Test Types ................................................................................................................................................. 5

Test Data ................................................................................................................................................... 6

Test Plan ........................................................................................................................................................6

Test Team .................................................................................................................................................. 6

Test Schedule ............................................................................................................................................ 7

Environment ..................................................................................................................................................7

Features ........................................................................................................................................................8

Testing Procedure ..........................................................................................................................................8

Defect Management & Risks .........................................................................................................................9

Exit Criteria ....................................................................................................................................................9

Risks ............................................................................................................................................................10

Documentation ...........................................................................................................................................10

Test Approval ...............................................................................................................................................10

Business Requirements ...............................................................................................................................11

Introduction

This test plan will describe the strategies, processes, objectives, and methodologies used when testing our SFT221 Project.

Test Objectives

* Testing the software to ensure it meets the functional system requirements as outlined in the SFT221 project instructions.
* Testing the software to check for potential issues and bugs with the program.
* Ensure that the test results are properly documented and communicated to the team.

Scope

The test plan will cover the testing of all functions within the software. The plan will define the unit, integration, end to end, and regression testing plan. The plan will include testing of all system use case requirements outlined in the project instructions, as well as performance and security testing to ensure that the program is free of bugs.

Overview

The software we are testing is a program that will assist a local delivery company. This company has three different trucks that deliver on three different routes in the city. Our objective is to design a program that will calculate the shortest delivery path for the trucks and keep track of the truck’s current and maximum load capacity, the number of shipments, and the shipment destination.

Glossary

SFT221 Software Testing course at Seneca College.

Euclidean Distance It is the distance between two points. The formula is: 2√(a2+b2).

Test Strategy

We will be performing:

**Blackbox Unit Testing**: Where we will divide the test into two categories: testing general use-case scenarios, and exploratory testing which tests the limits of the code. It may take roughly 1 hour to design the tests, and 30 mins to execute them.

**Whitebox Unit Testing**: Where we will analyse the code for defects to find more test cases. It may take approximately 1 hour total to design and execute the Whitebox tests.

**Integration Testing**: Make another set of test cases once the functions are integrated into the program and create general-use test cases as well as exploratory tests for the software. It should take around 2 to 2.5 hours to fully design and execute the integration tests.

**Acceptance Tests**: These test cases will be designed after the software-requirements outlined in the project’s pdf. It will take roughly 1 hour to prepare the tests, and 20 mins to execute them.

**Approach**

Paths Constraint

* The path’s origin position is fixed to 1A.
  + All paths are fixed, unless a truck needs to make a detour in order to deliver a package outside of the path.

Path Selection/Selection Criteria

* Truck select is based on which truck can reach the destination in the shortest path.
* Trucks cannot drive through buildings.
* If distances between 2 paths are equal, the truck with the lighter load is chosen.
* Distance is calculated using the Pythagorean theory.
* Trucks cannot move backwards during its’ route.
* Destination must be within the delivery range, which is defined by the map’s row and column.
* Each grid position is listed from (x axis, y axis), i.e.: A1.

Truck Specs

* Maximum weight load: 1000kg
* Maximum volume: 36 metre cube

Box Specs

* Small Box: 0.25 m3
* Medium Box: 0.50 m3
* Large Box: 1.0 m3
* All boxes are perfect cubes, with same dimensions on all sides.

Test Coverage

* upper and lower bounds of each function
* upper and lower bounds of each parameter type
* misuse of function, such as: incorrect datatype parameter, and passing in NULL
* display/printing errors with incorrect, correct parameters
* performance checks
* check for unused parameters, variables, and functions.
* check if calculated distance is correct, i.e., no miscalculation.

Software Component Validation [will be done after functions are created

Test Tools

Visual Studio Community’s Native Unit Testing Module will be utilized, along with JIRA, GitHub, and Microsoft Office. The full details of these software are found under the Environment section.]

Test Types

**Functional tests**

* test each and all functions, initially with black box tests, followed up with white box
* Upper bounds of function specifications
* Lower bounds of function specifications
* Upper bounds of function parameter(s) datatype
* Lower bounds of function parameter(s) datatype

**Unit tests**

* Test the display related functions with black box tests, followed up with white box.
* White box testing used to formulate validation process.
* Validation of correct data (i.e., validate arguments passed into function)
* Black box and white box testing for all algorithm related functions.
* Black box testing to initially check the upper and lower bounds of algorithm specifications.
* White box testing to test limits of parameter datatype values.

**End-to-End Tests**

* test entire application, executing the executable produced by complier
* go through possible user stories (i.e. the sequence of how the user would use the

application from start to end)

* including all the inputs the user could put input

**Regression Test**

After each code update, preceding tests should be performed again to ensure any changes do not break the code.

**Test Data**

The test data will be generated using following process:

* brainstorm possible use-case scenarios.
* break down each scenario and determine what type of function is needed to implement the user action.
* Develop Blackbox test data based on function prototypes, and use-case scenarios.
* Develop Whitebox test data based on the implemented functions.

**Test Plan**

|  |  |
| --- | --- |
| Test Team Faaz Sherwani | Set Test cases, assign tasks. Attend meetings |
| Jubril Akolade | Help with editing, consulting, and setting up test cases. Attend meetings and give opinion |
| Iraklis Tsanachtsidis | Help with editing, consulting, and setting up test cases. Attend meetings and give opinion |
| Tarun Thomas | Help with editing, consulting, and setting up test cases. Attend meetings and give opinion |
| Frank Perez | Help with editing, consulting, and setting up test cases. Attend meetings and give opinion |
| Aum Patel | Help with editing, consulting, and setting up test cases. Attend meetings and give opinion |
| Rutarj Shah | Help with editing, consulting, and setting up test cases. Attend meetings and give opinion |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Schedule **Milestone** | **Description** | **P Start T.** | **P End T.** |
| MS3 | Document a set of Blackbox tests for the functions | July 12 | July 15 |
| MS3 | Blackbox test code | July 15 | July 16 |
| MS3 | Set up the function-test matrix | July 16 | July 21 |
| MS4 | Execute the Blackbox tests | July 23 | July 25 |
| MS4 | Create Whitebox tests based on implemented functions | July 25 | July 27 |
| MS4 | Execute the Whitebox tests | July 27 | July 28 |
| MS4 | Debug and re-execute tests | July 27 | July 28 |
| MS5 | Create integration tests | July 29 | July 31 |
| MS5 | Create acceptance tests | July 31 | Aug 2 |
| MS5 | Execute the integration tests | Aug 2 | Aug 3 |
| MS5 | Debug and re-execute tests | Aug 2 | Aug 3 |
| MS5 | Update test-matrix | Aug 3 | Aug 5 |
| MS6 | Execute acceptance tests | Aug 5 | Aug 7 |
| MS6 | Debug and re-execute tests | Aug 7 | Aug 8 |
| MS6 | Create the final test report | Aug 8 | Aug 10 |

**Environment**

|  |  |
| --- | --- |
| To perform the tests, we will need a machine capable of running Visual Studio Community. The following software are also required: **Visual Studio** | This is required to run the Unit Testing framework that comes with Visual Studio Community. Since this software only supports the Windows operating system, a Windows machine is required. |
| **Program’s source code** | Since we will be building and programming test cases on top of the source code, we will need to have the source code analysed to properly test it. Knowledge in C and the VS Community’s Native Unit Testing Module is required. |
| **JIRA** | Jira will be utilized to manage and keep track of our testing. Jira is a tool that a lot of teams use to organize projects, and for our test plan, we will use Jira to keep track of which software tests needs to be written, executed, and documented.  ] |
| **GitHub** | For each milestone, we will deliver any deliverables onto GitHub. This includes final test documentation, reports, and source codes for this project. |
| **Microsoft Suite** | We are using Microsoft Suite to document the test data and results, and to communicate with the team |

**Features**

All features will be tested, such as: being able to calculate which truck should be selected to deliver a shipment. The software should also test all the constraints which are listed in the Approach section.

Since all functions will be tested, no functionality will be left out in this test plan.

**Testing Procedure**

|  |  |
| --- | --- |
| The testing procedure is broken down into four steps: The Blackbox unit testing, Whitebox unit testing, Integration testing, and acceptance testing. **Blackbox Unit Test** | The Blackbox unit test will be done through the Native Unit Testing module in Visual Studio Community. We will create Blackbox test cases using the function’s prototype and create exploratory tests such as testing each variable in the parameter list. We will also create general-use test cases based on the program specifications outlined in the project’s pdf. We will also test the minimum and maximum limits of each variable. The pass criteria would be for the program to be smart enough to provide a descriptive error message if the user’s input is invalid, and to give the correct output if the user’s input is valid. This may take roughly 4 days to complete. |
| **Whitebox Unit Test** | The Whitebox unit test will also be done through the Native Unit Testing module in Visual Studio Community. After having access to the function’s code, we will be able to test the code more thoroughly by checking every possible path the function takes. The pass criteria would be for the program to be smart enough to provide a descriptive error message if the user’s input is invalid, and to give the correct output if the user’s input is valid. This may take 3 days to complete. |
| **Integration Test** | The third procedure is the integration test. This is when we merge the new functions with the source code and see if the program can work together with all the given functions and give the correct output. This may take roughly 2 to 3 days to complete. |
| **Acceptance Test** | The acceptance test will verify that the code runs, and that the correct output is generated. We will analyze and test all requirements and program specifications to finalize and our software. This should take around a day to complete. |

**Exit Criteria**

Testing is complete when all test data has been executed, and at least 95% of tests pass. Also, there should be no critical, high, medium, or low-level defects remaining.

**Risks**

There are 4 types of risks that might occur during the implementation of this plan:

**Schedule Risk:** Tight deadlines and unavailability of teammates may result in missing meetings, deadlines, or important information that may affect and compromise the quality of this project.

**Technical Risk:** Technical issues with teammate’s hardware may occur, but with GitHub, JIRA, and OneDrive, the risks are minimalized. If a component of the project becomes too complicated, this should be mentioned in the group asap to schedule a group meeting to work through any issues together.

**Management Risk:** May occur when leadership is not taken, and teammates feel confused with the direction of the project. Inaccurate time-estimation may also lead to compromises with the quality of the project.

**Requirements Risk:** The requirements and deliverables for the project should be properly analysed to ensure that this issue is minimalized and that teammates have a clear understanding on what the expectations are for each component of the project.

**Test Approval**

The testing results will be approved by everyone through a team meet-up. During the meet-up we will discuss the test results in-depth to ensure the testing was done correctly.

**Business Requirements [will be added after functions are made in m3]**