Test Plan Template

1. **Introduction**

Test Plan Objectives  
This test plan ensures that the truck delivery program works as intended. That is when a package is inputted by the user with attributes, weight, size and destination. The program will figure out which route will be the most efficient for delivering that package. Routes are decided by three trucks in a 25x25 square grid city map, with three predestined routes blue, green and yellow. The routes can detour to accommodate excess packages even when the route is not the most efficient in the event a truck is full. At the end when either the trucks are full or there are no more packages the map will be generated with the routes. The objectives include verifying the accuracy of the algorithm, measuring distances, and ensuring optimal assignment of packages to trucks. The expected result is a well-tested, functional, and efficient delivery system. The detoured route will also be outputted and should adjust to route conditions.

1. **Scope**

The test plan will cover the algorithm's correctness, with respect to the Euclidean distance calculations, truck capacity constraints, package assignment, and route diversion. It will not test the total distance of the route, gas usage and other customer interaction such as delivery time. Packages are either able to be delivered or will be delivered on another day if trucks are not available.

1. **Test Strategy**

**In this project, we will utilize a combination of testing methodologies to ensure the accuracy and validity of the truck delivery program. We will obtain test data from a variety of sources, including user input through referencing route data provided by the sample output/ map, as well as simulated city grid data that is outside the parameters of the 25 by 25 grid. The testing process will begin with a controlled test that is the sample output. Through that we hope to identify and fix critical defects. Then we will proceed with functional testing to verify that the core functions of the application are working within expectations. The testing strategy for the functions will include the following scenarios and more than likely some variation of edge cases. Coincidentally these function testing scenarios also cover most of the system testing requirement. Here are some of the test cases for the system/function testing:**

**3.1  
Package Input Validation Test Cases:**

1. **Valid package inputs (weight, box size, and destination)**
2. **Invalid package inputs (negative weight, unrealistic box size, or invalid destination)**
3. **Boundary values for package inputs (minimum and maximum allowed values which includes weight, size and out of grid values)**

**Truck Management Test Cases:**

1. **Packages within truck capacity limits of trucks (weight and volume)**
2. **Packages exceeding truck capacity limits (weight or volume) and the redistribution of packages.**
3. **The assignment of packages to trucks with optimal routes.**
4. **The most important thing to test here is truck assignment.**

**Distance Calculation and Shortest Path Determination Test Cases:**

1. **Comparison of optimal routes and the correctness of the routes, i.e. if a truck has space and package is part of the predestined route there should be no detour and the predestined route should be the default option. Detour routes should only occur when a truck is at capacity.**
2. **The most important part to test here is path diversion vs no diversion.**

**Output Generation Test Cases:**

1. **The expected output format and accuracy for valid input scenarios, referencing that of the sample provided.**
2. **The map out putted shows diverted routes that match with user inputted packages.**
3. **Truck routes are marked with the correct symbol for different truck assignments and path diversions. Also output for complex scenarios involving multiple trucks and path diversions resemble sample output.**

**Other test Execution and Reporting**

**The testing process will be executed following the defined approach, with each test case being documented and tracked. Test results, including any defects or issues identified, will be recorded, and reported to the development team for resolution. A final test report will be generated, summarizing the testing that was done, results, and any other outstanding issues.**

**3.2 Performance testing:  
Measurement of route efficiency/route planning. Ensuring the software correctly calculates the shortest path to the delivery destination while avoiding obstacles such as buildings at the same time determining if packages can be delivered within route parameters if they are not the program should output the appropriate response in an almost instantaneous manner. Any sluggishness in specific testing scenarios should be noted by the quality assurance team as bottlenecks to the program and will be rectified by the coding team.**

**3.3 Security Testing**

**The security testing of this program should ensure that vulnerabilities such as memory leaks and overflow are taken care of. Packages should also work within parameters i.e. weight, size not over limit and the destination stays within the 25 by 25 grid. The delivery of this program should not make it susceptible to SQL injection and other forms of attacks.   
  
3.4. Automated Test:  
  
Automation could be used to shorten the testing cycle such as in IPC144 when we used python to automate user inputs as opposed to user’s having to manually each individual variable which were slower and not as precise this not only ensures accuracy in testing but also improves speed.**

**3.5 Stress and volume Test:  
  
One of the core features of the program is the ability to stop accepting packages and begin delivery and output the mapped-out routes when the trucks are at maximum capacity. That is why it is imperative that testing must be done for scenarios when weight of trucks is at maximum capacity and size of trucks are at maximum capacity. Errors or crashes in such cases should be recorded by the quality assurance team.**

**3.6 Recovery Test:**

**The program itself does not save routes of packages. The recovery testing is based the programmer’s backup of the program code and documentation. Hopefully in case of unexpected events causing data loss, the backup plan will restore what was loss. For the it is vital that members backup data via teams, OneDrive, GitHub, and Jira.**

**3.7 Documentation Test:**

**This part involves mostly reviewing the project requirement outlines to ensure that the program accurately reflects the project outline in terms of functionality and usage. Updating testing plans and other documentation should also be vital in this step. As well as modifying documentation as needed to address any discrepancies or areas of confusion.**

**3.8 Beta Test:**

**This step is used to gather feedback from group members on the functionality performance and usability of the program. The first major implementation of the program will get an entire number change, incremental changes will receive a revision change. For example, 1 to 2 denotes a major change. While 1 to 1.1 is an incremental change in the program.**

**3.9 User Acceptance Test:**

**This will be the final version of the program that will be submitted for marking. The representative will run the program and see if the program meets expectation and sign off so that the program is ready for deployment, in our cases a grade will be given to us.**

**Traceability Matrix: A traceability matrix will be done to map system requirements to specific test cases, ensuring complete test coverage and easy tracking, specifically the matrix will follow the guideline of the sample included in the week 7 sample with the heading “ traceability-matrix-template.xlsx “**

**6. Test Schedule**

**a. The timeline of the test schedule should adhere to these dates.**

**If the due date for each milestone is on Tuesday of every week beta versions should be done by Friday providing ample testing time on the weekends and allowing revisions to be done on the following Monday.   
  
Milestone 1 & 2 due @ 11.59PM on Tuesday March 14**

**Milestone 3 due @ 11.59PM on Tuesday March 21**

**Milestone 4 due @ 11.59PM on Tuesday March 28**

**Milestone 5 due @ 11.59PM on Tuesday April 4**

**Milestone 6 due @ 11.59PM on Tuesday April 11**

**Milestone 7 due @ 11.59PM on Tuesday April 18**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SUNDAY** | **MONDAY** | **TUESDAY** | **WED** | **THURS** | **FRIDAY** | **SATURDAY** |
| **Test/document and correct** | **Final correction** | **MS due** | **Group meeting** | **Work on assigned roles** | **Beta for testing** | **Test/document and correct** |

1. **Control Procedures**
   1. 6.1 Reviews

Code reviews will be conducted to ensure that the developed solution adheres to the project requirements. This will begin as early as possible; dates can be found above.

6.2 Bug Review Meetings

At the moment group meeting is done on Wednesday before class at 3:30pm. If contact is necessary, team members are to communicate through teams. The object of the meeting is to discuss any reported issues, prioritize them and assign them to responsible team members for resolution.

6.3 Change Request

In cases where there is any discrepancies in the project requirements and the project itself or a need for change in requirements/ scope of project it will be noted and a formal change request will be submitted.

6.4 Defect Reporting

Defects discovered should be recorded in their respective outline templates:

traceability-matrix-template.xlsx

test-strategy-template-sample.docx test-strategy-template-sample.docx - Alternative Formats

test-plan-template.docx test-plan-template.docx - Alternative Formats

test-strategy-template-form.docx test-strategy-template-form.docx - Alternative Formats

test-strategy-template.docx test-strategy-template.docx - Alternative Formats

1. Functions to be tested are any functions that have not been covered in section 3. Specifically in the system testing procedure 3.1.
2. A. Resources:
3. Weekly notes/ course website
4. Project outline (the project pdf)
5. The professor (Robin Huang)
6. Visual Studio for coding.
7. Other C library resources available online

B. Responsibilities

Software Developers: Responsible for designing, building, and testing the solution.

Testers: Responsible for designing and executing test cases and providing feedback.

Project Manager: Responsible for overseeing the project and coordinating the resources.

1. Deliverables: