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 “**

1. **In this project, the testing environment requirements are as follows:**

**Hardware:**

**Desktop or laptop computer with a minimum of 4 GB RAM**

**High-speed internet connection**

**Mobile devices (if mobile testing is required)**

**Software:**

**Operating system: Windows 10, macOS, or Linux**

**Web browser: GoogleChrome, Mozilla Firefox, Microsoft Edge**

**Testing tools: Selenium WebDriver, JMeter, Postman**

**Test management tool: TestRail**

**Additionally, the testing environment should be set up with the latest version of the application to be tested, along with any necessary dependencies and configurations. It is also recommended that a staging environment is set up for testing, separate from the production environment, to prevent any consequences.**

5) Execution Strategy:

Entry and Exit Criteria: The entry criteria for testing will be defined as the completion of the development phase and finalizing the build for testing. The exit criteria will be met when all the test cases are executed, and the following conditions are met:

* No critical or high defects remain
* All medium and low defects have been reviewed and approved for release
* No cosmetic defects remain
* All test scripts are executed, and results are documented

Defect Severity Levels:

* Critical: These defects cause the system to crash or produce bad results, and there is no workaround.
* High: These defects cause the lack of program functionality and might have a workaround to give the desired functionality.
* Medium: These defects degrade the quality of the system, but often have a workaround to give the desired functionality.
* Low: These are minor errors that have minimal impact on functionality, such as unclear error messages.
* Cosmetic: These are issues that make the user interface less than optimal but still perfectly functional.

Test Reporting: Reports will be generated after each testing cycle and delivered to the project manager and development team. The report will include the following information:

* Number of tests conducted
* Number of tests passed and failed
* Description of the areas being tested
* Areas that are failing
* Severity of defects found
* Test coverage achieved
* Recommendations for improvement

Resolution: The quality assurance team will interact with the developers and project managers to ensure defects are resolved promptly. The communication channels will be through email, project management tools, and daily standup meetings. The quality assurance team will provide detailed defect reports to the development team, and they will work together to prioritize and resolve the defects. The quality assurance team will retest the resolved defects to ensure they have been fixed correctly.

**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

The group meeting is 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 team members responsible for resolution.

6.3 Change Request

In cases where there are any discrepancies in the project requirements and the project itself or a need for a 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.

10. Deliverables:  
 The deliverables of this project are outlined in the project assignment pdf.

For Milestone 1.

1. Complete team contract
2. Initialize got repository.
3. Setup Jira

For milestone 2.

1. Analysis of the problem
2. Series of data structures in header file
3. Test plan stored in repository.

For milestone 3.

1. Set of functions
2. Blackbox test documentations for the functions
3. Start writing Blackbox test code in repository.
4. Writing black box test code in repository.
5. Function test matrix added to repository.
6. Update Jira to show activities and progress.

For milestone 4.

1. Implement functions.
2. Implement Blackbox tests store in repository, execute results in Jira and debug
3. White box tests written and store in repository
4. White box tests implemented store in repository and executed in results in Jira and debugged
5. Updated function test matrix stored to repository.

For milestone 5

1. integration tests written and stored in repository,
2. integration tests written (store in repo), executed (results in Jira) and debugged.
3. acceptance tests written and stored in repository.
4. Updated function-integration-requirements-test matrix stored in the repository.

For milestone 6

1. Execute acceptance tests (results in Jira), and debug.
2. Updated function-test matrix stored to the repository.
3. Final Testing report listing tests conducted, bugs fixed and the final test passe

11. This section will outline the criteria for suspending or exiting the testing process. It's important to have clear guidelines for when to suspend or end testing to ensure that the software being tested is at an acceptable level of quality and is ready for release.

Some examples of suspension/exit criteria are:

The software has achieved a predetermined level of stability, performance, and functionality, and has passed all required tests. For this program, the accepted exit criteria should be when the trucks are at capacity, when there are no more packages than the user inputs in 0 0 x.

The number of critical or high severity defects reaches a predetermined threshold, indicating that the software is not ready for release.

The project timeline has changed, and testing must be suspended or cut short to meet the new deadline.

A major defect is found that requires significant rework of the software, and it is more efficient to suspend testing until the issue is resolved.

Resource constraints require testing to be suspended or reduced.

12. Resumption Criteria refers to the criteria that must be met for testing to resume after testing has been suspended or halted. This section should describe the conditions that need to be met before testing can resume. These criteria will depend on the reason why testing was suspended or halted in the first place.

For example, if testing was halted due to a critical defect, the resumption criteria may include the following:

* The critical defect has been fixed and verified by the development team. The fix has been integrated into the build and the build has been deployed to the testing environment.
* Regression testing has been performed to ensure that the fix did not introduce any new defects.
* The test results have been reviewed and approved by the quality assurance team.

It is important to define clear resumption criteria to ensure that testing is not resumed prematurely, which can lead to incomplete testing and potentially missed defects.

13. Dependencies

12.1 Personnel Dependencies: This section would outline the key personnel who will be involved in the testing process, including the roles and responsibilities of each member of the quality assurance team, as well as any dependencies on other teams such as developers, project managers, or business analysts.

12.2 Software Dependencies: This section would describe any dependencies on software that are required to conduct the testing, including operating systems, software tools, or other applications necessary for testing.

12.3 Hardware Dependencies: This section would outline any hardware requirements for testing, such as servers, computers, or mobile devices, including specifications for each item and any compatibility requirements.12.4 Test Data & Database: This section would describe any dependencies on test data or databases required for testing, including where the data is sourced, how it is stored, and any specific requirements for accessing or manipulating the data during testing.

Risks

14. Risks

13.1. Schedule: The testing phase may take longer than expected, causing delays in the overall project timeline.

13.2. Technical: The software may have complex interdependencies that are difficult to test, leading to incomplete test coverage.

13.3. Management: The test team may not have sufficient resources or budget allocated to properly execute the testing plan.13.4. Personnel: Key members of the testing team may become unavailable due to unexpected absences or resignations, impacting the testing schedule and quality.

13.5. Requirements: The requirements for the software may be unclear or incomplete, making it difficult to develop comprehensive test cases and identify defects.

15. Tools used in test planning for this project include but are not limited to:

a. Communication/ documentation: Jira, Microsoft teams, GitHub.

b. Debugging/Coding: primarily using visual studio and logging with specific linkers.

c. Dynamic analysis in case code requires memory allocation. Memory leak testing program will be implemented.

d. automation: to assist the quality assurance team automated keystrokes such as python is recommended to improve accuracy of test inputs.

16. Documentation will be maintained throughout the software development process using templates provided following the test plan guidelines/ schedules etc.

Some highlights in documentation are the test plan for outlining the strategy and approach we are taking to complete the deliverables within the scope in a timely manner.

The system and function test cases must have detailed descriptions following the guidelines such as input data expected output, actual output, recorded bugs and suggested fixes. Outlined in past assignments.

Test results and reports should be done with extra care, since it is the focus of this course. Documentation of test results, including pass/fail status, any identified issues, and recommended actions to address them.

17. Approvals  
 The approval process will be done by the stakeholders, the group members, who are responsible for meeting deadlines, and reviewing the quality of all group members' work and documentation change where they deem necessary. The key here being the documentation of the changes especially on GitHub and Jira.

Finally, any change that differs from the project outline would need approval by the client, the professor, to sign off. So that the program can be delivered to the end users.