#SFT211-NCC GROUP 6 TEST PLAN  
  
**# Aims of the Testing Plan**

We aim to investigate the following aspects with our tests:

* Ensuring that the weight and dimensions of packages are within the truck's capacity.
* Verifying the accuracy of shortest path computation for each package.
* Inspecting the correct route display for the user.
* Counting the number of shipments accurately.
* Checking that user input correctly determines package weight, size, and destination.
* Validating user inputs to prevent program crashes.

**## Part 2. Extent of Testing**

We have devised a comprehensive testing strategy that covers every function within the software.

Here is what our testing includes:

**Unit Testing**

* Detailed examination of each function and its output to confirm alignment with functionality requirements.

**Integration Testing**

* Checking the seamless interplay among our functions.

**Regression Testing**

* Confirming the successful execution of all previous tests after any modifications.

**End-to-End Testing**

* Validating the code's smooth operation from the initial to final stages, simulating real-world scenarios.

The following aspects are not part of our testing scope:

* Actual delivery performance.
* Program performance.
* Program compatibility with various operating systems.
* Accuracy of information provided, such as addresses, package sizes, etc.

**## Part 3. Testing Tactics**

Our testing approach includes the following:

**Blackbox Unit Testing**

* Pushing the code to its limits in regular use case scenarios.
* Crafting these tests will take at least 2 hours

**Whitebox Unit Testing**

* Thoroughly review the code for any hidden bugs.
* This process may uncover additional test cases.
* Designing white box tests at least 2 hours

**Integration Testing**

* Verifying the synergy between groups of interconnected functions.
* Designing this testing phase will take at least 2 hours.

**Acceptance Tests:**

* Creating tests based on the software requirements outlined in the project document.
* Designing and executing these tests will take 2 and 1 hour respectively.

All test cases will be circulated among team members for review before execution. We will collectively inspect the criteria and potential errors before proceeding.

**## Part 4. System Requirements**

**### Hardware:**

* A x64 processor.
* Wifi adapter or ethernet port.
* Stable internet connection.
* At least 8GB of RAM
* At least 10GB of free hard disk space.
* A video card or integrated graphics chip compatible with a modern monitor.

**### Software:**

* Visual Studio IDE.
* Source control integration via GitHub.
* A web browser.
* A current operating system equipped with the necessary frameworks to run Visual Studio.

**## Part 4b. Preparing our Testing Environment**

* Update your computer's software.
* Install Visual Studio Community Edition.
* Integrate log4c into your project file as needed.

**## Part 5. Execution Strategy**

1. Collaboratively devise test data covering every main function of the program.

2. Run this test data through black box, white box, and individual function testing until all tests pass.

3. Conduct integration testing of functions in pairs once confident in their individual functionality.

4. Perform functional tests and end-to-end testing to verify the expected output.

**## Defect Management and Potential Risks**

When reporting bugs, we will provide the following details:

1. A clear description of the problem.

2. The file where the issue originated.

3. The precise line number where the bug occurred (if known).

4. The bug's level of severity.

Bugs will be classified based on the following severity levels:

**### Critical**

A bug that crashes the program or severely impairs its functioning.

These bugs are the highest priority and may cause project delays. They must be resolved urgently.

**### High**

Bugs that disrupt some program functions but allow others to operate normally.

**### Medium**

Bugs that lower the system's quality (i.e., produce incorrect output) but do not affect functionality.

**### Low**

Bugs that lead to unclear error messages or minor issues with minimal impact on functionality.

**### Cosmetic or Informational**

Minor issues that affect the user interface but do not disrupt the program's functionality.

**## Test Reporting**

Our final testing report will be presented via an Excel spreadsheet listing all conducted tests and their outcomes.

We will hold a team meeting to discuss the tests, ensuring everything was conducted correctly before submission.

Each executed test will include the following information:

* The function under test.
* The input data.
* The expected outcome.
* The actual outcome.
* Test success status.
* Additional comments (optional).

**## Part 6. Testing Timeline**

To Be Determined.

**## Section 7. Control Procedures**

We will use the following control measures to manage our testing activities throughout the development lifecycle:

**### Reviews**

Whenever a new phase of testing is completed, the results will be reviewed by other team members.

**### Defect Reporting**

Any bugs found in the program will be logged in a Jira ticket until they are resolved.

**### Bug Review Meetings and Progress Monitoring**

When bugs are assigned for resolution, relevant members will collaborate to share insights about the issue and track the progress of its resolution.

For example, if someone discovers a bug, they will share their findings and thoughts with the person assigned to resolve it.

**### Change Request**

Change requests are formal proposals to modify parts of the software. They will be addressed on a case-by-case basis depending on the severity of bugs found and the required solutions for proper function integration.

**## Section 8. Functions to be Tested**

A list of all functions that need testing.

To Be Determined (basically all functions)

**## Section 9. Resources and Responsibilities**

**Task/Resource management: Jira**

* This is where all tasks will be assigned and progress will be tracked.
* Documents, including GitHub file links, will be posted here.

**Communication: Discord**

* This is our chosen platform for communication, in addition to Jira.

**Codebase and Documentation: GitHub**

* All files, with detailed descriptions of each commit, will be backed up on GitHub. All documentation, including test data and results, will be stored here too.

**Informational Resources**

* SFT211.ca
* Seneca Blackboard
* Stack Overflow

**## Part 10. Deliverables**

**### Milestone 2**

* Jira setup
* Completed test plan document
* Unit testing case creation
* Data structures

**### Milestone 3 - To Be Determined**

**### Milestone 4 - To Be Determined**

**### Milestone 5 - To Be Determined**

**## Part 11. Suspension/Exit Criteria**

Testing efforts will be suspended under the following circumstances:

* Discovery of a crash-causing bug that terminates the program
* Incorrect return values
* Failure to validate user input correctly

Any team member has the authority to halt testing, and all discovered bugs must be documented in Jira and the relevant testing document.

**## Part 12. Resumption Criteria**

Testing efforts will resume once the cause of the crash has been rectified and properly tested.

Escalation:

If the bugs cannot be fixed within a reasonable timeframe, they should be noted in Jira, and other team members should be alerted via Discord for additional assistance.

**## Part 13. Dependencies**

**### Personnel Dependencies**

* Availability of team members at agreed times
* Team members' ability to comprehend tasks and follow through with assignments
* Team members' knowledge of how to complete assigned tasks effectively

**### Software Dependencies**

* Jira
* Discord
* GitHub
* Visual Studio Community Version
* Web browser

**### Hardware Dependencies**

* Functional computer
* Internet connection
* Keyboard
* Monitor

**## Part 14. Risks**

**### Scheduling Risks**

* Lack of availability of team members within the tight milestone deadlines
* Insufficient testing time due to late work submission, potentially resulting in missed bugs that compromise the program's integrity.

**### Technical Risks**

* Potential inability to access or use GitHub, Jira, or Discord during outages
* Technical difficulties preventing team members from using computers or accessing the internet

These risks can be managed by having an effective plan in place. Communication through Jira, Discord, and Seneca emails will allow for the coordination of alternative methods to complete tasks in case of service or hardware failures.

**### Management Risks**

* Lack of leadership, clear communication, and misunderstandings may cause delays, missed criteria, and compromised project quality.

**### Personnel**

* Team members should communicate promptly if they are unable to complete assigned work on time, are ill, or experience hardware/software failures.
* If team members have concerns or lack knowledge to complete specific tasks, they should seek help and guidance from other team members.

**### Requirements**

* Full comprehension of the requirements for each project milestone is crucial to avoid missing them and failing to meet client expectations.
* If requirements change during the work process, additional time and personnel may be required to accommodate the updated work, or else the deadline may be at risk.

**## Part 15. Tools**

We will utilize the following tools during testing:

* Visual Studio's native unit testing
* GitHub
* Jira
* Excel (for test cases)
* Visual Studio

**## Part 16. Documentation**

Various documents required throughout the testing process:

To Be Determined

**## Part 17. Approvals**

Throughout the testing process, results will be approved by all team members in a team meeting. Detailed discussions will take place to ensure proper handling of the results.