Test Plan Template

1. **Introduction**
   1. Test Plan Objectives
      1. **The purpose of this test plan is to ensure that the delivery algorithm accurately allocates the package to the truck based on its capacity, proximity to the destination, and available routes.**
2. **Scope**
   1. **Assigning packages to trucks based on weight and volume capacity.**
   2. **Determine the truck closest to the destination.**
   3. **Calculate the shortest path from the nearest point to the destination of the truck path.**
   4. **Handling of the case where the truck cannot reach its destination.**
3. **Test Strategy**

**Our comprehensive testing procedures are meticulously designed to ensure the absence of any functional defects and to evaluate the overall quality of the system thoroughly. These tests encompass a wide range of functional assessments aimed at identifying even the most critical errors. Furthermore, our testing process serves as a robust documentation tool, meticulously recording any issues within the program for further analysis and improvement. In particular, the user acceptance test is a pivotal phase, determining the program's user-friendliness and its ability to seamlessly integrate into the user's environment. During this phase, we assess whether the program not only meets the technical requirements but also aligns with the user's expectations and needs. This user-centric approach is fundamental to our testing philosophy. All our testing efforts are dedicated to validating user satisfaction and ensuring the seamless functionality of the system. We believe that a user's experience is at the heart of any successful software, and our rigorous testing procedures are geared towards delivering not just functional software, but delightful and reliable user experiences.**

**Required Exploratory Tests**

* Scenario-based testing: In this approach, our testers craft realistic scenarios that mirror real-world situations, enabling them to simulate user interactions. By exploring various paths, functions, and edge cases, we delve deep into the intricacies of the software. This method not only helps us identify potential issues but also ensures that the software performs seamlessly under diverse conditions.
* Document testing: Our meticulous document testing process ensures harmony between the provided documentation and the actual software implementation. We focus on verifying that the documents offer precise and comprehensive instructions to users, guiding them effectively. Moreover, we guarantee that the documents accurately depict the system's functionality and behavior. This meticulous verification process helps us maintain clarity and consistency, enhancing the user experience and understanding of the software.

**Required Functionality Tests**

* User Acceptance Test: This crucial testing phase is dedicated to validating whether the program aligns with the end users' expectations and requirements. During this process, real-world scenarios are meticulously tested, ensuring that the program accurately assigns packages to trucks, calculates distances flawlessly, and delivers packages to their intended destinations. By meticulously verifying these tasks, we guarantee that the software operates in the real world as seamlessly as envisioned.
* System Test: This comprehensive testing stage focuses on verifying the system's overall functionality, ensuring that it operates cohesively as a unified whole. Through rigorous testing, the integrated system is examined to confirm that all components work harmoniously, meeting the specified requirements. Some specific tests, such as Performance Testing and Integration Testing, are included in this phase to guarantee that the system performs optimally and integrates flawlessly.

Moreover, we recognize the importance of ensuring individual components or modules function correctly and integrate seamlessly. Therefore, specific functionality tests are conducted using Unit Testing or Assertions, complementing the broader scope of system testing. This meticulous approach not only validates the system's integration as a whole but also assures that each component operates accurately, contributing to the system's overall efficiency and reliability.

**Unnecessary testing**

* Security Test: Security is very important in a project, but this test is unnecessary because there are no security requirements in the project description.
* Stress and Volume Test: The project description does not specifically state the requirement for testing the system's scalability or its performance under extreme loads. If the system isn't anticipated to manage significant volumes of packages or trucks, stress and volume tests might not be applicable.
* Recovery Test: It was determined that this test was not necessary to proceed because there was no specific failure scenario or recovery method requirement in the project description.

**Test Design Process and Conduct Overview**

* Understand Requirements: The person conducting the test identifies and understands the project's requirements, detailed functions, and documents related to the project in order to pinpoint the project.
* Build a traceability matrix: A visual representation of the interrelationships between project requirements, designs, test cases, and other project elements. This matrix allows to track what design or development stages each requirement was implemented and in what test cases it was validated.
* Prepare Test Cases: The test cases are created based on identified requirements and related test scenarios. Each test case defines necessary input data, expected results, and specific steps or conditions. It ensures comprehensiveness by encompassing positive and negative scenarios, boundary conditions, and edge cases.
* Test Case Review: In order to improve the quality of the test case, it is reviewed by members other than those who conduct the test. The member reviewing the test provides feedback on the clarity, accuracy, and functionality of the test. The review process helps identify problems and improvements to be addressed.
  1. **You could describe the test design process and give an overview of how it will be conducted. You could provide a broad overview of** 
     1. **how to understand requirements,**
     2. **build a traceability matrix,**
     3. **prepare test cases,**
     4. **and have them reviewed by another member of the quality assurance team.**

1. **Environment Requirements**
   1. Hardware:
      1. Windows: The Windows test environment requires a computer based on a Windows operating system with sufficient performance to run the test environment.
      2. Mac: The Mac test environment requires a computer based on a Mac operating system with sufficient performance to run the test environment.
   2. Software:
      1. Windows: Testing is done with the latest version of Windows (e.g., Windows 10) with the required dependencies and software tools installed.
      2. Mac OS: Testing is performed using the latest version of mac OS with the required dependencies and software tools installed.
2. **Execution Strategy**
   1. Entry Criteria: If the software to be tested is built and compiled without problems, start testing.
   2. Exit Criteria: If the test performed does not indicate a serious error or defect, the test is considered complete.

severity levels of:

* + 1. **critical** which cause the system to crash or produce anomalous results,
    2. **high** which causes lack of program functionality and might have a work around,
    3. **medium** which is a bug which D crates degrades the quality of a system but often has a work around to give the desired functionality.
    4. **Low** which might be an unclear error message or some other minor error that has minimum impact on functionality.
    5. **Cosmetics** is something that makes the user interface less than optimal but still perfectly functional.
  1. **Test Reporting**
     1. The report has been prepared to document the progress and results of the test. Therefore, each test should provide information on the type of test, the number of passes, and the number of failures. If a test fails, it should include an explanation of the failure. The report is then delivered to the project team's manager, development team, and quality assurance team. The development team is tasked with resolving program bugs, defects, and improvements. Regular communication with the quality assurance team occurs through meetings and emails to ensure cooperation.

1. **Test Schedule**
   1. **Estimated Testing: Testing is expected to take about two weeks to complete the test.**
   2. Estimated Complete: Testing will be completed by the end of the third week after the start of the testing.
2. **Control Procedures**
   1. **7.1** Reviews: Regularly conduct evaluations to gauge the advancement of testing and verify that testing operations align with the project's requirements and goals. These evaluations include essential stakeholders such as team members, project managers, and other pertinent individuals. The aim of these assessments is to offer input, pinpoint any shortcomings or issues in the testing procedures, and implement essential modifications to enhance the overall quality of the delivery management system.  
      **7.2** Bug Review Meetings: Scheduled meetings are conducted to deliberate on and rank identified problems or faults. In these meetings, test and development teams collaborate to examine reported glitches, assess their severity and impact, and assign responsibility for resolving them. These bug review sessions play a crucial role in ensuring transparent communication, monitoring the advancement of bug repairs, and promptly resolving recognized issues.  
      **7.3** Change Request: In the testing phase, a system modification or enhancement might be necessary, prompting a change request. These alterations can stem from stakeholder feedback, identified enhancements, or shifts in project requirements. The change request follows a formal procedure, involving documentation of proposed changes, evaluating their impact, obtaining approval from pertinent stakeholders, and executing approved modifications. This process guarantees that required alterations can be documented, reviewed, and incorporated accurately, minimizing disruptions to the project schedule.  
      **7.4** Defect Reporting: Reporting defects is vital in the testing phase, and testers will document identified issues or glitches using a standardized defect reporting form. This report includes particulars like defective description, reproduction steps, expected behavior, actual behavior, and supporting attachments. Defects are categorized by severity such as critical, major, and minor and prioritized for resolution. This defect reporting process efficiently tracks, communicates, and resolves identified problems.
3. **Functions To Be Tested -**

**8.1** **Existing functions:**

populateMap(): Generates a map to facilitate route planning and navigation. Testing involves verifying the correct creation of the map.

getNumRows() and getNumCols(): These get the number of rows and columns in the map. Testing involves confirming they return the accurate row and column counts.

printMap(): Prints the map with visual symbols representing different entities. Testing this function involves confirming the correct display of the map, symbols, and their alignment.

getBlueRoute(), getGreenRoute(), and getYellowRoute(): Generate predefined routes for different trucks. Testing includes if these routes are correctly generated.

addRoute(): Adds a given route to the map. Testing confirms the addition of the route and checking for any errors on the map.

distance(): Calculates the distance between two points. Testing involves verifying the accuracy of distance calculations.

shortestPath(): Finds the shortest path between two points on the map. Testing includes confirming it generates the shortest route.

getPossibleMoves(): Determines the possible moves from a specific point. Testing involves validating if it correctly identifies available moves.

getClosestPoint(): Identifies the closest point from a point. Testing includes if it verifies the accurate, closest point.

eqPt(): Checks whether two points are equal. Testing involves validating the correct comparison of two points.

**8.2. New header:**

calCapacity(struct Size max, struct Size loaded): Computes the capacity available in a truck, considering its maximum capacity and the size of boxes already loaded.

validateInput(struct customerInput input): Verifies whether the customer input is valid.

findTruck(struct customerInput input): Determines the truck that can accommodate a particular shipment based on the customer's input.

The following should be tested to see if the algorithm works properly.

**8.1** Shipment Allocation: This function will store the user input and be tested to ensure that shipments are allocated correctly to trucks based on weight, box size, and destination. Test cases will cover various scenarios, including valid and invalid inputs, truck availability, and weight and volume capacity.

**8.2** Shortest Path Calculation: This will verify if it correctly determines the shortest path between two points while avoiding buildings. Test cases will include scenarios with different starting and destination points, the buildings obstructing the way, and corners.

**8.3** Capacity Calculation: This will validate the available capacity of each truck. Test cases will cover scenarios with different weight and box size combinations, reaching the maximum weight or volume, and handling trucks with diverse limitations.

**8.4** Output Message Generation: This will test if it generates accurate and informative messages regarding the truck selection, diversion paths, and other relevant information. Test cases will cover different allocation scenarios and edge cases to validate the correctness and clarity of the output messages.

1. **Resources and Responsibilities**

**9.1.** **Resources:**

**a.** Testers: A team of dedicated testers who will execute the test cases, document the results, and report any issues or defects. A scheduled testing, when the testers will start and end, and the expected deliverables, will be decided.

**b.** Test environment: A list and description of the necessary hardware and software components, and the simulated data will be needed to support the testing activities. Find out any special training needed for the testing team.

**9.2 Responsibilities:** Efficient collaboration, understanding of requirements, clear communication, and fast issue resolution will be required for a more successful testing phase.

**a.** Testers: Testers will be responsible for understanding the software requirements thoroughly to develop effective test cases that cover various scenarios. Testers need to run the test cases against the software to identify bugs, issues, or areas that need improvement. They need to document and report defects, including details such as expected behavior and observed results.

**b.** Developers: The development team should understand the test scenarios to create high-quality software that meets requirements, and perform code reviews, identifying potential issues before the testing phase. Developers are also responsible for working on fixing defects reported by testers, assisting in understanding issues, and conducting thorough unit testing to identify issues. Developers should cooperate with testers, providing necessary clarifications and addressing identified issues promptly.

**c.** Project Managers: Project managers are responsible for developing a comprehensive test plan, including timelines, resources, and responsibilities, coordinating with the testing team, developers, and other stakeholders. Project managers will make sure that testers have the necessary tools and support and identify potential risks in the testing process and planning strategies. The project managers will also oversee the testing activities to ensure the software meets quality standards. The project manager will facilitate communication between different team members, confirming proper coordination between the testing and development teams, and provide necessary support and resources.

**d.** Team: The whole team needs to collaborate effectively, ensuring testers have the necessary information about changes in the software. Team should provide support during the testing phase, offering guidance and assistance as needed.

1. **Deliverables**

As part of the test plan for the local delivery company's algorithm:

**10.1.** Algorithm Testing Documentation: Detailed documentation outlining the testing steps, including test scenarios, test data, and expected outcomes based on various delivery and diversion scenarios.

**10.2.** Input Validation Report: A comprehensive report on the validation checks for input data, including valid and invalid inputs for weight and box sizes, and different delivery destinations and accuracy. It should identify and handle different types of errors.

**10.3.** Package Allocation Results: The algorithm should correctly assign packages to trucks based on available space, distance to destination, and diversion required.

**10.4.** Calculation of the Shortest path: The algorithm should calculate the shortest path from the nearest point on the truck's route to the destination. Testing the shortest path algorithm to confirm its accuracy in determining the most efficient routes and considering buildings and other constraints in delivering packages.

**10.5.** Path Diversion Verification and Output: The path diversion calculated by the algorithm for each truck and verifying the selected path. If a truck diverts from its initial route, a detailed path description needs to be provided.

**10.5.** Simulation Output: A sample output depicting multiple test cases and the expected shipping routes. This includes the weight, box size, and destinations and demonstrates the paths taken by different trucks. The algorithm should handle cases where a truck cannot reach the destination due to obstacles.

**10.6.** Final Report: A conclusive report summarizing the testing results, displaying the effectiveness of the algorithm in selecting trucks and delivering packages based on weight, available space, and proximity to the destination. Any updates or enhancements in the system documentation based on the discovered issues and improvements during testing.

**10.7.** Testing Sign-off: The final approval confirming the completion of testing and the readiness of the system for deployment.

These deliverables will ensure a well-tested and validated delivery system, ready for successful deployment and operational use.

Test Execution Reports and any Defect Reports can be also delivered.

1. **Suspension / Exit Criteria**

Suspension Criteria will be algorithm unavailability or errors like truck route defects or truck diversion errors here. Suspension and exit criteria are integral parts of a testing process. They help define when to temporarily stop or permanently conclude testing activities. This will be when test environment, tools, or required data are unavailable or compromised or when the severe defects are found. Suspension criteria help manage the testing flow, allowing resolution of issues that prevent productive testing. Exit criteria set clear guidelines for determining when the testing phase is completed, and the software is ready for the next steps in the development life cycle.

**11.1.** If the algorithm fails to assign packages to trucks based on available space, distance, and diversion requirements successfully.

**11.2.** If the algorithm fails to calculate the shortest path correctly or if the algorithm encounters errors.

**11.3.** If the algorithm cannot handle cases where a truck cannot reach the destination due to buildings.

**11.4.** If the algorithm does not print the necessary information accurately.

**11.5**. Shut down the algorithm if the baggage weight exceeds 1000 kg.

1. **Resumption Criteria**

After fixing the bugs in the code that caused the testing to be suspended, testing should resume from that point.

Make sure the functionality of what is to be test is logically correct. Meaning latest code should be able to continue testing.

Make or update test cases in order to resume testing given that an update has been made to the functions or methods used in the program.

1. **Dependencies**  
   **13.1** Personnel Dependencies

Team members who are responsible for writing code must be available.

Ensure team members who will be conducting the testing are available during the testing phases.

Ensure team lead is available to coordinate all testing tasks.   
**13.2** Software Dependencies

Ensure software to be used for testing is available.

If the software is out of date, make sure it is updated before testing begins.

Ensure we have the program to be test readily available.  
**13.3** Hardware Dependencies

Make sure the computer or laptop to be used for the testing is in good condition.

Ensure the computer is compatible with the latest version of the software to be used for the testing.  
**13.4** Test Data & Database

Identify the data we need for the testing.

Ensure the test data is accurate and meets the test requirements.

Test data should cover a wide range of scenarios, including different package weights, sizes, destinations, and consider edge cases.

1. **Risks**  
   **14.1.** Schedule

Make sure right amount of time is scheduled for testing phases.

Ensure all testing phases are started on time and completed within the scheduled period.

**14.2.** Technical

Address issues that have to do with hardware requirements and challenges.

Also identify the issues with software that might impact the testing process  
**14.3.** Management

Look out for poor project management that would affect the testing phases.

Ensure there is strong management support to enhance the testing process.  
**14.4.** Personnel

Make sure the whole team is on the same page and communicate more often.

Mak sure team members are willing to put in work to have an effective testing.

Provide support for each team member to ensure effective testing.  
**14.5** Requirements

Make sure all requirements are up to date.

Any update to the initial requirements must be discussed with the whole team and attain approval before moving forward with it.

Make sure all team members are aware of any new requirements.

1. **Tools**

Esure the availability of tools that would be used in the testing phase. Choosing the right tools will go a long way to affect our software testing.

VS code can be used to test the functions or program being created.

To be able to track testing being done we need to make sure we have such tools.

Project management tool such as JIRA can be used here.

1. **Documentation**

**9.1 Documentation Purpose**

Documentation plays a vital role in maintaining quality, process control, organization and clarity throughout the testing process. It serves as a point for team members and course mentor to be sure that everyone is aligned with the test plan's objectives and processes.

**9.2 Document Control**

* **Test Plan Version:** 1.0
* **Last Revision Date:** November 6, 2023
* **Revision History:**
  + Version 1.0 (November 6, 2023): Initial draft

#### **9.3 Document Ownership**

The test plan and associated documentation are owned and maintained by the Team 1. All 6 members are responsible for updating and maintaining these documents.

#### 9.4 Document Access

Access to this test plan and related documents is restricted to authorized team members and Software Development Testing course mentor involved in the project process. Access permissions will be controlled to maintain data integrity and security.

#### 9.5 Document Repositories

The test plan and associated documents will be stored in the [FALL23-SFT221-NDD-1](https://github.senecacollege.ca/thong14/FALL23-SFT221-NDD-1)/[Documents](https://github.senecacollege.ca/thong14/FALL23-SFT221-NDD-1/tree/main/Documents)/[Testing](https://github.senecacollege.ca/thong14/FALL23-SFT221-NDD-1/tree/main/Documents/Testing)/[TestPlan](https://github.senecacollege.ca/thong14/FALL23-SFT221-NDD-1/tree/main/Documents/Testing/TestPlan)

document repository, accessible to team members through the project management tool.

#### 9.6 Related Documents

The following documents are related to this test plan and can be found in the project repository:

* Scrum reports

#### 9.8 Distribution

The test plan and associated documents will be reviewed by the relevant team members at the beginning of the testing phase.

1. **Approvals**

Reviewed by the group members:

Natalya Pak

Farouk Alhassan

Hyunjoo Han

Jenna Moon

Taehwa Hong

This test plan is awaiting approval from the course mentor Rashmi Shikhariya.