Test Plan

1. **Introduction**
   1. Test Plan Objectives
      1. **Our objectives serve as the safeguard for the local delivery system's quality and reliability. It scrutinizes the consistency of decision-making, emphasizing the verification of flawless system operation, understanding performance and capacity constraints, and ensuring overall stability. With a broad commitment to user satisfaction, the test plan aspires to construct a resilient system. This system aims to seamlessly distribute packages, maintain consistent performance, and deliver a pleasing user experience. Any identified issues will be meticulously addressed to enhance the system's functionality and finesse.**
2. **Scope**

**Testing:**

* 1. **Performance Evaluation:** 
     1. **Check how well the system responds and works efficiently.**
     2. **Simulate different situations to see how it handles shipments and assigns them to trucks on time.**
  2. **Functionality Validation:**
     1. **Test if the system correctly assigns packages to trucks based on available space and distance to the destination.**
     2. **Make sure it figures out the shortest path, considering obstacles like buildings.**
  3. **Capacity Testing:**
     1. **Test each truck's and the whole system's capacity.**
     2. **See if the system can handle packages of different sizes and weights within set limits.**
  4. System Stability Assessment:
     1. Check how well the system holds up and recovers from unexpected issues.
     2. Look at how it handles errors and disruptions.

Not testing:

* 1. External Dependencies:
     1. Don't dig too deep into testing third-party connections. Just make sure they work well with our system.
  2. Security Testing:
     1. Don't extensively test for security issues. However, keep security in mind during regular testing.
  3. Hardware Assessment:
     1. Don't test the physical parts like trucks. Focus mainly on how the software works.
  4. Network Infrastructure:
     1. Don't check the underlying network structure. Concentrate on how well the software functions.

1. **Test Strategy**
   1. **Approach:**

**The test approach will encompass a combination of manual and automated testing techniques to ensure thorough coverage of the system's functionality, reliability, and usability. Testing will be performed iteratively throughout the development lifecycle, with continuous communication between development and testing teams to address issues promptly.**

* 1. **Levels of Testing:**

**i. Exploratory Testing: Initially, exploratory testing will be conducted to identify critical defects and usability issues.**

**ii. Functional Testing: Once critical defects are addressed, functional testing will ensure that all prime functions of the application are delivered correctly.**

**iii. System Testing: Comprehensive system testing will be performed to validate the entire system against the specified requirements.**

**iv. User Acceptance Testing: Finally, user acceptance testing will be conducted to ensure that the system meets the user's expectations and is ready for deployment.**

* 1. **t**est design process:

i. Understanding Requirements:

- Begin by thoroughly understanding the requirements provided by the stakeholders or project managers.

- Ensure clarity on functional and non-functional requirements.

- Collaborate with stakeholders to clarify any ambiguities or uncertainties.

ii. Building a Traceability Matrix:

* Create a traceability matrix to map requirements to test cases.
* dentify each requirement and its corresponding test case(s).
* This matrix helps ensure that all requirements are covered by test cases and vice versa.

iii. Preparing Test Cases:

* Based on the identified requirements, create detailed test cases.
* Test cases should cover both positive and negative scenarios.
* Include steps to execute the test, expected results, and any prerequisites or test data required.
* Test cases should be written in a clear and understandable manner to facilitate easy execution.

iv. Reviewing Test Cases:

* Have the prepared test cases reviewed by another member of the quality assurance team or by stakeholders.
* Reviewers can provide valuable feedback on the completeness, accuracy, and clarity of the test cases.
* Address any identified issues or suggestions for improvement before finalizing the test cases.

1. **Environment Requirements**
   * 1. Processor: Minimum 1GHz; Recommended 2GHz or above.
     2. Memory: Minimum 1GB; Recommended 2GHz or above.
     3. Hard Drive: Minimum 32GB; Recommended 64GB or above.
     4. Screen resolution: 1024 x 768 or above.
     5. Keyboard and Mouse.
   1. Operating Systems:
      1. Windows 10 or 11 (64bit)
2. **Execution Strategy**
   1. I. The entry criteria:

- Check if all environmental requirements are met before executing the test program.

- Make sure to agree on the test strategies for all stakeholders.

ii. The exit criteria:

- exit a test if it passes 90% of the scripts and if you can get expected output even though there are some minor warnings or issues.

* 1. You can describe the severity of defects in this section and break them down into severity levels of:
     1. **critical** System crush, Security flaws.
     2. **high** overwritten issue, data loss, unexpected output with major functionality.
     3. **medium** functionality issue with minor functions which is not affecting often to the desired output.
     4. **Low** no problem with expected output, but there are some warning signs.
     5. **Cosmetic** issues with design-area (spaces, typo, etc).
  2. **Test Reporting**
     1. - It should produce the report including

1) the percentage of passing the scripts, 2) Main issues, 3) Suggestion for the issues.

How often: once every major function is made.

Report to: Team leader (Project Manager)

* + 1. The report should include the issued lines and functions, purpose of the test unit, input, expected output, actual output, statement of pass or fail (with a total percentage of the pass), suggestion to fix issues, tester’s info, and report to.
  1. After receiving the report from QA team, QA team and developers can prioritize the issues based on the severity levels and impact of the issues. By doing this, they can

Share insights of the issues. To do this, they can utilize GitHub, Gira, Teams, etc. to communicate with each other. After fixing issues, the QA team can make steps for test scenarios again. Until it meets the exit criteria, they must do the same process as a loop.

1. **Test Schedule**
   1. **Testing Estimate: The testing phase is projected to span approximately two weeks, encompassing a comprehensive review of project requirements, scenario identification, test case design, execution, and iterative testing.**
   2. **Completion Deadline: Anticipated completion of the testing phase is expected by the conclusion of the third week of the initiation of the testing process. This timeline includes factoring in all necessary activities such as defect management, collaboration with the development team, finalizing test execution, and reporting to stakeholders.**

**This condensed testing timeline is geared towards achieving efficiency without compromising the thoroughness and accuracy of the testing procedures.**

1. **Control Procedures**
   1. 6.1 Reviews
      1. Regular evaluations will be conducted to gauge the advancement of testing, ensuring alignment with project requirements and objectives. These assessments will involve key stakeholders, including team members, project managers, and relevant parties. The primary purpose of these reviews is to offer constructive feedback, pinpoint gaps or issues in the testing process, and implement necessary adjustments to enhance the overall quality of the delivery management system.
   2. 6.2 Bug Review Meetings
      1. Scheduled gatherings will be held regularly to discuss and prioritize identified issues or defects. In these sessions, the testing and development teams will collaborate to assess reported bugs, determine their severity and impact, and allocate responsibilities for resolution. Bug review meetings play a pivotal role in fostering clear communication, monitoring bug fix progress, and ensuring the timely resolution of identified issues.
   3. 6.3 Change Request
      1. During the testing phase, possible changes to the system might become apparent and call for improvements or modifications. These requests for changes may result from project requirements changes, improvements that have been noticed, or input from stakeholders. Documenting suggested changes, assessing their effects, obtaining consent from pertinent parties, and putting the approved changes into practice are all steps in the change request process. This methodical approach minimizes delays to the project schedule by making sure that any necessary changes are properly documented, reviewed, and implemented.
   4. 6.4 Defect Reporting
      1. Integral to the testing process is defect reporting. Testers will meticulously document identified defects or issues using a standardized format. The report will encompass details such as defect description, steps for reproduction, expected and actual behavior, and any supporting attachments. Categorized by severity (e.g., critical, major, minor), these defect reports will be prioritized for resolution. The defect reporting process facilitates effective tracking, communication, and resolution of identified issues, contributing to the overall refinement of the delivery management system.
2. **Functions To Be Tested**
   1. **User input data format verification.**
   2. Error message handling.
   3. Destination verification.
   4. Box size verification.
   5. Maximum weight verification.
   6. Truck assignment based on shortest routes algorithm.
   7. Assign another truck if there are no available spaces for the truck with the best routes.
   8. Assign another truck if there is lower utilization usage of the truck for both trucks have same destination cost.
   9. Event handling if all trucks are full loading or cannot have spaces for the shipment.
   10. Trucks weight / volume updating procedure handling.
   11. Exit Program handling.
   12. Program reliability Test.
3. **Resources and Responsibilities**  
   8.1. Resources
   1. Testing Team:
      1. Competent testers dedicated to executing test cases, recording results, and notifying any identified issues or defects.
   2. Testing Environment:
      1. A suitable setup equipped with necessary hardware, software, and simulated data to facilitate testing.

8.2. Responsibilities

a. Testers:

i. Tasked with executing test cases, recording results, and reporting any identified issues or defects during testing.

b. Developers:

i. Accountable for addressing reported issues, implementing required code changes, and retesting fixes.

c. Project Managers:

i. Supervise testing activities, ensure effective coordination between testing and development teams, and provide essential support and resources. Coming up with test plans and strategies to enhance the overall workflow.

1. **Deliverables**

**i. Test Plan: Prepared by the testing team, outlining the testing approach, scope, objectives, and timelines. Responsible: Testing Team.**

**ii. test Cases: Detailed test cases covering various scenarios and edge cases. Responsible: Testing Team.**

**iii. Test Execution: Carried out by the testing team according to the test plan and test cases. Responsible: Testing Team.**

**iv. Defect Reports: Any defects identified during testing will be documented and reported to the development team for resolution. Responsible: Testing Team.**

**v. Test Summary Report: A comprehensive report summarizing the testing efforts, results, and any remaining risks. Responsible: Testing Team.**

1. **Suspension / Exit Criteria**

-Suspension criteria

Suspend a test if the severity level is more than medium.

-Exit criteria

Exit a test if it passes 90% of the scripts and if you can get the expected output even though there are some minor warnings or issues.

1. **Resumption Criteria**

After fixing issues or bugs in the algorithm, the next step is careful retesting to make sure it fits well with the project's goals. This involves testing the algorithm in various situations and with different cases to not only check its basic functions but also see how accurate it is in different scenarios. Retesting is crucial for maintaining the reliability and performance of the algorithm. By testing how it responds in different conditions, we understand how adaptable and strong it is. This is important to confirm that the algorithm meets the initial requirements and can handle different inputs and situations. Also, when changes are made to the algorithm, it's important to systematically retest it. This step goes beyond just checking if the changes work, it ensures that the modifications don't unintentionally introduce new problems or disrupt how the algorithm originally worked. The main goal is to make sure any adjustments fit well with the project's goals without hurting the overall performance of the algorithm.

1. **Dependencies**  
   12.1 Personnel Dependencies
   1. Identify the project team members and outline their respective roles and responsibilities. Highlight any dependencies linked to the availability or skills of specific team members, emphasizing the need for seamless collaboration.

12.2 Software Dependencies

b. To execute testing successfully, specific software components are essential. These encompass the application under scrutiny, tools for test management, defect tracking systems, and any automation tools required. Ensuring the stability and accessibility of the software being tested is crucial, as it must function correctly with all necessary features. Compatibility between testing tools and the software guarantees accurate results and a seamless testing process.  
 12.3 Hardware Dependencies

c. Clearly state any hardware dependencies essential for the project. Enumerate the necessary hardware components or devices that should be available or interconnected for successful project execution.  
 12.4 Test Data & Database

d. For testing related to data, having access to a suitable test database or environment is paramount. This enables the execution of tests focusing on data manipulation, storage, retrieval, and other database-related functionalities. To facilitate effective testing, ensure access to valid and representative test data that spans various scenarios, encompassing different package weights, sizes, destinations, and accounting for potential edge cases.

1. **Risks**  
   13.1. Schedule
   1. Unexpected events that cause delays in project timelines, like unanticipated technical problems or changes in the project's scope, can endanger the schedule. The answer is to revisit and update the project schedule on a regular basis while leaving room for flexibility. Promptly communicate any changes and put plans in place to minimize any delays.

13.2. Technical

b. technical difficulties, such as integration issues or software bugs, could appear and hinder the project's development, possibly leading to delays. To quickly address technical issues, the solution is to prioritize thorough testing, set up a strong debugging procedure, and guarantee constant communication between development teams.

13.3. Management

c. The smooth execution of the project can be harmed by ineffective project management, which includes poor communication and decision-making. It can also cause confusion and misunderstandings. Enhancing one's ability to communicate clearly, outlining decision-making procedures, and routinely evaluating the efficacy of project management are the answers.

13.4. Personnel

d. Project team-related problems, including important members quitting or a lack of qualified workers, could jeopardize the project's viability and continuation. Assisting fellow team members, keeping a knowledgeable and diverse group, and creating backup plans in case of unplanned absences or shortages are the answers.

13.5 Requirements

e. Project requirements that are altered or misunderstood may result in rework, delays, and even conflicts, all of which could negatively affect the project's overall success. The answer is to be meticulous and pay great attention when collecting requirements so that you don't miss anything crucial in the deluge of project data. Establish a strong change management system as well to ensure that requirement alterations are handled precisely and clearly.

1. **Tools**
   1. **Jira:** i. Project management and issue tracking. Facilitates collaboration, task tracking, and progress monitoring.
   2. **Git (Version Control)**
      1. Tracking changes in source code. Ensures version control, collaboration, and seamless code integration.
   3. Unit Test Frameworks:
      1. Automated testing of individual code units. Enhances code reliability, identifies bugs early, and supports continuous integration.
   4. Visual Studio:
      1. Integrated development environment (IDE) for coding, testing, and debugging. Streamlines development processes offers debugging tools and supports multiple programming languages.
2. **Documentation**

In the testing phase, several documents like test cases, scripts, reports, defect logs, and user guides are generated. These documents serve distinct purposes, offering vital information. The format and arrangement of these documents can differ, and there might be specific templates or guidelines to adhere to. The team must comprehend and adhere to these templates and guidelines. This ensures effective documentation of the testing process and clear communication of the outcomes.

1. **Approvals**

**Before commencing any testing activities, ensure that all relevant parties, especially the clients, have provided their consent and that the process and timeline for obtaining it are clearly outlined. This methodical process ensures that decision-makers will have a thorough assessment, which builds trust in the testing procedure and promotes mutual understanding between all involved parties. It highlights how crucial a well-organized and authorized test plan is to the success of testing projects.**