Test Plan

1. **Introduction:**
   1. Test Plan Objectives
2. **The goal of this test plan is to make sure the delivery algorithm correctly assigns packages to trucks based on their capacity, closeness to the destination, and available route.**
3. **Scope:**

**We will be testing about the following cases:**

* + 1. Putting packages in trucks based on how much they can hold.
    2. Finding the truck that's closest to the destination.
    3. Figuring out the quickest way from where the truck is to the destination.
    4. Dealing with times when trucks can't get to where they need to go.

1. **Test Strategy (JC, CJ)**

The testing plan is to make sure the system works well and meets expectations. We use different tests to find big problems, to check important functions, and documentation tests to make sure everything matches the software. User acceptance tests and system tests check if users are happy and if the whole system works right. Some tests like security or handling lots of data might not be needed, depending on the project. Our goal is to create a system that works well, meets needs, and gives users a positive experience. Some of the test strategy we are going to include are as follow:

3.1. System Test:

System testing checks that the whole system works correctly together as one unit. It tests how all the parts of the system work together to make sure they meet the requirements. During system testing, important tests like Performance Testing and Integration Testing help ensure the system works well in different situations and that all the parts fit together smoothly. As we move closer to completing the project, the amount of testing required will increase. This means more test cases will need to be executed, which will take additional time.

3.2. User Acceptance Testing:

Tester verifies that the program meets the expectations and needs of the end users. It includes testing real-world scenarios to ensure the program correctly assigns packages to trucks, calculates distances accurately, and delivers packages to the right destinations.

3.3. Documentation Test:

Documentation testing involves checking that the documentation matches the actual software, offers clear user instructions, and accurately describes the system's features, functions, and behavior.

3.4 Risk-based Test:

Risk-based testing means testers prioritize their testing based on identified risks. They focus more on critical areas that are likely to have problems or big impacts.

C programs are generally expected to compile and run consistently across various operating systems and hardware architectures without significant compatibility issues. We are developing exclusively in C, **compatibility testing** may not be necessary because the language itself ensures broad compatibility across different environments. **Recovery testing** might not be necessary for the project because there are no specific failure scenarios or recovery mechanisms mentioned in the project description. Likewise**, stress and volume testing** may not be relevant for the project because there are no explicit requirements to test the system's scalability or performance under extreme load conditions. If the system isn't expected to manage large volumes of packages or trucks, conducting these tests may not be necessary.

**Test Design and Overview Process:**

1. Understanding requirements: Testers carefully study project requirements, functional details, and relevant documents to grasp what needs testing. They identify main features, functions, and expected behaviors.

2. Creating a traceability matrix: A traceability matrix links requirements to test cases. Testers connect each requirement to one or more test cases to ensure all aspects are covered. This matrix helps track test coverage and ensures all requirements are tested.

3. Preparing test cases: Test cases are developed based on identified requirements and associated scenarios. Testers define input data, expected results, and steps needed for each test case. They make sure test cases cover positive and negative scenarios, limits, and unusual situations.

4. Reviewing: Another team member reviews test cases to improve their quality and effectiveness. The reviewer gives feedback on clarity, correctness, and coverage of the test cases. This process finds any gaps, errors, or ways to enhance the test cases.

1. **Environment Requirements (JC, CJ)**

**The testing process does not require high-end hardware specifications because testing will be conducted on developers' workstations, which include both Mac laptops and Windows machines. We have decided to work with the C compiler and Visual Studio Code as the preferred code editor.**

1. **Execution Strategy (JC, CJ)**
   1. **Entry Criteria:** The software builds to be tested should be available and deployed on the test machines.
   2. **Exit Criteria:** The tests can be considered completed when 100% of the test scripts pass, with no severe or critical defects remaining.
   3. **Severity levels:**
      1. Critical: Defects that cause the system to crash or produce anomalous results.
      2. High: Defects that cause a lack of program functionality, but there might be a workaround available.
      3. Medium: Defects that degrade the quality of the system but have a workaround to achieve the desired functionality.
      4. Low: Minor errors with minimal impact on functionality, such as unclear error messages
      5. Cosmetic: Issues that make the user interface less optimal but do not affect functionality.
   4. **Test Reporting:**
2. Reports: Test reports will track testing progress and results.
3. Frequency: Daily reports will summarize tests conducted, passed, and failed. They will include descriptions of tested areas and any issues encountered.
4. Recipients: Reports will go to the project manager, development team, and quality assurance team.
5. Communication: Testers will report bugs to project managers, who will assign developers to fix them. Regular meetings will facilitate collaboration between QA and development teams.
6. **Test Schedule (KL, KO)**

**The estimated duration for testing is approximately a few weeks. We anticipate completing all testing activities by the beginning of the 4th week from the start of the testing process.**

1. **Control Procedures (KL, KO)**
   1. 7.1. Reviews

Regular reviews will be held to evaluate testing progress and ensure alignment with project requirements and goals. Their purpose is to give feedback, of any testing process issues or gaps, and enhance the delivery management system's quality.

7.2. Bug Review Meetings

Regular meetings will be scheduled to address and prioritize high issues, the testing and development teams will work together to review reported bugs, assess their severity and impact, and assign tasks for resolution.

7.3. Change Request

Change requests might come up during testing if we need to modify or enhance the system. Stakeholder feedback, identified improvements, or shifts in project requirements could drive these requests. They follow a formal process involving documenting the changes, assessing their impact, getting approval from stakeholders, and then making the approved changes. This process ensures that modifications are properly documented, reviewed, and implemented without disrupting the project timeline.

7.4. Defect Reporting

Defect reporting is crucial in testing. Testers will document any found issues using a standard format. This report includes details like the defect's description, steps to replicate it, expected versus actual behavior, and supporting files. Defects are categorized by severity (like critical, major, minor) and prioritized for fixing. This process helps track, communicate, and resolve issues efficiently.

1. **Functions To Be Tested (KL, KO)**

The following function will be tested:

8.1. Allocation Shipment Function:

This function will be tested to make sure shipments are correctly assigned to trucks based on weight, box size, and destination. Test cases will include different scenarios, such as valid and invalid inputs, availability of multiple trucks, and constraints.

8.2. Capacity Calculation Function: This function will be tested to confirm its accuracy in calculating the available capacity of each truck. Test cases will include scenarios with various combinations of weight and box sizes, reaching maximum weight or volume limits, and handling trucks with different constraints.

8.3. Shortest Path Calculation Function: We'll test this function to ensure it accurately finds the shortest route between two points, while avoiding buildings. Test cases will cover various scenarios, including different start and end points, obstacles like buildings, and edge cases.

8.4. Message Function: We'll test this function to ensure it generates accurate and clear messages about truck selection, diversion paths, and other important details. Test cases will include various allocation scenarios and edge cases to verify the correctness and clarity of the output messages.

1. **Resources and Responsibilities (KL, KO)**  
   9.1. Resources:

**We will need** a team dedicated to running test cases, documenting results, and reporting issues or defects. A proper test environment is required which includes the hardware, software, and simulated data needed to conduct the tests effectively.

9.2. Responsibilities:

Testers will execute test cases, document results, and report any issues or defects found. Developers will be responsible for fixing reported issues, making necessary code changes, and retesting the fixes. Project Managers will oversee testing activities, coordinate between testing and development teams, and provide support and resources for testing.

1. **Deliverables (KL, KO)**

The algorithm needs to:

1. Assign packages to trucks based on available space, distance to the destination, and any diversion required.
2. Calculate the shortest path from the nearest point on the truck's route to the destination.
3. Handle situations where a truck cannot reach the destination due to obstacles.
4. Print information about the assigned truck, delivery destination, and any diversion path if necessary.
5. **Suspension / Exit Criteria (RR)**
6. If the algorithm can't assign packages to trucks based on available space, distance, and other requirements.
7. If the algorithm makes errors while calculating the shortest path.
8. Stop the algorithm if the baggage weight exceeds 1000 kg.
9. If the algorithm can't manage situations where a truck can't reach the destination due to obstacles.
10. If the algorithm produces unnecessary output.
11. **Resumption Criteria (RR)**

After fixing any identified issues or bugs, the algorithm should undergo retesting to ensure it meets the requirements. This involves testing the algorithm with various scenarios and test cases to verify its functionality and accuracy. Additionally, after making modifications, the algorithm should be retested to ensure these changes haven't introduced new issues.

1. **Dependencies (RR)**  
   13.1 Personnel Dependencies

List the people working on the project and what they do and explain if the project needs certain people to be available or to have certain skills.

13.2 Software Dependencies

For successful testing, we require certain software components. These include the application under test, tools for managing tests, systems for tracking and managing defects, and any needed automation tools. The application being tested must be stable and accessible, meaning it should work correctly and have all the required features. It's also important that our testing tools are compatible with the software being tested. This ensures they can work smoothly together and deliver accurate results.

13.3 Hardware Dependencies

There are no specific hardware dependencies required for this testing project. No hardware components or devices need to be available or connected for the testing process.

13.3 Test Data & Database

Accessing a suitable test database or environment is crucial for testing that involves working with data. It lets us run tests that check how well the system handles tasks like storing, finding, and using data in databases. Good testing needs the right kind of data that's real and shows what could really happen. This data should include lots of different situations, like different package sizes and where they go, even the tricky ones.

1. **Risks (RR)**  
   14.1. Schedule

Schedule risks in testing are about facing difficulties with finishing testing on time. These risks include delays in development, changes in what needs to be tested, not having enough resources, problems getting test data, relying on outside factors, and not having enough time planned for testing. To handle these risks, it's important to plan the project well, talk clearly with everyone involved, decide what to do first, and act in time.

14.2. Technical

Technical risks in testing can cause problems that affect how well the testing works. These risks include issues with networks or computers, trouble setting up test environments, limits with testing tools, and problems handling data.

14.3. Management

Identify potential risks related to project management and coordination.

Assess the impact of poor project management on the project's progress.

14.4. Personnel

This mostly includes individual challenges within the team like the members feeling low motivation, communication problems, skill gaps, and maybe even personal conflicts. To reduce these risks, it's crucial to create a positive team atmosphere, promote open communication, help each other, and resolve conflicts quickly. Regular meetings and personal support can help keep the testing team motivated and solve any issues within the members.

14.5 Requirements

Identify risks due to unclear, incomplete, or changing requirements. Then we plan to evaluate how these issues can affect the project's scope and what gets delivered.

1. **Tools (RR)**

Testing tools are very important for testing well. They include tools for planning and tracking tests, managing software problems, checking how well the system works, and creating test data. Using the right tools can make testing faster and more effective.

1. **Documentation (RR)**

During testing, different documents are made, such as test cases, test scripts, test reports, defect logs, and user guides. These documents have different purposes and provide important information. Their format and structure can vary, and there might be specific templates or guidelines to follow.

1. **Approvals (RR)**

Identify the people or stakeholders who need to approve the test plan and related documents. Describe the steps and timeline for getting their approval, making sure everyone has reviewed and agreed before beginning the testing activities.

(Note: Initials of the team member have been mentioned on the part they worked on.)