Test Plan Template

1. **Introduction -SJL**
    a. Test Plan Objectives
        i. The objective of this test plan is to ensure that the delivery algorithm accurately assigns packages to trucks based on their capacity, proximity to the destination and available route.
2. **Scope - SJL**
    a. Package assignment to trucks based on weight and volume capacity
    b. Determining the closest truck to the destination
    c. Calculating the shortest path from the nearest point on the truck's route to the destination
    d. Handling cases when trucks cannot reach the destination.
3. **Test Strategy - SJL**

Our test strategy focuses on ensuring the quality and functionality of the system. It includes exploratory tests to uncover critical defects and functional tests to validate key application functions. Documentation testing ensures alignment with the software implementation. User acceptance testing and system testing are performed to meet user expectations and verify the overall system functionality. Certain tests like security, stress and volume, and recovery testing may not be required based on project specifications. Additionally, unit testing or assertions are used to validate individual components. Our strategy aims to deliver a reliable solution that meets requirements and provides a high-quality user experience.

Required Exploratory Tests

- Scenario-based testing: Testers create realistic scenarios and simulate user interactions to explore different paths, functionalities, and edge cases.
- Documentation Test: Ensuring that the documentation aligns with the actual software implementation, provides clear and comprehensive instructions for users and accurately reflects the system's features, functionalities and behavior.
- Risk-based testing: Testers prioritize their testing efforts based on identified risks, focusing on critical areas that have a higher likelihood of failure or impact.

Required Functionality Tests

- User Acceptance Test : Validating the program meets the expectations and requirements of the end users. This can involve testing real world scenarios and verifying that the program correctly assigns packages to trucks, calculates distances, and delivers packages to the correct destinations.
- System Test: Verifies the system's overall functionality, ensuring that it functions correctly as a whole. It tests the integrated system to ensure that all components work together and meet the requirements. Few of the tests from the list can be included as System Test, such as PerformanceTesting and Integration Testing.

Required Functionality Testings will be performed with Unit testing or Assertions. Even though system testing aims to ensure integration as a whole, it can help that individual components or modules work correctly and integrate properly.

Tests might NOT required

- Security Test : While security is important, the project description does not indicate any specific security requirements or sensitive data handling. Therefore this test may not be necessary

- Stress and Volume Test : The project does not explicitly mention the need to test the system's scalability or its performance under extreme load conditions. If the system is not expected to handle large volumes of packages or trucks, conducting stress and volume tests may not be relevant.

- Recovery Test : Since the project does not mention any specific failure scenarios or recovery mechanisms, testing the system's recovery capabilities may not be required.

Test Design Process and Conduct overview

Understanding requirements: Testers thoroughly analyze the project requirements, functional specifications, and any relevant documentation to gain a clear understanding of what needs to be tested. They identify key features, functionalities, and expected behavior.

Building a traceability matrix: A traceability matrix establishes a clear link between the requirements and the corresponding test cases. Testers map each requirement to one or more test cases to ensure all aspects are covered. This matrix serves as a reference for tracking the test coverage and ensuring all requirements are tested.

Preparing test cases: Test cases are created based on the identified requirements and their associated test scenarios. Testers define the input data, expected outcomes, and any specific steps or conditions required for each test case. They ensure the test cases are comprehensive, covering positive and negative scenarios, boundary conditions, and edge cases.

Test case review: To enhance the quality and effectiveness of the test cases, they are reviewed by another member of the quality assurance team. The reviewer provides feedback on the clarity, correctness, and coverage of the test cases. This review process helps identify any gaps, inconsistencies, or improvements that need to be addressed.

4. Environment Requirements - SO

a. Hardware:
    i. Mac OS: The test environment requires Mac computers with sufficient hardware specifications to run the software being tested.
    ii. Windows: The test environment requires Windows computers with sufficient hardware specifications to run the software being tested.
b. Software:

i. Mac OS: The testing will be conducted using the latest version of macOS (e.g., macOS Big Sur) with required dependencies and software tools installed.
ii. Windows: The testing will be conducted using the latest version of Windows (e.g., Windows 10) with required dependencies and software tools installed.

c. Test Harness: A pre-existing set of testing tools will be used to conduct the tests. These tools should be installed and properly configured on the test machines.

5. **Execution Strategy - SO**
    a. Entry Criteria: The software build to be tested should be available and deployed on the test machines.
    b. Exit Criteria: The tests can be considered completed when 100% of the test scripts pass, with no severe or critical defects remaining.
    c. severity levels:
        i. **critical** Defects that cause the system to crash or produce anomalous results,
        ii. **high** Defects that cause a lack of program functionality, but there might be a workaround available
        iii. **medium** Defects that degrade the quality of the system but have a workaround to achieve the desired functionality
        iv. **Low** Minor errors with minimal impact on functionality, such as unclear error messages
        v. **Cosmetic** Issues that make the user interface less optimal but do not affect functionality.
    d. **Test Reporting**
        i. Reports: Test reports will be produced to track the progress and results of the testing activities.
        ii. Frequency: Reports will be generated daily, providing an overview of the number of tests conducted, passed, and failed. The reports will include a brief description of the areas being tested and the areas that are failing.
        iii. Recipients: The reports will be sent to the project manager, development team, and quality assurance team.
        iv. Communication: Testers will provide feedback and bug reports to the project managers, who will then assign developers to resolve the defects found in the software. Regular communication channels, such as meetings, emails, and issue tracking systems, will be utilized for collaboration between the quality assurance team and the development team.
6. **Test Schedule - SO**
    a. Testing Estimate: The testing is estimated to take approximately two weeks to complete.
    b. Completion: The testing is expected to be finished by the end of the third week from the start of the testing process.
7. **Control Procedures- shine**

7.1 Reviews: Regular reviews will be conducted to assess the testing progress and ensure that the testing activities align with the project requirements and objectives. The reviews will involve key stakeholders, including team members, project managers, and other relevant parties. The purpose of these reviews is to provide feedback, identify gaps or issues in the

testing process, and make necessary adjustments to improve the overall quality of the delivery management system.

7.2 Bug Review Meetings: Regular meetings will be scheduled to discuss and prioritize any identified issues or defects. During these meetings, the testing and development teams will collaborate to review the reported bugs, determine their severity and impact, and assign responsibilities for resolution. The bug review meetings are crucial for maintaining clear communication, tracking the progress of bug fixes, and ensuring the timely resolution of identified issues.

7.3 Change Request: Change requests may arise during the testing phase if system modifications or enhancements are required. These change requests could be driven by stakeholder feedback, identified improvements, or changes in the project requirements. The change requests will go through a formal process that includes documenting the requested changes, evaluating their impact, seeking approval from relevant stakeholders, and implementing the approved changes. The change request process ensures that any necessary modifications are correctly documented, reviewed, and implemented while minimizing disruption to the project timeline.

7.4 Defect Reporting: Defect reporting is essential to the testing process. Testers will document any identified defects or issues using a standardized defect reporting format. The report will include details such as the description of the defect, steps to reproduce it, expected behavior, actual behavior, and any supporting attachments. The defect reports will be categorized based on severity (e.g., critical, major, minor) and prioritized for resolution. The defect reporting process allows for effective tracking, communication, and resolution of identified issues.

8. **Functions To Be Tested - shine**

The functions that will be tested include:

8.1 Shipment Allocation Function:

This function will 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, multiple truck availability, and capacity considerations.

8.2 Shortest Path Calculation Function:

The shortest path calculation function will be tested to verify that it correctly determines the shortest path between two points while avoiding buildings. Test cases will include scenarios with different starting and destination points, buildings obstructing the way, and corner cases.

8.3 Capacity Calculation Function:

The capacity calculation function will be tested to validate its accuracy in determining 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 Function:
The output message generation function will be tested to ensure that 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.

9. Resources and Responsibilities - shine

9.1. Resources: The following resources will be required for the testing phase:

a. Testers: A team of dedicated testers who will execute the test cases, document the results, and report any issues or defects.

b. Test environment: A suitable testing environment comprising the necessary hardware, software, and simulated data to support the testing activities.

9.2 Responsibilities

a. Testers: The testers will be responsible for executing the test cases, documenting the test results, and reporting any issues or defects discovered during testing.

b. Developers: The development team will address the reported issues and defects, make necessary code changes, and retest the fixes.

c. Project Managers: The project managers will oversee the testing activities, ensure proper coordination between the testing and development teams, and provide necessary support and resources to facilitate effective testing.

By following these control procedures, conducting thorough testing, and assigning clear responsibilities, the project team can ensure the delivery management system's quality, reliability, and compliance with the defined requirements.

10. Deliverables - Ji Ho Nam

The algorithm should correctly assign packages to trucks based on available space, distance to destination, and diversion required.
The algorithm should calculate the shortest path from the nearest point on the truck's route to the destination.
The algorithm should handle cases where a truck cannot reach the destination due to obstacles.
The algorithm should print the assigned truck, delivery destination, and any diversion path if required.

11. Suspension / Exit Criteria - Ji Ho Nam

1. If the algorithm fails to assign packages to trucks based on available space, distance, and diversion requirements.
2. If the algorithm fails to calculate the shortest path correctly or encounters errors in pathfinding.
3. If the algorithm cannot handle cases where a truck cannot reach the destination due to obstacles.

4. If the algorithm does not print the necessary information accurately.

5. Shut down the algorithm if the baggage weight exceeds 1000 kg

12. Resumption Criteria - Ji Ho Nam

Once any identified issues or bugs are fixed, the algorithm should be retested to ensure the deliverables are met.
The algorithm should be retested using different scenarios and test cases to validate its functionality and accuracy.
The algorithm should be retested after modifications to confirm that changes do not introduce new issues.

13. Dependencies - Yoojin Lee

13.1 Personnel Dependencies:
List the personnel involved in the project and their roles and responsibilities.
Specify any dependencies on the availability or skills of specific team members.

13.2 Software Dependencies:
To conduct the testing successfully, we need specific software components. These include the application we are testing, tools for managing tests, systems to track and manage defects, and any automation tools required. The software we are testing need to be stable and accessible. This means that it should be working properly and have all the necessary features. We also need to ensure that the testing tools we are using are compatible with the software being tested. This ensures that the tools can work well with the software and provide accurate results.

13.3 Hardware Dependencies:
Specify any hardware dependencies required for the project.
List the hardware components or devices that need to be available or connected.

13.4 Test Data & Database:
Having access to a suitable test database or environment is important for data-related testing. This allows for the execution of tests that specifically focus on data manipulation, storage, retrieval, and any other database-related functionalities. For effective testing, it is essential to have access to valid and representative test data. This data should cover a wide range of scenarios, including different package weights, sizes, destinations, and consider edge cases.

14. Risks - Yoojin Lee

14.1 Schedule:
Schedule risks in testing refer to challenges related to meeting testing deadlines. Risks include development delays, scope changes, resource constraints, test data availability, dependencies on external factors, and inadequate time allocation. To mitigate these risks, it

is important to have realistic project planning, clear communication with stakeholders, effective prioritization, and proactive risk management.

14.2 Technical:
Technical risks in testing can impact the quality and effectiveness of the process. Risks include infrastructure issues like network problems and hardware limitations, challenges in setting up test environments, tool limitations, and data management concerns. It is important to address these risks by ensuring compatibility, configuring environments properly, and managing data effectively. Collaborating with a professor and staying updated on tools and security measures can help mitigate these risks and improve testing efficiency.

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:
Personal risks in testing involve challenges at an individual level within the team, such as turnover, lack of motivation, communication issues, skill gaps, and personal conflicts. To mitigate these risks, it is important to foster a positive team environment, encourage open communication, provide support, and address conflicts promptly. Regular meetings and individual support help maintain a cohesive and motivated testing team, promoting personal growth and satisfaction.

14.5 Requirements:
Identify potential risks related to unclear, incomplete, or changing requirements.
Assess the impact of requirements issues on the project's scope and deliverables.

15. Tools - Yoojin Lee

Testing tools are essential for efficient and accurate testing. They include test management tools for planning and tracking tests, defect tracking tools for managing software issues, performance testing tools for assessing system performance, and test data management tools for creating test datasets. Choosing the right tools can enhance testing efficiency and productivity.

16. Documentation - Yoojin Lee

During testing, various documents are created, such as test cases, test scripts, test reports, defect logs, and user guides. These documents have different purposes and provide important information. The format and structure of these documents may vary, and specific templates or guidelines may be provided to follow. It is important for international students to understand and follow these templates and guidelines to document the testing process effectively and communicate the outcomes clearly.

17. Approvals - Yoojin Lee

Identify the individuals or stakeholders responsible for approving the test plan and associated documents. Define the process and timeline for obtaining their approval, ensuring that all necessary parties have reviewed and given consent before starting the testing activities.