Test Plan Template

1. **Introduction** 
   1. Test Plan Objectives

* + 1. The objective of this test plan is to ensure that the delivery algorithm effectively assigns packages to trucks based on their capacity, distance to the destination and the available routes.

1. **Scope** 
   1. Assign packages to trucks based on weight and volume capacity
   2. Identify the closest truck to the destination provided
   3. Calculate the shortest route from the nearest point on the truck’s path to the destination
   4. Handle cases where trucks cannot reach their destination.
2. **Test Strategy**

Our test strategy is designed to ensure the system’s functionality and quality. Our goal is to deliver a reliable solution that meets requirements and ensures a high-quality user experience. It will include exploratory testing to identify any critical defects and functional testing to verify key application functions. Documentation testing will ensure consistency with the software implementation. User acceptance testing and system testing are conducted to meet the user’s expectations and confirm the overall system functionality. Several tests such as security, stress and volume, and recovery testing may not be necessary based on project specifications. Furthermore, unit testing or assertions are used to validate individual components.

**Required Exploratory Tests**

* Scenario-based Testing: Testers will develop realistic scenarios and simulate user interactions to explore different paths, functionalities, and edge cases.
* Documentation Testing: Ensures that the documentation matches the actual software implementation, provides clear and comprehensive instructions for users and accurately represents the system’s features, functionalities and behavior.
* Risk-based Testing: Testers prioritize testing efforts based on identified risks, focusing on critical areas with a higher probability of failure or considerable impact.

**Required Functionality Tests**

* User Acceptance Testing: Confirming that the program meets the end users' expectations and requirements. This involves testing real world scenarios and verifying that the program accurately assigns packages to trucks, calculating distances, and delivering packages to the correct destinations.
* System Test: Verifies overall functionality of the system, ensuring that it operates accordingly as a whole. Tests the integrated system to ensure all components work together and meet requirements. This can include tests such as performance testing and integration testing.

The above required functionality testing will be performed using Unit testing or Assertions. Even though system testing aims to ensure integration, it can also help with individual components or modules work correctly and integrate properly.

**Tests that might not be required**

* Stress and Volume Testing: The project does not specify the need to test the system’s scalability or performance under extreme load conditions. If the system is not expected to handle large volumes of packages or trucks, stress and volume testing may be unnecessary.
* Recovery Testing: As the project does not state any specific failure scenarios or recovery mechanisms, testing the system’s recovery capabilities may not be needed.
* Security Test: Although security is important to consider, the project description does not specify any specific security requirements or sensitive data handling. Thus, this test may not be required.

**Test Design Process and Conduct Overview**

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

Building a traceability matrix: A traceability matrix is made to establish a clear link between the requirements and 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 test coverage and ensuring all requirements are tested.

Test cases preparation: Test cases are developed based on the identified requirements and their associated test scenarios. Testers will define input data, expected outcomes, and any specific steps or conditions needed 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 further improve the quality and effectiveness of the test cases, another quality assurance team member reviews them. The reviewer will provide feedback on the clarity, correctness, and coverage of the test cases, helping identify any gaps, inconsistencies, or improvements.

1. **Environment Requirements** 
   1. Hardware:
      1. Mac OS: The test environment requires Mac computers with sufficient hardware specifications to run the software being tested.
      2. Windows: The test environment requires Windows computers with sufficient hardware specifications to run the software being tested.
   2. Software:
      1. 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.
      2. Windows: The testing will be conducted using the latest version of Windows (e.g., Windows 10) with the required dependencies and software tools installed.
   3. Test Harness: A pre-existing set of testing tools will be used to conduct the tests. These tools must be installed and properly configured on the test machines.
2. **Execution Strategy** 
   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, without severe or critical defects remaining.
   3. Severity levels:
      1. **Critical**: Defects causing 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 system quality 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**
      1. Reports: Test reports will be produced to track the progress and results of the testing activities.
      2. 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.
      3. Recipients: Testers will provide feedback and bug reports to the project managers, who will then assign developers to fix any issues found in the software. The quality assurance and development team will collaborate using regular communication methods, such as meetings, emails, and issue tracking systems.
      4. Communication: Testers will provide feedback and bug reports to the project managers, who will then assign developers to fix any issues found in the software. The quality assurance and development team will collaborate using regular communication methods, such as meetings, emails, and issue tracking systems.
3. **Test Schedule** 
   1. Testing Estimate: The testing is estimated to take about one week to complete.
   2. Completion: The testing is expected to be finished by the end of one week after the start of the testing process.
4. **Control Procedures**

**7.1** Reviews: Regular reviews will be conducted to assess the testing progress and ensure that it meets the project's goals and requirements. The reviews will involve team members, project managers, and other relevant parties. The purpose of these reviews is to find any problems in the testing process and improve the delivery management system's quality.

**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 client 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 clients, 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.

1. **Functions To Be Tested**

The functions that will be tested include:

**8.1** Shipment Allocation Function:

This function will be tested to ensure that shipments are correctly allocated to trucks based on weight, box size, and destination. Test cases will include various scenarios, such as 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 accurately determines the shortest path between two points while avoiding buildings. Test cases will include scenarios with different starting and destination points, obstacles such as buildings, and edge cases.

**8.3** Capacity Calculation Function:

The capacity calculation function will be tested to ensure it accurately determines each truck's available capacity. Test cases will include 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 important information. Test cases will include different allocation scenarios and edge cases to validate the correctness and clarity of the output messages.

1. **Resources and Responsibilities**

**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.

1. **Deliverables**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.

1. **Suspension / Exit Criteria**

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

2. If the algorithm fails to correctly calculate the shortest path 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 accurately print the necessary information.

5. Shut down the algorithm if the baggage weight exceeds 2500 kg

1. **Resumption Criteria**

Once any identified issues or bugs are fixed, the algorithm should be retested to ensure the deliverables are met.

The algorithm should be retested with different scenarios and test cases to validate its functionality and accuracy.

The algorithm should be retested after any modifications to confirm that changes did not introduce new issues.

1. **Dependencies**   
   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:

For testing to be successfully conducted, specific software components are required. These include the application we are testing, tools for managing tests, defect tracking systems, and any needed automation tools. The software we are testing needs to be stable and accessible, with the necessary features to function properly.

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 will specifically focus on data manipulation, storage, retrieval, and any other database-related functionalities Valid and representative test data is important for effective testing. This covers a wide range of scenarios, including different package weights, sizes, destinations, and edge cases.

1. **Risks**   
   14.1 Schedule:

Schedule risks in testing involve challenges related to meeting testing deadlines. Risks may include test data availability, scope changes, development delays, limited resources, dependencies on external factors, and ineffective time allocation. To reduce the likelihood of such risks, it requires realistic project planning, clear communication with clients, effective prioritization, and proactive risk management.

14.2 Technical:

Technical risks in testing can affect the process's quality and effectiveness. These risks include infrastructure issues like network problems and hardware limitations, challenges in setting up test environments, tool limitations, and data management concerns. Addressing these risks involves ensuring compatibility, properly configuring environments, and effectively managing data. Collaboration with a professor and staying updated on tools and security measures can help mitigate these risks and enhance testing efficiency.

14.3 Personnel:

Personal risks in testing involve challenges affecting individuals within the team, such as turnover, lack of motivation, communication issues, skill gaps, or even personal conflicts. To limit these risks, it is required to adopt a positive team environment, encourage open communication, provide support, and promptly address conflicts. Regular meetings and individual support will help maintain an organized and motivated testing team, promoting personal growth and satisfaction.

14.4 Management:

Identify potential risks related to project management and coordination.

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

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.

1. **Tools**

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.

1. **Documentation**

During testing, different documents are created. These may include test reports, test cases, test scripts, defect logs, and user guides. These documents have different purposes, and all provide crucial information. The format and structure may differ, and specific templates or guidelines may be provided. 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.

1. **Approvals**

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