Test Plan Template

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

**This Testing is to just outline all the things this project can do and how we are going to use the testing method on delivery company mapping software. The software for this project takes care of all these packages by allocating them to 3 delivery trucks which operate in a 25x25 grid city map. These test plans describe the following features, Right input validation for package details; Right assignment of packages to corresponding trucks, Calculation of accurate shortest delivery routes, Truck capacity handled correctly.**

1. **Scope**
   1. **In this scope, we will test:**

* Package assignment to trucks based on weight and volume capacity
* Determining the closest truck to the destination
* Calculating the shortest path from the nearest point on the truck’s route to the destination
* Handling cases when trucks cannot reach the destination.

1. **Test Strategy**

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 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 Performance Testing and Integration Testing.

1. **Environmental Requirements**

** Hardware:** Windows-based PCs, C++ compiler (e.g., g++).

** Software**: Visual Studio, GitHub for version control, and Jira for issue tracking.

 **Test Data:** Simulated shipment data covering various weights, sizes, and destination locations within the 25x25 grid map.

1. **Execution Strategy**
   1. **Entry Criteria**: All components needed for testing (software build, test environment) are available.

**Exit Criteria**: Completion of 100% of test scripts with no critical or high-severity defects remaining.

* 1. 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

**v) Cosmetic** Issues that make the user interface less optimal but do not affect functionality.

1. **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:** The reports will be sent to the project manager, development team, and quality assurance team.
     4. **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.
2. **Test Schedule**

Once Development is done, the testing phase will begin which approximately takes up to a few days or a week, The testing is done by the testers and thus developers will code as to allow the tester to test software at the scheduled pace.

1. **Control Procedures**

Reviews - code review before merging into main branch  
Bug Review Meetings: Identify any new issues in a weekly meeting  
Change Request – GitHub pull request for updating code  
Defect Reporting – Use Jira for tracking

1. **Functions To Be Tested**

Package input validation

Truck assigning method

Shortest path calculation

Capacity management

Calculation of diversion routing

Output formatting

1. **Resources and Responsibilities**  
   8.1. Resources: 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.

8.2. **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.

1. **Deliverables:**
   1. Test cases document
   2. Execution reports
   3. Final Report
2. **Suspension / Exit Criteria:**

Just like this as it is a critical defect which will not allow us to do further testing

Unable to prepare all required resources

1. **Resumption Criteria:** Once any identified issues or bugs are fixed, the algorithm should be retested to ensure the deliveries 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.

1. **Dependencies**  
   12.1 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.

**12.2 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 needs 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.

**12.3 Hardware Dependencies:**

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

**12.4 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.

1. **Risks**  
   **13.1. 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.

**13.2. Technical:**

Technical risks in testing can impact on 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.

**13.3. Management:**

Identify potential risks related to project management and coordination. Assess the impact of poor project management on the project's progress.

**13.4. 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.

**13.5 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**

**1. C++ compiler**

**2. IDE**

**3. Text editors**

**4. Desktop**

**5. Git and GitHub**

**6. Jira**

**7. Unit testing framework**

1. **Documentation**
2. **Reports**
3. **Test plan & Strategy**
4. **Logs**
5. **Comments**
6. **Final Output**
7. **Approvals: 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.**