Test Plan Template

1. **Introduction**

The primary objective of this comprehensive test plan is to assess and validate the delivery algorithm's ability to accurately assign packages to trucks. This assignment of packaging to trucks is based on various parameters, including truck capacity considerations, weight and size of the shipment, proximity to the destination, and the optimization of available routes.

1. **Scope**

a. Package Assignment:

The system will be rigorously tested to ensure precise package assignment to trucks, factoring in weight and volume capacity.

b. Truck Selection:

Thorough examination of the algorithm's capability to determine the closest truck to the destination.

c. Pathfinding:

Validation of the algorithm's efficiency in calculating the shortest path from the nearest point on the truck's route to the delivery destination.

d. Exception Handling:

Examination of the system's response in scenarios where trucks encounter obstacles preventing them from reaching the destination.

1. **Test Strategy**

Our testing strategy encompasses a multi-faceted approach. It includes exploratory tests to unearth critical defects, functional tests to validate key application functions, and documentation testing to ensure alignment with the software implementation. User acceptance testing and system testing are integral to meet user expectations and verify overall system functionality. Specific tests such as security, performance, stress & volume, and recovery testing may be excluded based on project specifications. Additionally, unit testing and assertions are deployed to validate individual components, contributing to the overarching goal of delivering a reliable solution with high-quality user experience.

* + 1. Required Tests

|  |  |
| --- | --- |
| **Test type** | **Description** |
| Scenario-based Test | Testers will create realistic scenarios simulating user interactions to explore different paths, functionalities, and edge cases. |
| Automated Test | Automatic tests are performed by software (Visual Studio) validating the functionality of software and ensuring it meets requirements. |
| Documentation Test | Ensuring that the documentation accurately reflects the software implementation, providing clear and comprehensive instructions for users. |
| Risk-based Testing | Prioritizing testing efforts based on identified risks, with a focus on critical areas. |
| User Acceptance Test | Validation of the program meeting end-user expectations and requirements, including scenarios of accurate package assignment, distance calculation, and correct deliveries. |
| System Test | Verification of the system's overall functionality, ensuring seamless integration and meeting specified requirements. This may include performance and integration testing. |

* + 1. Tests NOT required:

|  |  |
| --- | --- |
| Security Test | Omitted if project specifications do not indicate specific security requirements or sensitive data handling. |
| Stress and Volume Test | Unnecessary if the system is not expected to handle large volumes of packages or trucks. |
| Recovery Test | Excluded if the project lacks specific failure scenarios or recovery mechanisms |

1. **Environment Requirements**

a. Hardware:

i. Mac OS:

The test environment necessitates Mac computers with sufficient hardware specifications.

ii. Windows:

The test environment requires Windows computers with specified hardware capabilities.

b. Software:

i. Mac OS:

Testing conducted on the latest version of macOS (e.g., macOS Big Sur) with all required dependencies and software tools (VS code) installed.

ii. Windows:

Testing conducted on the latest version of Windows (e.g., Windows 11) with necessary dependencies and software tools installed (Visual Studio).

c. Test Harness:

Utilization of a pre-existing set of testing tools, requiring installation and proper configuration on the test machines.

1. **Execution Strategy**

a. Entry Criteria:

The program build to be tested should be available and deployed on the designated test machines.

b. Exit Criteria:

Test completion is determined when 95% of the test scripts pass, with no severe or critical defects remaining.

c. Severity Levels:

i. Critical:

Defects causing system crashes or anomalous results.

ii. High:

Defects causing a lack of program functionality, with potential workarounds available.

iii. Medium:

Defects degrading system quality but with workarounds to achieve desired functionality.

iv. Low:

Minor errors with minimal impact on functionality, such as unclear error messages.

v. Cosmetic:

Issues affecting the user interface aesthetics but not impacting functionality.

d. Test Reporting:

i. Reports will be generated depending on the progress of the software developing cycle, offering an overview of conducted tests, passed and failed scenarios, along with brief descriptions of the areas tested and those experiencing issues.

ii. Reports will be disseminated to the project leader, development team, and quality assurance team.

iii. Communication channels such as meetings, emails, communication applications (MS teams, WhatsApp) and issue tracking systems will facilitate collaboration between the quality assurance team and the development team.

1. **Test Schedule**
2. Testing Estimate: The testing is estimated to take around two weeks to complete.
3. Completion: The testing is expected to be completed by the end of the third week from the inception of the testing process.
4. **Control Procedures**

**7.1** Reviews

Testing progress will be evaluated on a regular basis to make sure that the activities are in line with the project's requirements and criteria. Key stakeholders, such as team members, project managers, and other pertinent parties, will be involved in the reviews. These reviews serve as a means of gathering input, pointing out holes or problems in the testing procedure, and recommending necessary changes to improve the delivery management system's overall quality.

**7.2** Bug Review Meetings

A regular schedule of meetings will be established to prioritize and address any found flaws or issues. The testing and development teams will work together to examine the reported issues, assess their effect and severity, and assign resolution responsibilities during these sessions. The bug review sessions are essential for keeping lines of communication open, monitoring the status of bug fixes, and guaranteeing that issues are fixed quickly.

**7.3** Change Request

If any changes or improvements to the system are needed, change requests could come up during the testing phase. Stakeholder input, recognized enhancements, or modifications to the project specifications may be the driving forces for these change requests. The change requests will be subject to a formal procedure that entails recording the changes, assessing their implications, obtaining consent from pertinent parties, and putting the authorized changes into effect. The change request procedure minimizes delays to the project schedule by making sure that any required adjustments are accurately recorded, examined, and put into effect.

**7.4** Defect Reporting

Reporting defects is a crucial part of the testing procedure. Using a standard defect reporting format, testers will record any flaws or issues found. Details like the defect's description, how to replicate it, predicted and actual behavior, and any supporting attachments will all be included in the report. The defect reports will be prioritized for resolution and classified according to severity (e.g., critical, major, minor). Defect reporting facilitates efficient tracking, communication, and issue resolution of detected problems.

1. **Functions To Be Tested**

**8.1** Shipment Allocation Function:

This function will be tested to make sure that shipments are allocated correctly to trucks based on a combination of factors including weight, box size, and destination. Various scenarios will be covered in the test cases, considering valid and invalid inputs, the availability of multiple trucks, and capacity.

**8.2** Shortest Path Calculation Function:

The shortest path calculation function will be tested to ensure that it correctly identifies the shortest path between two points while avoiding buildings. Test cases will be created based on 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 each truck’s available capacity. Test cases will cover scenarios for various weight and box size combinations, including the cases when maximum weight or volume is reached, and handling trucks with different limitations.

**8.4** Output Message Generation Function:

The output message generation function will be tested to make sure that it creates accurate and informative messages in relation to the truck selection, diversion paths, and other related information. Test cases will cover different allocation scenarios and edge cases to ensure correct and clear output messages are generated.

1. **Resources and Responsibilities**  
   **9.1.** Resources

In the testing phase, the following resources are required:

**a.** Testers: A dedicated testing team who will execute the test cases, compare and document the results, and report to relevant parties when any issues or defects are found.

**b.** Test environment: A suitable testing environment including 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, comparing and documenting the test results, and reporting to relevant parties any issues or defects discovered during testing.

**b.** Developers: The development team are responsible for handling the reported issues and defects, making necessary changes to the code, and retesting the fixes.

**c.** Project Managers: The project managers will monitor the testing activities, ensure appropriate coordination between the testing and development teams, and provide necessary support and resources to improve effective testing.

1. **Deliverables**

**a.** The algorithm should correctly assign packages to trucks based on available space, distance to destination, and diversion required.

**b.** The algorithm should correctly calculate the shortest path from the nearest point on the truck's route to the destination.

**c.** The algorithm should handle cases of encountering obstacles therefore a truck cannot reach the destination.

**d.** The algorithm should correctly print the assigned truck, delivery destination, and any required diversion path.

1. **Suspension / Exit Criteria**  
   Testing may be halted, or the algorithm shut down under the following circumstances:  
   11.1 Inability to allocate packages to trucks based on available space, distance, and diversion requirements.  
   11.2 Incorrect calculation of the shortest path or errors in pathfinding.  
   11.3 Incapacity to handle situations where a truck cannot reach the destination.  
   11.4 Inaccurate printing of essential information.  
     
   Additionally, the algorithm will cease operation if the baggage weight exceeds 1500 kg.
2. **Resumption Criteria**  
   Once identified issues or bugs are rectified, the algorithm should undergo retesting to ensure that deliverables are met. Retesting will involve different scenarios and test cases to validate functionality and accuracy. Any modifications to the algorithm will trigger retesting to confirm that changes do not introduce new issues.
3. **Dependencies**  
   13.1 Personnel Dependencies  
   Identification of project personnel and their roles and responsibilities. Specification of dependencies on the availability or skills of specific team members.  
   13.2 Software Dependencies  
   To conduct successful testing, specific software components are necessary, including the application under test, test management tools, defect tracking tools, systems for tracking and managing defects, and any required automation tools. The stability and accessibility of the software under test, compatibility of testing tools with the software, and adherence to project timelines are critical factors for effective testing.  
   13.3 Hardware Dependencies  
   Specification of any required hardware components for the project, including a list of devices or components that need to be available or connected.  
   13.4 Test Data & Database  
   Access to a suitable test database or environment is crucial for data-related testing. Valid and representative test data covering a range of scenarios, including different package weights, sizes, destinations, and edge cases, is necessary for effective testing.
4. **Risks**  
   14.1. Schedule  
   Risks associated with challenges in meeting testing deadlines, including development delays, scope changes, resource constraints, test data availability, dependencies on external factors, and inadequate time allocation. Mitigation strategies include realistic project planning, clear communication with stakeholders, effective prioritization, and proactive risk management.  
   14.2. Technical  
   Risks that may impact the quality and effectiveness of the testing process, such as infrastructure issues, network problems, hardware limitations, challenges in setting up test environments, tool limitations, and data management concerns. Addressing these risks involves ensuring compatibility, proper configuration of environments, and effective data management. Collaboration with experts and staying updated on tools and security measures can aid in mitigating these risks and improving testing efficiency.  
   14.3. Management  
   Identification of potential risks related to project management and coordination. Assessment of the impact of poor project management on the project's progress.  
   14.4. Personnel  
   Risks at an individual level within the team, including turnover, lack of motivation, communication issues, skill gaps, and personal conflicts. Mitigation strategies involve fostering a positive team environment, encouraging open communication, providing support, and addressing conflicts promptly. Regular meetings and individual support contribute to maintaining a cohesive and motivated testing team, promoting personal growth and satisfaction.  
   14.5 Requirements  
   Identification of potential risks related to unclear, incomplete, or changing requirements. Assessment of the impact of requirements issues on the project's scope and deliverables.
5. **Tools**
6. ****Usage of Visual Studio:** For this project, ***Visual Studio*** would like to be utilized as software development environment (IDE). The Visual Studio provides coding features like C#, C++ which perform debugging and other facilitates, so it would be our key resort during the project. In addition, Visual Studio supports the integration of variety of libraires and APIs, which provide additional features appealing functionality.**
7. **Documentation**
8. **While this project processing, some important phases such as, coding, data structures, algorithms, should be documented, so this way would boost understandabilities of updated changing of codes and future updates. For documentation ***JIRA*** would be key resort for the among of our team.**
9. **Approvals**

**a-**  ****Internal Approvals:** After completion of each development’s steps and testing process, the respected parts of the process should be approved. This confirms that the project is according to quality standards. During the process, we would like to discuss on JIRA and getting approval decision for each steps.**

**b- **External Approvals:** External stakeholders, such as project owner, should check it out ,and approve the project at specific stages. This step contributes to be make sure that the project is on the right track.**