

Test Plan Template

1. **Introduction**
   1. Test Plan Objectives
      1. The purpose of this test plan is to ensure that the delivery system functions efficiently and correctly. The system should assign packages to trucks based on their capacity, the available routes with shortest distance to the destination.
2. **Scope**
3. **Algorithm to verify assigning packages to trucks based on their capacity**
4. **Ensure the algorithm considers the shortest distance to the destination**
5. **Check that the algorithm selects the optimal route for delivery**
6. **Handling fails cases**
7. **Test Strategy**
   1. This section outlines the approach for executing various tests on the truck delivery algorithm. The focus will be on ensuring the functionality of the delivery system, primarily through unit testing, user acceptance testing (UAT), and functional testing.

Stress testing, security testing, recovery testing, and automated testing are not required due to the system requirements. The strategy aims to provide a reliable solution that assures efficient and accurate delivery performance.

* + We will start with exploratory testing to identify critical defects in the early stages, ensuring the application is stable enough for further detailed testing.
  + Functional testing will focus on verifying that all prime functions of the application work as expected, ensuring the delivery algorithm assigns packages correctly based on capacity, distance, and route.
  + Unit testing will test individual components or units of the system in isolation to ensure each part functions correctly, validating the core logic of the delivery algorithm, such as capacity management and route optimization.
  + Integration testing will be conducted using an extra program to test a set of statements using functions from mapping.h, shipment.h to ensure finding routes, reading maps, and assigning shipments execute correctly as a whole picture. This can ensure that different components of the system interact seamlessly.
  + User acceptance testing (UAT) will validate the system against business requirements and user needs, conducted with end-users to ensure the system meets their expectations and performs reliably in real-world scenarios.
  1. The test design process will involve a thorough analysis of system requirements and user stories, engaging with stakeholders to clarify any ambiguities. A requirements traceability matrix will be created to map test cases to specific requirements. Detailed test cases will be developed for each identified requirement, specifying preconditions, test steps, expected results, and postconditions. Test preparation will include defining inputs and expected outputs based on the program's design and behavior. Test cases will be reviewed by other team members, with feedback incorporated in real-time via Jira to ensure completeness and accuracy. Additionally, all task processes will be shared on the Kanban board provided on Jira to ensure that every group member is on the same page and understands the workflow.

1. **Environment Requirements**
   1. Hardware: Mac OS or Windows with the latest version of OS/windows installed. Internet installed for stability resources and remote collaboration, with VPN access for Seneca to use GitHub.
   2. Software: To test and build the program, Intergrated Developer Environment such as Visual Studio, Visual studio code, x-code for mac, depending on the hardware being used. Version Control with GitHub and Jira to communicate within team.
   3. Test harness: tests will be conducted by pre-existing testing tools.
2. **Execution Strategy**
   1. The software should provide a mechanism to build and run tests on a machine.
   2. The test is said to be passed when there isn’t any critical defect.
   3. You can describe the severity of defects in this section and break them down into severity levels of:
      1. **critical** which causes the system to crash or produce anomalous results,
      2. **high** which causes lack of program functionality and might have a work around,
      3. **medium** which degrades the quality of a system but often has a work around to give the desired functionality
      4. **Low** which might be an unclear error message or some other minor error that has minimum impact on functionality
      5. **Cosmetic** which is something that makes the user interface less than optimal but still perfectly functional.
   4. **Test Reporting**
      1. The reports will be created to allow tracking of progress.
      2. **The reports would be produced when there are enough changes in the source code that warrants a run of tests**
      3. **The reports would be sent to everyone in the team.**
      4. The tester will highlight any important feedback and report bugs to the team.
3. **Test Schedule**
   1. **The testing is expected to require 10 days to complete**
   2. **The testing should finish in the third week**
4. **Control Procedures**
   1. Reviews: Regular meetings will allow discussion of progress and align with milestone objectives
   2. Bug Review Meetings: Defects should be discussed to determine the severity and assign a team member for resolution
   3. Change Request: if a specification is changed during testing phase. The documentation for the changes will be created, discussed and modified, followed by approval by the team and implementation.
   4. Defect Reporting: tester will document any identified defects with the description, steps to reproduce, expected output, actual output and any relevant screenshots.
5. **Functions To Be Tested**

Function to find the best truck choice for shipment, and the diverted route.

Function to validate truck can ship a shipment with respect to its capacity.

Functions to validate shipment by weight, size and destination.

Functions to calculate the limiting factor of a truck with an extra shipment.

1. **Resources and Responsibilities**   
   8.1. Resources: Team leader, testers, developers, Test environment includes hardware, software and simulated data to support the test (Jira, GitHub, Visual Studio or Visual Studio code).

8.2. Responsibilities:

1. The team leader should manage the team and project considering schedule.
2. Testers should test cases, record logs and report testing progress.
3. Developers should develop a program that meets the user requirement, implement testing, trace and fix bugs.
4. **Deliverables**

**Use below methods to ensure algorithm assign packages to trucks correctly and searching the optimal route to the detonation**

1. **Test Plan Document: detailed outline of the test strategy, scope, schedule, and resources.**
2. **Test Cases: Individual test scenarios, execution steps, expected outcomes, Comparison with actual outcomes.**
3. **Test Reports: Summarized results of the testing activities, Metrics and analysis, Recorded comparisons of expected and actual data.**
4. **Defect Logs: Identified defects and severity. Steps to reproduce defects and status.**
5. **Traceability Matrix: Document linking requirements to corresponding test cases and ensure coverage of all requirements**
6. **Test Data and Quality Assessment Reports: evaluation of the test process and quality based on test data.**
7. **Documentation updates**
8. **Suspension / Exit Criteria**

A. Suspension criteria:

1. The program cannot block the invalid user input.

2. The chosen truck exceeds its capacity with the given package.

3. The chosen truck is not the best option for shipping to the destination as the shortest route.

4. The generated diverted route cannot connect to the destination from the truck route.

5. The generated diverted route passes through buildings.

6. The limiting factor calculated with the truck and an extra package is incorrect.

B. Exit Criteria:

1. The program can validate parameters received from users.

2. The program can assign the best truck for shipments

3. The program can generate correct diverted routes.

4. The program can calculate the limiting factor correctly.

1. **Resumption Criteria**
2. **Bug Fix Confirmation**:
   1. After a bug is fixed, the testing process should resume from the beginning to ensure the issue is fully resolved.
3. **Comprehensive Testing**:
   1. After a bug is fixed, the program should be tested using multiple methods to ensure accuracy and validation.
4. **Environment Restoration**:
   1. Once the test environment is rebooted or fixed, the testing process can restart.
5. **Data Availability**:
   1. Once the required test data is printed and verified, the testing process can be resumed.
6. **Dependencies**  
   12.1 Personnel Dependencies

There may be dependencies between team members due to availability or skillset of the team members and the nature of the task  
12.2 Software Dependencies

Software dependencies refer to any prerequisites that the software requires before a successful build and execution can be conducted. The dependencies should be listed so that the build and execution result can be accurately repeated.  
12.3 Hardware Dependencies

Hardware dependencies refer to any hardware that is required for the task and has to be listed out.  
12.3 Test Data & Database

Availability of test data and a database server for testing allows testing of software which has database functionality.

1. **Risks**  
   13.1. Schedule: the schedule is potential to be delayed or not meet the due time.

13.2. Technical: scope changes, resource constraints and shortage of test data can affect the test results.  
13.3. Management: oversight of the project, failure in managing test cases and documentary (lost data)  
13.4. Personnel: team members’ availability and skills   
13.5 Requirements: risk related to changes in project requirements, different interpretations of the project scope and objectives.

To prevent and handle these risks, the team should adhere to the Software Testing Life Cycle (STLC), which includes the following phases: Requirement Analysis, Test Planning, Test Case Development, Environment Setup, Test Execution, and Test Closure Activities. Each phase should be meticulously documented, with well-written test cases and comprehensive reports. Additionally, the team should fully utilize software tools to facilitate effective communication and collaboration within the team. This structured approach will help mitigate risks related to scheduling delays, technical challenges, management issues, personnel availability, and requirement changes or misunderstandings.

1. **Tools**

**Test environment tool: Visual Studio, Visual Studio Code, CMake**

**Version control: GitHub**

**Communication: Jira**

1. **Documentation**

To ensure the program runs correctly and efficiently identify and fix bugs, it is essential to understand and follow several key documents. These include the test cases report, test scripts, defect logs, user guides (if needed), test strategy, detailed test plan, traceability matrix, and Kanban board. Each document plays a vital role in guiding the testing process and maintaining project organization.

1. **Approvals**

Identify the individuals or stakeholders responsible for approving the test plan and associated documents. Establish a clear process and timeline for obtaining their approval, ensuring that all necessary parties have reviewed and provided their consent before initiating the testing activities.