Test Plan Template

1. **Introduction**

Test Plan Objectives**.**

The project involves creating a package delivery application for a local delivery company. This company has three different trucks, each assigned to its own specific route. When a cargo comes in, the program determines which vehicle can carry the box closest to its destination while simultaneously having enough room for it. The customer has provided the following information: the shipment's weight (kg), the box's size (cubic metres), and the destination (row number, column letter, for example, 3T).

1. **Scope**

The scope of testing for the delivery management system includes various things we want to check. We want to make sure the system can assign shipments to trucks correctly based on weight, size, and how close they are to the destination. It should also figure out how much stuff the trucks can carry, find the best routes, and know what to do when a truck can't reach a place. We'll test it in different situations, see if it uses resources well, handles problems, follows the rules, and gives us the right reports. It also needs to work with other systems, be easy to use, keep data accurate, and deal with issues safely. We'll check how it performs under a heavy workload, ensure it's secure, follows the rules, and review the system documents. But we won't be looking at truck dispatch, how it works with other systems, or how people are trained to use it. This scope helps us understand what's part of the testing and what's not.

1. **Test Strategy**
   1. To guarantee the quality and dependability of the local delivery system, a thorough approach will be taken in both the test execution and methodology. The purpose of the system test is to verify the entire functioning, which includes route optimisation, shortest path computation, and proper package assignment to trucks. In order to do this and guarantee the resilience of the system, a variety of test data will be created to mimic various situations and edge cases. The QA team is in charge of carrying out the system test; they will keep a close eye on and evaluate the behaviour of the system.

In order to assess the system's responsiveness and efficiency under varied workloads, performance testing will be essential. To assess the system's performance at maximum capacity and under extreme stress, volume and stress tests will be used. The QA team will evaluate the system's scalability, reactivity, and stability by creating test data that mimics extreme circumstances. Through this kind of testing, any possible performance problems may be found and the system's resilience and dependability—even in the face of extreme conditions—can be confirmed.

The goal of recovery testing is to evaluate the system's resilience to errors or disturbances. In order to evaluate the system's recovery mechanisms, including error handling and fault tolerance, the QA team will purposefully generate failures. technical material, installation instructions, and manuals. To guarantee consistency and clarity, the QA team will examine and verify the documentation against the features of the live system.

* 1. **You could describe the test design process and give an overview of how it will be conducted. You could provide a broad overview of**

**Ensuring comprehensive software testing for the local delivery system is mostly dependent on the test design process. To properly prepare and carry out the tests, there are a number of crucial processes involved.**

**Understanding the system requirements in their entirety is the first step. This entails examining the software's functional and non-functional features to determine precisely what needs to be evaluated. The testing team can make sure that all important system components are tested throughout the process by having a thorough grasp of the requirements.**

**The creation of a traceability matrix is the next stage. This matrix is a useful tool for connecting the relevant test scenarios with the system requirements. The testing team may make sure that every requirement is appropriately addressed and that all test cases correspond with the software's intended functions by generating this matrix. Additionally, it aids in locating any holes or discrepancies in the test coverage.**

**The testing team creates test cases covering several scenarios and tests the program against varying inputs and expected outputs after establishing the traceability matrix. In addition to providing complete test coverage with edge cases and boundary conditions, these test cases also serve as a reference for testing the functionality of the local delivery system. They also provide explicit instructions for running tests. Another member of the quality assurance team must be included in the review process to preserve the integrity, precision, and comprehensiveness of the test cases and to maintain the quality of the test design. In addition to verifying alignment with system requirements, this review process aids in identifying possible gaps and improvements. By taking these precautions, the local delivery system is thoroughly tested, lowering the possibility of hidden flaws and improving system performance.**

1. **Environment Requirements**

Procedure for Setting Up the Test Environment: In order to install and configure the required software and tools on the test workstations and PCs, comprehensive instructions should be provided. Guidelines for data preparation and validation procedures are also essential to guarantee the correctness and preparedness of the test environment.

Hardware Configuration: The test environment must have correctly configured test workstations and PCs, as well as reliable internet access, for network-related testing.

Software requirements: The test environment should have compatible development environments, operating systems, and specialised testing tools. Simulation tools may also be used to recreate package data, trucks, routes, and city maps.

The test environment should have predefined or created city map data, sample package data, and numerous test datasets representing different scenarios in order to conduct thorough testing.

1. **Execution Strategy**
   1. Particular entry and exit criteria have been set in order to guarantee a methodical and efficient testing procedure in the package distribution project. The requirements listed in the admission criteria must be fulfilled in order to start the testing process. These include having a clear understanding of the functional and non-functional requirements of the project, setting up the hardware, software, and data in the test environment correctly, creating a thorough test plan and accompanying test cases, having enough test data available, allocating the required test resources, putting a defect management system in place, and defining a test schedule.
   2. The development team may more efficiently organise and prioritise their efforts to fix the most serious issues that can have a major impact on system operation and user experience by classifying defects into severity categories. Through early defect resolution, the team can guarantee system stability and avert significant interruptions.

Even though they are not as serious as critical or high-severity flaws, medium-severity defects nevertheless need to be fixed in order to preserve the overall quality and functionality of the system. By fixing these flaws, which may have an impact on particular features or performance metrics, the user experience and system operation may be enhanced.

* 1. **Test Reporting**

Various reports are necessary in the package distribution project to convey and record the testing progress and outcomes. After every test cycle, test execution reports are produced. These reports provide a description of the test cases that were completed, faults that were found, test coverage, and suggestions for more testing. Each detected issue is included in a defect report, along with its description, severity, status, timetable for resolution, and any other remarks. Test activities, coverage metrics, outcomes, risk assessment, and timetable updates are all tracked in test progress reports, which also provide an overall view of the testing process. The project schedule and stakeholder needs determine how often reports are generated and who receives them. To promote efficient communication and decision-making, reports should be distributed to all project stakeholders, such as project managers, development teams, and quality assurance teams.

1. **Test Schedule**

|  |  |
| --- | --- |
| **Milestone** | **Activities** |
| **1** | * **Review project requirements and testing objectives.** * **Define test environment and tools setup.** * **Create the test plan and obtain necessary approvals.** |
| **2** | * **Design and document test cases based on identified scenarios.** * **Map test cases to requirements for traceability.** * **Review and refine test cases for accuracy and completeness.** * **Prepare necessary test data and test environment setup.** * **Creating Blackbox testing** |
| **3 - 4** | * **Execute test cases based on the defined test plan.** * **Record test results and capture any defects or issues.** * **Verify system functionality against expected outcomes.** * **Conduct regression testing as needed.** * **Iteratively execute test cases and retest resolved defects.** |
| **5** | * **Log defects and issues identified during test execution.** * **Assign severity levels and prioritize defect resolution.** * **Collaborate with the development team to investigate and fix defects.** * **Retest resolved defects and update defect status.** |
| **6** | * **Finalize test execution and ensure all planned tests are completed.** * **Compile test results and generate test summary reports.** * **Conduct a final review of the testing process and outcomes.** * **Share testing reports with stakeholders for review and feedback.** |

1. **Control Procedures**

**Various reports are necessary in the package distribution project to convey and record the testing progress and outcomes. After every test cycle, test execution reports are produced. These reports provide a description of the test cases that were completed, faults that were found, test coverage, and suggestions for more testing. Each detected issue is included in a defect report, along with its description, severity, status, timetable for resolution, and any other remarks. Test activities, coverage metrics, outcomes, risk assessment, and timetable updates are all tracked in test progress reports, which also provide an overall view of the testing process. The project schedule and stakeholder needs determine how often reports are generated and who receives them. To promote efficient communication and decision-making, reports should be distributed to all project stakeholders, such as project managers, development teams, and quality assurance teams.**

1. **Functions To Be Tested**

1) int getNumRows(const struct Map\* map);

2)int getNumCols(const struct Map\* map);

3)void printMap(const struct Map\* map, const int base1, const int alphaCols);

4)struct Map addRoute(const struct Map\* map, const struct Route\* route);

5)(void addPtToRoute(struct Route\* route, struct Point pt);

6)void addPointToRouteIfNot(struct Route\* route, const int row, const int col, const struct Point notThis);

void addPointToRoute(struct Route\* route, const int row, const int col);

7)struct Route getBlueRoute();

8)struct Route getGreenRoute();

9)struct Route getYellowRoute();

1. **Resources and Responsibilities**

**The following resources are needed to satisfy the requirement for a section that enumerates team members, their roles, and obligations related to testing activities:**

**1.Active team members. 2.Testing tools and environment 3.frequent team meetings**

**4.Proper documentation**

**The Responsibilities needed to satisfy the testing activities are :**

1.Defining team member roles 2.Allocating task on behalf of role 3.Meetings 4.Task execution

5.Progress reporting 6.Review and improvement 7.Documentation.

1. **Deliverables**

**Every group member must complete their respective assigned tasks and must pass them on to the group leader for final check. Group leader will submit the final versions of the tasks on GIT. Every milestone has different files associated with being submitted.**

1. **Suspension / Exit Criteria**

**This part is about when we should stop or take a break from testing. We have some conditions that tell us when to do this. These conditions include finishing the planned tests, fixing important issues, meeting performance goals, and making sure the system matches the requirements and quality standards. If we have big problems that need fixing right away, we might take a break. But when we complete all our testing goals and the system is ready for the next steps, we'll stop testing. These conditions help us make good decisions about when to continue or finish testing.**

1. **Resumption Criteria**

The Resumption Criteria in our project outlines the conditions for restarting testing after a temporary pause. This section details specific circumstances or events that trigger the need for change, identifies the decision-maker for suspending testing, sets the criteria for testing to resume (like fixing critical defects or meeting performance goals), and establishes an escalation plan if the criteria can't be met within a specified timeframe. These criteria ensure that testing can restart in a well-informed and controlled manner, keeping the project on track and aligned with quality standards.

1. **Dependencies**

In our project, dependencies play a crucial role in ensuring the smooth execution of testing activities. These dependencies can be categorized into the following aspects:

\*\*12.1 Personnel Dependencies\*\*:

This part talks about how we need the right people to do the testing. These people include the testers, the test managers, and experts who know about the project. We need to make sure these people are available and trained properly to do the testing well. Having the right team is super important for successful testing.

\*\*12.2 Software Dependencies\*\*:

This part is about the computer programs we need for testing. We rely on other software to help us with testing. This software can be things like testing tools, programs that automate tasks, or third-party software that we use for running tests, creating data, or making reports. It's important to know and control these software dependencies so that our testing work goes smoothly and doesn't get disrupted.

\*\*12.3 Hardware Dependencies\*\*:

This section is about the physical equipment we need for testing. It includes things like computers, devices, and the setup for performance testing. We must make sure that we have the right hardware and that it's working well. If we don't, it can slow down or stop our testing work. So, having the necessary hardware in good shape is important to keep testing running smoothly.

\*\*12.4 Test Data & Database\*\*:

This part is about the information we need for testing. It's like having the right ingredients for a recipe. We rely on data and databases to test our system. So, it's important that we have the right data, that it's current, and that it matches what we want to test. This ensures that our testing makes sense and works properly. Having the correct and up-to-date data is crucial for good testing.

1. **Risks**

In our project, several categories of risks need to be addressed to ensure successful testing and project execution:

\*\*13.1 Schedule\*\*:

Schedule risks are like the chance of things taking longer than planned in our project. It could happen because the project schedule changes suddenly or the scope of what we're testing gets bigger. To deal with this risk, we need to plan carefully and have backup plans ready in case the schedule gets messed up. This way, we can handle any unexpected delays and keep our project on track.

\*\*13.2 Technical\*\*:

Technical risks are about problems with the technical stuff in our project. This can be issues with the software or hardware we're using, trouble with making different parts work together, or not having the right tools and resources. To manage these risks, we need to be proactive by checking the technical parts carefully and getting ready to deal with any issues. It's like making sure all the gears in a machine are working smoothly to prevent any technical hiccups.

\*\*13.3 Management\*\*:

Management risks are about problems in how we lead and organize our project. This can include issues with project leaders, problems with communicating, or not everyone working toward the same project goals. To avoid these risks, we need to have good project management practices in place. This means clear communication, knowing who does what, and making sure everyone is on the same page. It's like having a well-organized team to steer the project in the right direction and avoid problems.

\*\*13.4 Personnel\*\*:

Personnel risks are about the people working on our testing team. It can be issues like not having enough team members, some team members not having the right skills, or conflicts among team members. To handle these risks, we need to make sure we have the right people with the right skills, and they work well together as a team. It's like having a skilled and happy team that can do the job effectively and smoothly.

\*\*13.5 Requirements\*\*:

Requirements risks are about the things we need to do in our project. Sometimes, these things can be unclear or change over time. It might happen because the customer's needs change, or the project's goals shift, or we didn't define the requirements properly. To deal with these risks, we need to manage our requirements well, talk to the people involved regularly, and be flexible to adjust our plans when things change. It's like making sure we know exactly what we need to do and being ready to adapt when the plans change.

1. **Tools**

**For our project, we will utilize several software tools and technologies to support the testing process. These include:**

**1. Jira: We will use Jira for project management, issue tracking, and task assignment, helping us stay organized and manage project-related information efficiently.**

**2. GitHub: GitHub will be our platform for version control and collaboration, allowing team members to work together, share code, and track changes effectively.**

**3. Excel: Microsoft Excel will serve as a tool for data management, test case tracking, and reporting, helping us analyze and document our testing efforts.**

**4. Visual Studio: Visual Studio will be used for test automation and development tasks, enabling us to create and execute automated tests seamlessly.**

1. **Documentation**

Throughout the testing process, we will rely on various documents to ensure effective planning, execution, and reporting. These documents include the Test Plan, which outlines our testing strategy and objectives, scrum report and the source code representing the final project. These documents will play a critical role in documenting and managing our testing activities, ensuring clarity, and facilitating comprehensive reporting.

1. **Approvals**

**In our project, we will follow a structured approach to ensure that the necessary approvals are obtained at key stages of the testing process. This includes seeking approval for the Test Plan, Test Cases, and Test Reports. These approvals are essential to maintain effective communication, traceability, and quality assurance, providing a formal acknowledgment of the readiness and correctness of our testing activities.**