Test Plan Template

1. **Introduction**
   1. Test Plan Objectives
      1. **This section should describe the project that is being tested and what are the objectives for the test. You might give a quick overview of the project to tell people what it's about, and then describe the testing at a very high level and what do you expect to get as a result of the testing.**

The project is a Mapping Software project which is designed to provide a map representation along with delivery optimization capabilities for the company’s 3 trucks. The objective of the testing phase is to ensure that all the functions and structures of the software are properly working and are able to handle common types of user errors. The testing includes testing the integrity of the route definitions, route population, and functionality of the map.

1. **Scope**
   1. **In this section you can describe what will be tested and what will not be tested.**

The testing scope for this project includes all the functions and structures that are defined in the code base.

* Functionality – Testing to see how functional the program is and to see if all the functions and structures are being used properly and do not break down.
* Capacity – Testing to see that the functions are able to hold the capacity that they are supposed to hold and not overflow the functions.
* Accuracy – This would be tested to see that all the functions and structures are working exactly as they should be created to do.

1. **Test Strategy**
   1. **This section describes the approach you will take to perform the tests. There are sections below where you can elaborate on different types of tests. Not all these types of tests will be in every project and some projects might have tests which are not listed below. This would be a good section to describe where the test data is being obtained from. You could also describe the different levels of testing which might be used. For example, testing is often broken up into exploratory testing which attempts to make sure that critical defects are removed before the next level of testing begins. After exploratory testing catches some of the big critical defects you can go on to functional testing as the next testing cycle to make sure that all the prime functions of the application are being delivered correctly. You can continue to describe all the test deliverables and what roles are responsible for producing and delivering these. You could also include an estimate of how long it is going to take to do the testing.**  
      3.1. System Test – This can be used to test how the program works and make sure that the system runs, works and most of the functions and structures are working. The estimated time to do the test should be around 5-10 minutes.  
      3.2. Performance Test – This can be used to test how the function speed and response are like and how each component in the program is used. The estimated timing to do the test should be around 10-15 minutes.  
      3.3. Security Test – The security test can be used to test the access that  
      3.4. Automated Test -   
      3.5. Stress and Volume Test  
      3.6. Recovery Test  
      3.7. Documentation Test  
      3.8. Beta Test  
      3.9. User Acceptance Test
   2. **You could describe the test design process and give an overview of how it will be conducted. You could provide a broad overview of** 
      1. **how to understand requirements,**
      2. **build a traceability matrix,**
      3. **prepare test cases,**
      4. **and have them reviewed by another member of the quality assurance team.**
2. **Environment Requirements**
   1. This section will typically define the hardware and software environment necessary for the tests to be conducted. This could involve specifying that a test computer is necessary to run the tests in a continuous integration process or it might say that all testing is done on the developers' workstations. Test harness might need to be built to conduct the test or you might be using a pre-existing set of testing tools. All of this needs to be laid out with all its requirements so that the testing environment can be set up before the testing begins.

Software – Software that can be used on multiple different platforms. Software that can handle the programs that test tools that will be used for unit testing and other testing tools.

Hardware- For hardware, we need hardware that can run the development tools to conduct the tests on. Also able to maintain the environment by updating and debugging the issues that the program may have and validating functions and structures to make sure that the program is validating to make sure that the program works as it’s supposed to.

* 1. More requirements for our Mapping Software project would be to follow the below prompts:
  + Ensure that all trucks cannot drive through any of the black buildings on the map.
  + Trucks (blue, green and yellow) are available to deliver packages.
  + All truck sizes are the same and can hold up to 1200 kilograms of cargo.
  + Trucks are capable of carrying 50 cubic meters of square boxes used for shipping, each of which are ½ cubic meter, 1 cubic meter, and 5 cubic meters sizes.
  + For customer requirements, ensure the shipment weight is in kilograms, the box size in cubic meters and specify the box destination in terms of building (row number and column letter).
  + For every shipment, find a truck big enough to hold the shipment, and choose the truck closest to the package destination.
  + If 2 trucks are the same distance away from the destination, always put the package in the truck which is less full.
  + And lastly, if all trucks are full (that is, when it hits its maximum weight or volume), the delivery will be made the next day.

1. **Execution Strategy**
   1. this is the section where you will describe heavy chests are actually executed. You can describe what the entry and exit criteria for the tests are. For example you might be able to exit a test if it passes 95% of test scripts. In another situation, you might want to pass 100% of the tests. Or perhaps you want to declare but a test is completed if there are no severe or critical defects.
   2. You can describe the severity of defects in this section and break them down into severity levels of:
      1. **critical** which cause 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 is a bug which D crates 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.
   3. **Test Reporting**
      1. This action will describe what sort of reports should be produced as a result of testing, how often these reports should be produced, and to whom the reports should be sent. It should give some indication of the contents of the reports and under what conditions the reports are generated. You might say that a manager receives a daily report of the number of tests conducted, passed, and failed with a brief description of the areas being tested and the areas which are failing.
      2. This section could also have details of how the testers are going to feed information back to the project managers so that they can assign developers to fix the bugs. This section can detail the communication to occur between management, the development team, and the quality assurance team.
   4. You can also explain how the quality assurance team we'll be able to interact with the developers and how they will be able to work with the developers to resolve the defects found in the software.
2. Execution strategy will help with doing test cases when doing testing for the programs. When we start a testing, we should have the exiting criteria should be more than 95% and the entry criteria should be able to pass 100% of the testing cases.
3. If the entry criteria are not able to pass 80% of the testing, it should have a medium severity defect in the program. If the exit pass rate is lower than 70-80%, it would have a high severity defect in the program that would need to be fixed.
4. After each test case is performed, it should give a test report to show what happened and what error/bug and severity defect level has occurred so it can help manage the program and help fix the bugs/error that has occurred.
5. Having a quality assurance team allows communication to flow better and helps everyone in the project. Also helps to communicate what has happened in the testing so it can be helpful for everyone working in the project and allows everyone to have an understanding on what happened.
6. **Test Schedule** 
   1. **This is the section where you write layout a schedule for the testing and be able to give an estimate of how long the testing will take and approximately when it will be complete.**

|  |  |  |
| --- | --- | --- |
| **Testing** | **Description** | **Estimate Time** |
| **System Test** | **Testing out the program system** | **5-10 Minutes** |
| **Performance Test** | **Testing how the functions and program works** | **10-15 minutes** |
| **Stress and Volume Test** | **Testing to see how much the program is able to take** | **15-20 minutes** |
| **User Acceptance Test** | **Testing to see how the test would work by a user point of view** | **5-10 minutes** |
| **Documentation Test** | **Testing to see that the test looks like the document (its intended use)** | **10-15 minutes** |

1. **Control Procedures**
   1. 6.1 Reviews – All the test cases will be reviewed by another member of the team before being finalized.   
      6.2 Bug Review Meetings – Regular meetings will be set up to talk about bugs found.   
      6.3 Change Request – Change requests would be implemented only after approval  
      6.4 Defect Reporting – All defect reports would be reported using a tool defined by the organization.
2. **Functions To Be Tested**

All structures that are defined in the Mapping Software code base will be tested. These structures are

- struct Route getPossibleMoves(const struct Map\* map, const struct Point p1, const struct Point backpath);

- struct Route shortestPath(const struct Map\* map, const struct Point start, const struct Point dest);

- int getClosestPoint(const struct Route\* route, const struct Point pt);.

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

All project documents inside the project-starter.zip and the provided templates  
8.2. Responsibilities

We have group contract to stipulate everyone’s responsibilities in the group and list the consequences if they violate any rules.

1. **Deliverables**

This depends on the milestone requirements that needs to be submitted/uploaded to GitHub repository.

The code should correctly assign the packages to the trucks based on available space, distance to destination, and the diversion required.

The code should calculate the shortest path from the nearest point on the truck's route towards the destination.

The code should be able to handle cases where a truck cannot reach its destinations due to obstacles and if the weight capacity is reached.

1. **Suspension / Exit Criteria**

11.1 Critical bugs found in another function which can cause the system to crush

11.2 Unavailability of necessary resources (e.g., hardware)

11.3 Changes in the requirements of the software

11.4 Time constraints that require prioritization of certain tasks

11.5 All specified requirements have been met

11.6 Defects have been fixed

1. **Resumption Criteria**

**12.1 all critical bugs are fixed**

**12.2 necessary resources are available**

**12.3 requirements of the software have been confirmed**

1. **Dependencies**  
   12.1 Personnel Dependencies  
   12.2 Software Dependencies  
   12.3 Hardware Dependencies  
   12.3 Test Data & Database
2. **Risks**  
   13.1. Schedule – Tight deadlines and unavailability of group mates may cause missing meetings, or deadlines.  
   13.2. Technical - changing one part of the code might affect other parts of the system, which may lead to unforeseen bugs or failures.   
   13.3. Management – allocating appropriate resources like time to fix the code was a challenge.  
   13.4. Personnel – Skill gaps. One person might not be effective as another during the process.  
   13.5 Requirements – there might be a scenario where the requirements are misinterpreted and hence a fix that was hence implemented may not be what is required or incomplete.
3. **Tools**

The tools that were used for the testing phase are the C compiler and debugger. We also used breakpoints/F key binds to run through the code.

1. **Documentation**

Documents are created when testing, for example, test cases, test reports, test scripts, and logs. Each of these documents have a different purpose and provide important information. It is important to follow the guidelines written in these documents to understand them correctly.

1. **Approvals**

Identify individual roles to be responsible for the test plan and other documents.

Set a date and meeting to talk with one another to obtain approval with every group member to ensure that everyone is on the same page before beginning the test activities.