Test Plan Template

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

**Our group vision is finding the most efficient route or path for truck to deliver the package from the offices to the destination. And should be include three important things: the shortest route, can delivery on every possible location, ensure that the limit about the weight of the truck meet the requirement**

1. **Scope**
   1. **The test will not follow the real time things such as schedule day for shipping, because this is not the aim of this project.**
   2. **We will try to test based on the description, basically the project have 3 colors for 3 line, and what we must do is trying to determine which line (color) will be go first, which point that the truck can move to change the line. The shortest path is the most important to test in this project**
2. **Test Strategy**
   1. 3.1. Performance Test

The program should not be taken too long to display the output for user, also should be worked efficient as much as possible.  
3.2. Security Test

The program makes sure that there will not be any leak information during the program running and will not ask for any information from user  
3.4. Integration tests   
 The program makes that all function will work together as expected  
3.5. Stress and Volume Test

The program should be work under the heavily load of multiple that will run at the same time with the program.  
 3.6. Regression test

The program with additional function will still work well with the existing function  
 3.7. User Acceptance Test

This test will make sure that the user will have no problem when run the program and should work as it guided.

1. **Environment Requirements**

**The testing for the project will run on Window 10 and 11 and should work well with Visual Studio.**

1. **Execution Strategy**
   1. The test will cover all the function inside the project.

ONLY EXIT WHEN:

100% crucial functions has tested successfully and at least makes sure that some rare cases make by user will be tested successfully.

* Maps featuring predetermined building configurations (ex: populateMap function outputs from **mapping.c).**
* Truck configurations including varying capacity, colors, and beginning routes (defined in Truck.h).
* Packages with differing weights, dimensions, and destinations (Package struct from Truck.h).
* Routes (blue, green, yellow) produced utilizing functions like getBlueRoute, getGreenRoute, and getYellowRoute from mapping.c.
* Shortest routes computed with the shortestPath function in mapping.c for verification.
* Truck constraints, including:
  + Maximum weight and volume limits (LIMIT\_WEIGHT, LIMIT\_VOLUME).
  + Package handling: adding, removing, and validating (addPackage, canAddPackage).
  1. 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.

1. **Test Schedule**
   1. **This is the section where you wrote 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.**
      * 1. **MS1**

**Set up Jira, Github accounts.**

**Reading and understanding the problem of the project**

* + - 1. **MS2**

**Analysis the project**

**Adding needed structure in new header file to serve for the next upcoming milestone**

* + - 1. **MS3**

**Blackbox testing**

**Set up C++ testing naïve unit test**

**Doing the declaration for function if needed**

**Unit Testing and small integration testing**

* + - 1. **MS4**

**White box testing**

**Create a sheet to store fail or success test case**

* + - 1. **MS5**

**Integration test**

**Performance test and stress test**

**User acceptance test**

**Regression test**

* + - 1. **MS6**

**Debugging**

**Final testing**

**Prepare all the file and documents**

**Ending the project**

1. **Control Procedures**
   1. 6.1 Reviews

We will review the test plan and also the strategy and the kanran board on Jira to make sure that the project went well.  
6.2 Bug Review Meetings

The schedule will be made every week after the function has successfully implemented to test and see the issues of the program   
6.3 Change Request

If there is any problem with additional functions, the functions will be considered one more time, if it still not work with existing functions, we will fixed it and apply the new version later.

The test plan and the documents will also be subject to changed if anyone in the group facing any problem and want to change  
6.4 Defect Reporting

Make suer that the reporting will be detailed as much as possible about all the things we used to test the project

1. **Resources and responsibilities**

8.1. Resources

* Besides using only, the resources that are provided by the project, we will be considered to research more in the world wide web, to perfect the project.   
  - Also, we might try to dive into the algorithm used in this project, which is heavily based on mathematics.

8.2. Responsibilities

* We will not be copying the content and pasting it into the project, all the thing (If it have) will be only presented based on our knowledge that we read or learn in another resources.
* We will include the resources that we are using (if we have used in this project) to make sure that there will be no violation of copyright.

1. **Resources and Responsibilities**  
   8.1. Resources  
   8.2. Responsibilities
2. **Deliverables**

**The testing deliverables include:**

* **Test cases and results for:**
* Validation of functionality in route computations and pathfinding.
* Performance evaluation for extensive maps and various truck situations
* Logs of tests detailing all successful, unsuccessful, and edge-case occurrences.
* Record of problems and solutions encountered during testing.
* A conclusive test summary report including the testing period and outcomes.

1. **Suspension / Exit Criteria**

The test will be suspended if:

* Any major flaw obstructs additional testing of essential features, including shortest path computation and truck capacity validation.
* Environmental concerns include the malfunction of the testing platform or tools.

Exit criteria for the test include:

* All essential and high-priority test cases have been successfully executed.
* At a minimum, 95% of medium-priority test cases are successful.
* There are no outstanding critical problems, and all major bugs have been fixed.
* A final test summary report will be generated and authorized.

1. **Resumption Criteria**

* Testing will resume after suspension if:
* All significant concerns leading to suspension have been rectified and confirmed.
* The testing environment has been reinstated and is stable, encompassing tools, dependencies, and resources.
* Essential modifications or corrections to the code (e.g., shortest path logic or package management functions) have been executed and subjected to unit testing.
* All test cases pertaining to resolved problems have been re-executed and successfully passed.

1. **Dependencies**  
   13.1 Personnel Dependencies  
   13.2 Software Dependencies  
   13.3 Hardware Dependencies  
   13.3 Test Data & Database
2. **Risks**  
   14.1. Schedule

* Prolonged timelines in establishing feasible map layouts or truck configurations for testing.
* Delays occur in rectifying major faults, particularly those associated with route computations or shortest path algorithms.

14.2. Technical

* Errors or inaccuracies in the shortest path computation (shortestPath function in mapping.c), particularly in complex maps.
* Integration problems among the map, route, and truck components, resulting in unforeseen behavior during testing.

14.3. Management

* Miscommunication regarding functional requirements, like truck capacity or routing logic.
* Identify potential risks related to project management and coordination.
* Evaluate the effects of inadequate project management on the advancement of the project.

14.4. Personnel

* Individual risks in testing encompass obstacles inside the team, including turnover, diminished motivation, communication barriers, skill deficiencies, and interpersonal disputes. To avoid these risks, it is essential to cultivate a constructive team atmosphere, promote transparent communication, offer assistance, and resolve problems swiftly. Consistent meetings and personalized assistance foster a unified and motivated testing team, enhancing individual development and fulfillment.

14.5 Requirements

* Identify possible risks associated with ambiguous, insufficient, or evolving requirements.
* Evaluate the influence of requirements-related problems on the project's scope and outputs.

1. **Tools**

* Testing tools are crucial for effective and precise evaluation. Their components encompass test management tools for planning and monitoring tests, defect tracking tools for overseeing software bugs, performance testing tools for evaluating system performance, and test data management tools for generating test datasets. Selecting appropriate tools can improve testing efficacy and productivity.
* Visual Studio: For compilation, debugging, and running test cases.
* GitHub: Version control and collaboration.

1. **Documentation**

* During testing, several documents are generated, including test cases, test scripts, test results, defect logs, and user manuals. These publications serve distinct objectives and convey essential information. The format and structure of these documents may differ, and certain templates or guidelines may be supplied for adherence. Students must comprehend and adhere to these templates and criteria to document the examination procedure efficiently and convey the results correctly.

1. **Approvals**

* Determine the persons or stakeholders accountable for endorsing the test plan and related papers. Establish the procedure and schedule for securing their permission, ensuring that all requisite stakeholders have evaluated and granted consent prior to initiating the testing operations.