Test Plan MS-2 GROUP-B

1. **Introduction**
   1. Test Plan Objectives
      1. Evaluate Pathfinding Performance: Assess the efficiency and accuracy of pathfinding algorithms, particularly on large and complex maps.
      2. Verify Map Initialization and Route Management: Ensure accurate initialization of maps and effective management of routes, including their addition and manipulation.
      3. Test System Resilience: Confirm the system's ability to handle edge cases and invalid inputs gracefully.
2. **Scope**
   1. **What Will Be Tested / In-Scope**
      1. Map Initialization: Verify correct initialization with predefined values.
      2. Route Management: Ensure routes are accurately defined and added.
      3. Pathfinding Algorithms: Test correctness and performance in finding shortest paths and handling obstacles.
      4. Distance Calculations: Check the accuracy of Euclidean distance calculations.
   2. What will not be tested / Out-Of-Scope
      1. Paths Outside the Grid: Testing will not cover scenarios where routes or paths extend outside the defined grid boundaries.
      2. Integration with Other Systems: Testing will focus solely on the map and route management system in isolation, without considering its integration with other systems.
      3. User Interface: Since the project is a backend system, user interface testing is not applicable.
3. **Test Strategy**
   1. **We will use unit tests with the Microsoft C Unit Testing Framework to verify individual functions. Integration tests will check how modules interact, and system tests will cover end-to-end evaluations. Performance tests will assess pathfinding efficiency on large maps, and robustness tests will check handling of edge cases. Both black-box and white-box testing methods will be employed. Exploratory testing will help find unexpected issues. The testing timeline is one week each for unit, integration, system, and performance/robustness testing. Deliverables include a test plan, test cases, and test reports. Security, recovery, and beta tests are not included.**
      1. Black-Box Testing / Exploratory Testing
         1. Stress and Volume Testing: Evaluating system behaviors and simulate user interactions with edge cases.
         2. Documentation Testing: Review user documentations and manual making sure it provides clear and comprehensive instructions for the program implementation.
      2. White-Box Testing / Functionality Testing
         1. Acceptance Testing: Confirming that the program meets requirements and expectations of the client with real world scenarios.
         2. System Testing: Uses Integration testing to make sure that the functions all work together as a whole package, working together to meet requirements.
   2. Test Design Process
      1. First, we will analyze the project requirements by reading the specifications from the pdf file and the provided code to ensure everyone understands what is needed. Key features, Expected results and behaviors, functionalities, etc..
      2. Build traceability matrix. To establish links between requirements and the cases. This is mainly to keep track of the requirements and what needs to be done.
      3. Test Cases Preparation. Used the requirements to develop test cases covering edge cases, scenarios and boundary conditions.
      4. Review. We will finalize the project and have somebody give feedback on the coverage of the test cases and the coverage of the program to look for any bugs or improvements.
4. **Environment Requirements**
   1. **Windows or a Mac operating system that have sufficient specifications below to run the software being tested.**
      1. **Visual Studios 2022.**
      2. **Access to GitHub via GitBash/TortoiseGit.**
      3. **Access to Jira and KanBoard on Atlassian.**
   2. **Microsoft C Unit Testing Framework for automation testing and assert statements**
5. **Execution Strategy**
   1. We will begin testing once the code is complete, reviewed, and the test environment is ready. Unit, integration, and system tests will be executed using the Microsoft C Unit Testing Framework, followed by performance and robustness tests. Testing will be considered complete when 95% of test cases pass, and all critical and high-severity defects are resolved.
   2. Severity Levels:
      1. **Critical** defects cause crashes or major issues and must be fixed immediately.
      2. **High** defects significantly impact functionality but may have workarounds.
      3. **Medium** defects affect quality but usually have workarounds.
      4. **Low** defects are minor issues with minimal impact.
      5. **Cosmetic** defects only affect appearance.
   3. Addressing these will ensure the system works well and meets user needs.
   4. **Test Reporting**
      1. We will produce daily test reports detailing the number of tests conducted, passed, and failed. These reports will include a brief description of the tested areas and any issues found. The reports will be sent to the quality assurance team, which consists of three members. This will ensure that the team stays informed about the testing progress and can quickly address any critical issues.
      2. To ensure smooth communication and prompt resolution of issues, we will hold regular online or in-person meetings between our group members. This structured feedback loop will help maintain project progress and ensure the system meets quality standards.
6. **Test Schedule**
   1. **Estimate: 2-3 Weeks to complete**
   2. **Involves:**
      1. **Planning and preparation**
      2. **Execution of exploratory and functional tests**
      3. **Performance and security testing**
      4. **User acceptance testing and final report generation**
7. **Control Procedures**
   1. 6.1 Reviews: Regular review of test cases and results to ensure it meets the requirements and objectives.  
      6.2 Bug Review Meetings: Weekly meetings to discuss and prioritize defects and identified bugs, and setting deadlines for identified issues  
      6.3 Change Request: Formal processes will be made depending on the feedbacks of testcases and the implementation will be reviewed to ensure that any new modifications are correctly documented without disruption to the project timeline  
      6.4 Defect Reporting: Testers will report and document any defects/bugs on Jira. It will be used to track and log in based on severity and priority for effective tracking and resolution.
8. **Functions To Be Tested**
   1. **Truck Allocation Algorithm**
   2. **Shipment Entry and Validation**
   3. **Route Optimization**
   4. **Load Capacity and Management**
9. **Resources and Responsibilities**
   1. Resources:
      1. Environment for testing
      2. Data sets for testing
      3. Automation Framework
   2. Responsibilities
      1. Test Planning
      2. Test Execution
      3. Defect Fixing
      4. Review
10. **Deliverables**
    1. **Detailed Test Plan**
    2. **Working Algorithms**
    3. **Test Cases**
    4. **Test Execution logs**
    5. **Defect Reports**
    6. **Final Summary**
11. **Suspension / Exit Criteria**
    1. **Major Issues that prevent further testing.**
12. **Resumption Criteria**
    1. All Critical/High Severity defects resolved.
    2. Test coverage meets predefined criteria.
    3. Retested using different scenarios that validate functionality.
13. **Dependencies**  
    12.1 Personnel Dependencies - Availability of test developers and engineers  
    12.2 Software Dependencies – Availability of required tools and licenses of software  
    12.3 Hardware Dependencies – Workstations and PCs.  
    12.3 Test Data & Database – Access to repositories, both local and git.
14. **Risks**  
    13.1. Schedule – Delays in development that can affect the testing or project timeline  
    13.2. Technical – Integration issues with existing systems.  
    13.3. Management – Resource and task allocation.  
    13.4. Personnel – Availability of the members.  
    13.5 Requirements – Changes in the project/user/software requirements
15. **Tools**
    1. **Jira for defect tracking and project management**
    2. **Selenium for test automation**
    3. **Visual Studio for development and testing**
16. **Documentation**
    1. **All test-related documents will be stored in the project repository.**
    2. **Plans, Cases, Results will be reviewed regularly and documented.**
17. **Approvals**
    1. Test Plan – Kashish
    2. Test Execution – Jhonathan / Thiri
    3. Final Test Report – Aung Moe