**1. Introduction**

**a. Test Plan Objectives**

This project simulates a package delivery system for Seneca Polytechnic, with delivery routes represented as a 25x25 grid. Each delivery truck follows a route (blue, green, or yellow) and can make diversions to deliver shipments.

This test plan aims to verify the correctness of foundational functions provided in mapping.h and mapping.c, including route rendering, pathfinding, and map population. The goal of testing is to confirm that:

* The map and routes are initialized correctly.
* Pathfinding avoids buildings.
* Routes are correctly overlaid.
* Euclidean distance logic functions correctly.

**2. Scope**

**a. In Scope:**

* Map population and visualization
* Blue, green, and yellow route generation
* Route overlay on map
* Shortest pathfinding (without entering buildings)
* Euclidean distance calculation
* Displaying overlapping route symbols

**b. Out of Scope:**

* Shipment-to-truck assignment logic
* Truck volume/weight logic
* User input handling

**3. Test Strategy**

**a. Approach:**

Testing will be primarily **blackbox**, executed manually through main() and console output inspection. All test inputs are hardcoded, and results are compared against expected behavior.

**b. Design Process:**

* Understand functions and structure of the base system.
* Identify critical outputs for validation.
* Create test cases for each function.
* Record outputs manually for now (automated testing to begin in Milestone 3).

**c. Levels of Testing:**

* **Exploratory Testing** for route overlap and map display.
* **Functional Testing** for pathfinding, route logic, and distance.
* **Visual Testing** for printMap() formatting.

**3.1 – 3.9 Test Types (only relevant ones included)**

**3.1 System Test**

Validates map, pathfinding, and routing system-wide.

**3.4 Automated Test**

Not included yet — planned for Milestone 3+ using Visual Studio unit testing.

**3.7 Documentation Test**

Each function’s header comments were reviewed for clarity and correctness.

**4. Environment Requirements**

* **OS**: Windows 10/11
* **IDE**: Visual Studio 2022
* **Language**: C (C11 Standard)
* **Compiler**: Microsoft C/C++ Compiler
* **Testing**: Manual for now; unit testing coming later
* **Test Harness**: Main function (main.c)
* **Hardware**: Standard development PC

**5. Execution Strategy**

**a. Entry/Exit Criteria**

* **Entry**: Code compiles without errors
* **Exit**: All base functions pass at least 95% of functional test cases

**b. Defect Severity Levels**

* **Critical**: Pathfinding enters buildings or map fails to load
* **High**: Incorrect route overlays or symbols
* **Medium**: Wrong path distances or route lengths
* **Low**: Map formatting issues
* **Cosmetic**: Alignment errors in printed grid

**c. Test Reporting**

* Team manually records test results in test case table
* Reports include function tested, expected vs. actual, and pass/fail
* Issues are shared during weekly scrum and added to Jira

**6. Test Schedule**

| **Task** | **Date** |
| --- | --- |
| Complete test plan | [03-07-2025] |
| Execute functional tests | [04-07-2025] |
| Upload test results | [04-07-2025] |
| Discuss results with team | [04-07-2025] |

**7. Control Procedures**

**7.1 Reviews**

Peer review of test cases and map outputs.

**7.2 Bug Review Meetings**

Weekly bug discussion in scrum.

**7.3 Change Request**

Handled in Jira.

**7.4 Defect Reporting**

Logged in Jira with screenshots, severity, and description.

**8. Functions to Be Tested**

* populateMap()
* getBlueRoute(), getGreenRoute(), getYellowRoute()
* addRoute()
* printMap()
* shortestPath()
* distance()
* getPossibleMoves()

**9. Resources and Responsibilities**

**9.1 Resources**

* Visual Studio
* GitHub for version control
* Jira for task and bug tracking

**9.2 Responsibilities**

| **Team Member** | **Responsibility** |
| --- | --- |
| Dhanuth Chathurmika | Testing map and route functions |
| Dineth Damishka | Testing pathfinding |
| Sahan Vimukthi | Writing test documentation |
| Sean Li | Preparing test data |

**10. Deliverables**

* This test plan document
* Screenshots of output
* Test results log

**11. Suspension / Exit Criteria**

* Testing will be paused if more than 3 critical bugs are encountered
* Testing exits when 95% of test cases pass and no critical issues remain

**12. Resumption Criteria**

* Resume once critical bugs are resolved and builds are stable

**13. Dependencies**

**13.1 Personnel Dependencies**

All team members must be available for test execution and results review

**13.2 Software Dependencies**

Visual Studio 2022, GitHub Desktop

**13.3 Hardware Dependencies**

Basic PC with sufficient RAM and CPU for compilation

**13.4 Test Data**

Hardcoded in main.c and route functions

**14. Risks**

**14.1 Schedule**

Delay in writing test cases or setting up test environment

**14.2 Technical**

Missing error-handling in base code

**14.3 Management**

Team roles and responsibilities may not be clearly defined, leading to duplicated work.

**14.4 Personnel**

Delayed onboarding or understanding of project

**14.5 Requirements**

Incomplete understanding of truck logic (planned for MS3)

**15. Tools**

* Visual Studio 2022
* GitHub (source control)
* Jira (issue tracking)
* Microsoft Excel or Word (test documentation)

**16. Documentation**

* This Test Plan
* Scrum Report
* Traceability Matrix (to be added Milestone 3)

**17. Approvals**

| **Name** | **Role** | **Signature** |
| --- | --- | --- |
| Sean Li | Team Leader |  |
| Dineth Damishka | Developer |  |
| Sahan Vimukthi | Tester/Test plan |  |
| Dhanuth Chathurmika | Reflection |  |