**TEST PLAN**

**Group 2**

**1. Introduction**

The project is a system that allocates additional delivery locations to delivery trucks along 3 different routes. The system must allocate the deliveries in an efficient manner while considering the capacity requirements of each truck and monitoring these requirements throughout the program. Essentially, a delivery must be assigned to the tuck that’s route is closest to the delivery location, so long that the truck has capacity to carry the package.

The testing will include various unit testing to ensure that no semantic errors are found within the code. There will also be functional testing so that the business requirements of the application are tested. This can be combined with more in-depth integration testing so that we ensure that the program’s individual units are combined correctly.

**2. Scope**

**a. What Will Be Tested**

* Proper initialization and loading of the 25x25 city grid map.
* Functionality of route assignment for blue, green, and yellow delivery trucks.
* Correct algorithm for finding the shortest path avoiding obstacles.
* System's ability to assign packages to the correct truck based on truck capacity and proximity to delivery location.
* User input validation for package details and destination coordinates.
* Trucks are the same size and can hold up to 2500 kilograms of cargo.
* The trucks are also capable of carrying 100 cubic meters of boxes.

**b. What Will Not Be Tested**

* Interactions beyond the initial allocation (i.e. anything that happens after a package has been assigned to a specific truck based on its nearest route and capacity)

**3. Test Strategy**

**a. Overview of Testing Approach**

* **Unit Testing:** Individual functions like map population, route creation, and the pathfinding algorithm will be tested for correctness.
* **Integration Testing:** Ensuring that the integration of the map system with the route and delivery assignment functions operates as expected.
* **System Testing:** Full-system operation from user input through to delivery assignment will be evaluated.
* **Acceptance Testing:** Test scenarios will be created to mimic real-world operations to ensure the system meets all business requirements.

**b. Test Levels**

* Initial phase to identify any major bugs that could impact subsequent testing phases.
* In-depth tests to verify each specific function of the application, such as input handling, proper error messages, and the logic of route assignment.
* Evaluating the system's performance under various loads.

**4. Environment Requirements**

* **Hardware:** Tests will be conducted on standard development workstations and a dedicated test server for performance assessments.
* **Software:** Requires a development environment capable of running C++ applications, with dependencies on specific libraries for handling input/output operations.

**5. Execution Strategy**

* **Test Execution:** Tests will be performed manually, focusing on the thorough evaluation of all functions.
* **Bug Severity Levels:**
  + **Critical:** Causes system crashes or loss of functionality with no workaround.
  + **High:** Major functionality is impacted with temporary workarounds.
  + **Medium:** Minor functionality issues with acceptable workarounds.
  + **Low :** Minor issues that do not significantly affect system operations.

**c. Test Reporting**

* Regular reports will be generated outlining test progress, coverage, bugs found, and their severities. These reports will be distributed to all project stakeholders.

**6. Test Schedule**

Detailed timelines will be provided, outlining each phase of testing, including start and end dates, to ensure that testing aligns with the project delivery milestones.

**7. Control Procedures**

* **Reviews:** Regular review meetings to assess the progress and quality of testing.
* **Bug Review Meetings:** Scheduled meetings to discuss and prioritize bug fixes.
* **Change Requests:** Procedures for handling requests for changes or enhancements to the system.
* **Defect Reporting:** A formal defect reporting and tracking system will be utilized (i.e.. Jira comments).

**8. Functions To Be Tested**

Detailed in sections 2 and 3.

**9. Resources and Responsibilities**

**a. Resources**

* Testing and development software and hardware.
* Seneca Blackboard.

**b. Responsibilities**

* Assigned QA testers in the team execute tests and report issues.
* Other team members address and come up with solutions to resolve reported issues.
* Team lead to oversee all testing operations and ensure compliance with the test plan.

**10. Deliverables**

* Completed test cases and reports.
* Documented bug reports and fix logs.
* Final test summary report.

**11. Suspension / Exit Criteria**

Testing might be paused if serious problems make it ineffective or impossible to continue. Once these major issues are fixed properly, testing can start again.

**12. Resumption Criteria**

Defined by the solution of the issues that caused the suspension of testing activities.

**13. Dependencies**

* **Personnel Dependencies:** Availability of all team members.
* **Software Dependencies:** Necessary software tools and libraries like Jira, GitHub Enterprise.
* **Hardware Dependencies:** Availability of proper server and workstation.

**14. Risks**

* **Technical:** Complexity in implementing mapping algorithm correctly.
* **Management:** Coordination issues among team members.
* **Requirements:** Misunderstandings or changes in project requirements.

**15. Tools**

* Development tools like Visual Studio.
* Testing tools like VS debugger for unit and integration tests.

**16. Documentation**

**All included in the Git repository for the project.**

**17. Approvals**

Sign-off from all team members to confirm agreement with the test plan.