**Test Plan: Delivery Routing Algorithm**

**Group 4**

**Eric Jo, Huu Linh Nguyen, Hyeri Jang, Joe Peralta, Daniel Dominguez**

**1. Introduction**

a. Test Plan Objectives:

- The objective of this test plan is to validate the functionality and performance of the delivery routing algorithm developed for a local delivery company.

- The project involves optimizing truck selection, route calculation, and package allocation for efficient package delivery within a city grid.

- The testing aims to ensure accurate package allocation, proper handling of delivery constraints, effective route optimization, and error-free operation of the algorithm.

- The expected result of testing is a validated and optimized delivery routing algorithm that meets the company's requirements.

**2. Scope**

a. In-scope:

- Validation of the delivery routing algorithm's functionality, including truck selection, route calculation, and package allocation.

- Testing of package allocation based on weight, size, and destination within the city grid.

- Verification of accurate truck diversion and path calculation for efficient package delivery.

- Handling of different scenarios, such as limited truck capacity, package destination proximity, and obstacles (buildings).

- Error handling and reporting for scenarios where package allocation or delivery is not feasible.

- Testing of algorithm behavior in various load and capacity conditions.

- Assessment of algorithm performance and efficiency.

b. Out of scope:

- Physical truck operations and dispatching: The test plan does not cover the actual operations and dispatching of delivery trucks, as it focuses solely on validating the functionality and performance of the delivery routing algorithm. The plan assumes that trucks are dispatched and operate according to predefined routes.

- Integration with external systems or APIs: The test plan does not include testing the integration of the delivery routing algorithm with external systems or APIs. It focuses on the algorithm's internal functionality and behavior within the defined scope.

It is essential to acknowledge that even though certain aspects are not explicitly included in the test plan, they may still impact the overall performance and functionality of the system. Therefore, during the evaluation process, it is important to consider any potential influences from these excluded elements.

**3. Test Strategy**

a. Approach to Performing Tests:

- The testing approach will include various types of tests to ensure comprehensive coverage and validation of the delivery routing algorithm.

- Test data will be obtained from simulated package delivery scenarios, considering different package characteristics, truck availability, and delivery destinations.

- Different levels of testing, such as exploratory testing and functional testing, will be employed to catch critical defects before proceeding to the next testing phase.

- Test deliverables, roles, and estimated testing duration will be defined to ensure efficient and effective testing.

3.1. System Test:

- Objective: Validate the overall functionality and behavior of the delivery routing algorithm.

- Test scenarios: Test various package characteristics, truck availability, and delivery destinations to verify accurate truck selection, route calculation, and package allocation.

- Test data: Simulated package delivery scenarios with varying weights, sizes, and destinations.

3.2. Performance Test:

- Objective: Assess the algorithm's performance, response time, and scalability under different load conditions.

- Test scenarios: Evaluate the algorithm's performance as the number of packages and trucks increase, ensuring acceptable response times and resource utilization.

- Test data: Large dataset of package delivery scenarios to simulate real-world conditions.

3.3. Security Test:

- Objective: Identify and address potential security vulnerabilities, ensuring the confidentiality, integrity, and availability of sensitive data.

- Test scenarios: Assess resistance to attacks, data breaches, and unauthorized access.

- Test data: Test data with different security scenarios, including malicious inputs and potential vulnerabilities.

3.4. Automated Test:

- Objective: Automate repetitive test scenarios to improve efficiency and accuracy.

- Test scenarios: Identify test cases suitable for automation, such as regression tests, and develop automated scripts using appropriate testing tools or frameworks.

- Test data: Test data required for automated test execution.

3.5. Stress and Volume Test:

- Objective: Evaluate the algorithm's behavior under high load and stress conditions.

- Test scenarios: Simulate a high volume of package requests to assess system stability, performance, and resource usage.

- Test data: Large dataset of package delivery scenarios to stress the system.

3.6. Recovery Test:

- Objective: Verify the algorithm's ability to recover and resume operations in case of failures or disruptions.

- Test scenarios: Introduce failures or disruptions and assess the algorithm's recovery mechanisms.

- Test data: Test data with simulated failure and recovery scenarios.

3.7. Documentation Test:

- Objective: Review and validate the accuracy and completeness of the algorithm's documentation.

- Test scenarios: Review provided documentation for alignment with actual system behavior and features.

- Test data: Algorithm documentation and related reference materials.

3.8. Beta Test:

- Objective: Conduct testing in a real-world environment with selected users.

- Test scenarios: Collaborate with beta users to perform realistic tests, gather feedback, and validate the algorithm's usability and acceptance.

- Test data: User-specific scenarios and real-world test data.

3.9. User Acceptance Test:

- Objective: Validate the algorithm's usability and acceptance by end-users.

- Test scenarios: Collaborate with end-users to define acceptance criteria and conduct tests based on their use cases and workflows.

- Test data: User-specific scenarios and test data.

b. Test Design Process Overview:

- Understanding Requirements: Review and analyze the requirements for the delivery routing algorithm to ensure a clear understanding of expected behavior and functionality.

- Build Traceability Matrix: Create a traceability matrix to map requirements to test cases, ensuring comprehensive coverage.

- Prepare Test Cases: Develop test cases based on defined test scenarios, covering functional, performance, security, and other relevant aspects.

- Review by QA Team: Have test cases reviewed by another member of the quality assurance team to ensure clarity, accuracy, and thoroughness.

**4. Environment Requirements**

1. Hardware:

- Test computers/workstations: Adequate hardware configurations to support the testing activities.

- Network connectivity: Stable internet connection for any network-related testing.

2. Software:

- Operating System: Compatible operating systems for running the testing tools and applications.

- Development Environment: Software development tools required for building and executing the project code.

- Testing Tools: Specialized tools for automated testing, performance testing, and other specific testing needs.

- Simulation Tools: Software or tools that simulate the city map, routes, trucks, and package data for testing purposes.

It is crucial to ensure that the test environment closely resembles the production environment to provide accurate testing results. The test environment setup should be documented, allowing the testing team to recreate the environment whenever necessary.

Additionally, any required test harness or testing tools should be identified and properly set up to facilitate efficient test execution.

**5. Execution Strategy**

a. Entry and Exit Criteria:

- Entry criteria: Availability of the delivery routing algorithm, test environment, and necessary test data.

- Exit criteria: Completion of all planned tests, resolution of critical defects, and achievement of test objectives.

b. Severity Levels:

- Critical: Defects that cause the algorithm to crash or produce incorrect and anomalous results.

- High: Defects that significantly impact the algorithm's functionality, leading to a lack of program functionality.

- Medium: Defects that degrade the quality of the algorithm but may have workarounds to achieve the desired functionality.

- Low: Minor errors or inconsistencies that have minimal impact on the algorithm's functionality.

- Cosmetic: Issues that affect the user interface aesthetics but do not impact functionality.

c. Test Reporting:

i. Reports: Describe the reports that should be produced as a result of testing, their frequency, and recipients.

ii. Bug Reporting: Specify the process for testers to report defects, including communication channels and information required for effective bug resolution.

d. Interaction with Developers:

- Define how the quality assurance team will interact with developers and collaborate on defect resolution.

- Establish communication channels and processes for reporting and tracking defects, assigning developers to fix bugs, and ensuring timely resolutions.

**6. Test Schedule**

|  |  |
| --- | --- |
| **Week** | **Activities** |
| **1** | - Review project requirements and testing objectives  - Identify test scenarios, test cases, and test data requirements  - Define test environment and tools setup.  - Create the test plan and obtain necessary approvals. |
| **2** | - Design and document test cases based on identified scenarios.  - Map test cases to requirements for traceability.  - Review and refine test cases for accuracy and completeness.  - Prepare necessary test data and test environment setup. |
| **3** | - Execute test cases based on the defined test plan.  - Record test results and capture any defects or issues. |
| **4** | - Verify system functionality against expected outcomes.  - Conduct regression testing as needed. |
| **5** | - Iteratively execute test cases and retest resolved defects.  - Log defects and issues identified during test execution. |
| **6** | - Collaborate with the development team to investigate and fix defects.  - Retest resolved defects and update defect status.  - Finalize test execution and ensure all planned tests are completed.  - Compile test results and generate test summary reports.  - Conduct a final review of the testing process and outcomes.  - Share testing reports with stakeholders for review and feedback. |

**7. Control Procedures**

- To ensure the quality and accuracy of test plans, test cases, and documentation, the following review process will be implemented:

7.1. Reviews

- Test plans will be reviewed by a designated team member to verify completeness, adherence to requirements, and effectiveness in achieving testing objectives.

- Test cases will be reviewed by peers to ensure correctness, coverage, and clarity. Reviews will focus on verifying that test cases align with requirements and adequately test the system functionality.

- All testing-related documentation, such as test strategies and test reports will undergo review to confirm accuracy and consistency.

7.2. Bug Review Meetings

- A bug review meeting will be scheduled regularly and will review reported defects. The purpose is to prioritize and assign resources to address the identified issues based on severity and impact.

- The identified defects will be logged and tracked in a bug tracking system, including information on status, assignment, and resolution progress.

- Assigned developers will work on resolving the defects according to their priority and testers will retest the resolved issues.

7.3. Change Request

- Any proposed changes to the testing scope, schedule, or resources should be documented as change requests.

- The requests will be reviewed, and decisions will be made regarding the approval, rejection, or further analysis.

7.4. Defect Reporting

- Testers will report defects using a reporting template.

- The team will conduct regular bug triage meetings to review and prioritize the reported defects based on their severity.

- Assigned developers will investigate, analyze, and fix the reported defects. Testers will verify the fixes by retesting the affected functionality.

**8. Functions To Be Tested**

- List and describe the specific functions or feature of the delivery routing algorithm that will be tested. This may include truck selection, route calculation, package allocation, error handling, diversion calculations, and other relevant functionalities:

- populateMap(): The function that populates the map with the position of all buildings in it.

- getNumRows(): The function that returns the number of rows in the map.

- getNumCols(): The function that returns the number of columns in the map.

- printMap(): The function that prints the map with symbols representing different elements, such as obstacles(buildings), destinations, and routes.

- getBlueRoute(): The function that build and return the route for the blue truck.

- getGreenRoute(): The function that build and return the route for the green truck.

- getYellowRoute(): The function that build and return the route for the yellow truck.

- addRoute(): The function that adds a route to the map, incorporating the specified route into the existing map configuration.

- addPtToRoute(): The function that adds a point to a route.

- addPointToRoute(): The function that adds a point with specified row and column coordinates to a route.

- addPointToRouteIfNot(): The function that adds a point to a route if it is not the same as a specified point.

- distance(): The function that calculates the distance(Euclidian distance) between two points.

- shortestPath(): The function that determines the shortest path/route between two points on the map(does not pass through buildings).

- getPossibleMoves(): The function that retrieves the possible moves or directions from a given point on the map.

- getClosestPoint(): The function that identifies the closest point in a route to a specified point.

- eqPt(): The function that checks if two points are equal.

**9. Resources and Responsibilities**

9.1. Resources

- Personnel: The team should have the necessary skills and expertise in testing methodologies, test design, and test execution.

- Hardware: Sufficient hardware resources to support the testing activities, including computers, servers, and any specialized testing equipment required for specific scenarios.

- Software: Testing tools and environments required for test case management, test execution, test automation, and defect tracking.

- Test Data Sources(Test Cases): Data sources that provide representative and diverse test data to cover various scenarios.

9.2. Responsibilities:

- Test Manager/Lead: Responsible for overall test planning, coordination, and management.

- Test Analyst: Responsible for analyzing requirements, designing test cases, and creating comprehensive test scenarios.

- Test Engineer: Responsible for executing test cases, logging defects, and performing test result analysis.

- Developers: Collaborate closely with testers during defect triage, providing clarifications, fixing reported defects, and ensuring the correctness of the implemented functionality.

**10. Deliverables**

The test plan includes several key deliverables that will be produced during the testing process. These deliverables are as follows:

a. Test Plan Document: A comprehensive document outlining the objectives, scope, approach, and strategies for testing the delivery routing algorithm. It provides a roadmap for the testing activities and serves as a reference for the testing team.

b. Test Scenarios: A collection of test scenarios that cover various package characteristics, truck availability, and delivery destinations. These scenarios ensure that different aspects of the algorithm's functionality are tested.

c. Test Cases: Detailed test cases derived from the test scenarios. Each test case specifies the input data, steps to be executed, and expected results. These test cases provide a systematic approach to validate the algorithm's behavior.

d. Test Data: Simulated package delivery scenarios with varying weights, sizes, and destinations. The test data is used to execute the test cases and validate the algorithm's accuracy and performance.

e. Test Reports: Regular reports generated during the test execution phase to track the progress of testing, record test results, and document any defects or issues encountered. These reports provide visibility into the testing activities and facilitate effective communication with stakeholders.

f. Defect Reports: Reports documenting any defects or issues identified during testing, including detailed information about the defect, steps to reproduce it, and its impact on the algorithm's functionality. These reports help prioritize and address defects in a timely manner.

g. Test Summary Report: A comprehensive report summarizing the overall testing process, including the test coverage, test results, identified defects, and an assessment of the algorithm's performance. The test summary report provides stakeholders with an overview of the testing outcomes and any recommendations for further improvements.

**11. Suspension / Exit Criteria**

The suspension and exit criteria define the conditions under which testing may be suspended or concluded. These criteria help ensure that the testing process remains effective and efficient. The suspension/exit criteria for the delivery routing algorithm testing are as follows:

a. Suspension Criteria:

Critical defects: If critical defects are identified that prevent further testing or severely impact the algorithm's functionality, testing may be suspended until these defects are resolved.

Resource unavailability: If necessary resources, such as test environments or test data, are unavailable for an extended period, testing may be suspended until these resources are made available.

Change in requirements: If there are significant changes in the requirements that affect the test scenarios or test cases, testing may be suspended to reassess and update the testing approach accordingly.

b. Exit Criteria:

Test completion: All planned tests, as defined in the test plan, have been executed and completed.

Defect resolution: Critical and high severity defects have been resolved, and the algorithm's functionality meets the defined acceptance criteria.

Test coverage: The defined test coverage has been achieved, ensuring comprehensive validation of the delivery routing algorithm.

Stakeholder approval: The test results and findings have been reviewed and approved by the relevant stakeholders, indicating their satisfaction with the algorithm's performance and behavior.

**12. Resumption Criteria**

The resumption criteria define the conditions under which testing can be resumed after a suspension. These criteria ensure that the necessary prerequisites are met before continuing with the testing activities. The resumption criteria for the delivery routing algorithm testing are as follows:

a. Defect resolution: Critical defects identified during the suspension period have been resolved and verified.

b. Resource availability: The necessary resources, such as test environments, test data, and personnel, are available to resume testing.

c. Test environment stability: The test environment has been stabilized to ensure consistent and reliable test results.

d. Change assessment: Any changes in requirements or test scenarios that occurred during the suspension period have been assessed and incorporated into the testing approach.

**13. Dependencies**

Dependencies are factors or components that can impact the testing process or its outcomes. It is important to identify and address these dependencies to ensure smooth and effective testing. The dependencies for the delivery routing algorithm testing are as follows:

a. Development progress: The completion of the algorithm's development is a crucial dependency for testing. Testing activities can only commence once the delivery routing algorithm is available for validation.

b. Test environment availability: The availability of the required test environment, including hardware, software, and network connectivity, is essential for executing the tests. Testing should align with the availability of the test environment.

c. Test data availability: The availability of suitable and representative test data, including package characteristics, truck availability, and delivery destinations, is necessary for thorough testing. Test data should be prepared and made available before executing the tests.

d. Documentation availability: The availability of accurate and up-to-date documentation, including requirements, design specifications, and user guides, is important for understanding the expected behavior of the delivery routing algorithm and designing effective test cases.

e. External system integration: If the delivery routing algorithm depends on external systems or APIs for data exchange or functionality, the integration with these systems should be established and validated before testing the algorithm's behavior in the integrated environment.

f. Stakeholder availability: The availability of stakeholders, such as project managers, developers, and business representatives, is crucial for effective communication, requirement clarification, and obtaining timely approvals during the testing process.

g. Resource dependencies: The availability of skilled testing resources, such as testers, automation experts, and performance testing specialists, is necessary for executing the planned tests and ensuring their quality and effectiveness.

**14. Risks**

14.1. Schedule:

- Identify risks related to the testing schedule, such as delays in test execution or resource constraints.

- Develop a detailed test schedule with clear milestones/goals,

allocate sufficient resources for testing, and regularly monitor progress to quickly address any sort of delay.

- Dependencies on external systems or services causing delays in test execution, to mitigate this we identify and communicate dependencies early on, coordinate with team members to ensure timely availability of necessary resources and roles.

14.2. Technical:

- Identify risks related to technical challenges, such as algorithm complexity, performance bottlenecks, or integration issues.

- Conduct thorough and detailed technical reviews, code analysis, and implement proper testing techniques to identity and mitigate technical risks. Monitor system performance during testing and optimize as needed.

- Lack of compatibility with different platforms/operating systems, perform compatibility testing on target platforms/operating systems to identify and address any issues.

14.3. Management:

- Identify risks related to project management, such as unclear requirements, scope creep, or resource allocation.

- Establish effective communication within the stakeholders/company to help clarify requirements and guidelines, regularly review resource allocation and adjust dynamically when necessary.

- Inadequate project tracking and reporting leading to miscommunication and delays, utilize the project management tools and systems to track progress, communicate updates, and generate reports.

14.4. Personnel:

- Identify risks related to personnel availability, skill gaps, or dependencies on specific individuals.

- Identify key personnel and ensure appropriate backup plans are in place. Provide necessary training and support to help equalize the skill level of the team, collaborate to minimize dependencies on specific individuals.

- Key personnel leaving the project mid-way, develop a knowledge transfer plan and ensure proper documentation of critical information. Identify and train backup resources to mitigate the impact of personnel changes.

14.5. Requirements:

- Identify risks related to ambiguous or incomplete requirements that may impact testing.

- Collaborate with the stakeholders/company to clarify document assumptions. Conduct reviews to identify and address any gaps/ambiguities in the provided requirements and documents.

- Changing requirements during the testing phase, establish a change management process to handle requirement changes, prioritize them, and assess their impact on testing. Communicate changes to the testing team.

**15. Tools**

- Specify the testing tools or frameworks that will be utilized during the testing process, such as test management tools, bug tracking systems, automation frameworks, or performance testing tools.

- Select appropriate testing tools based on the requirements of the module, ensure that the tools are properly configured, and provide necessary training to the testing team on tool usage if needed.

- Incompatibility between testing tools and modules, conduct compatibility tests to ensure seamless integration between testing tools and the modules.

**16. Documentation**

- Describe the documentation that will be produced or updated as part of the testing process, such as test plans, test cases, test reports, and user documentation.

- Develop a comprehensive test plan that covers all testing activities, create and maintain well-documented test cases, test reports, and user documentation. Regularly update the documentation to reflect any changes/updates.

- Incomplete or outdated documentation leading to misunderstandings or incorrect test execution, establish a documentation review process to ensure accuracy, completeness, and relevance of all testing documentation.

**17. Approvals**

- Identify the individuals or roles responsible for approving the test plan and associated deliverables before the testing activities commence.

- Clearly define the approval process and identify the relevant stakeholders or roles responsible for reviewing and approving the test plan. Seek their approval before proceeding with the testing activities.

- Delays in obtaining approvals due to lack of clarity or miscommunication, establish clear communication with approvers, provide them with necessary information, and set realistic expectations regarding review and approval timelines. Regularly follow up to ensure timely approvals.