Test Plan Template

1. **Introduction**

* Test Plan Objectives
* The project being tested is a delivery management system designed to optimize the allocation of packages to delivery trucks based on proximity to destinations, available capacity, and efficient routing. This system targets the delivery process, ensuring that packages are efficiently distributed while minimizing diversion from established routes.

1. **Scope**

* **What will be tested:**
* Package: they are assigned to the most suitable size and weight and available space.
* Shorter Routes: ensure it accurately calculates the shortest path from trucks' current positions to destinations while considering obstacles such as buildings between.
* Capacity Management: ensure that packages are allocated to trucks with sufficient space and weight capacity and prevents overloading.
* Error Handling: where a truck cannot reach a destination, or all trucks reach capacity. This includes noticing meaningful error messages and taking appropriate actions to resolve issues.
* **What will not be test?**
* Physical Delivery: loading packages onto trucks and delivering them to destinations, will not be tested.
* External Factors: weather conditions, traffic congestion, or road closures will not be tested.
* Real Data Information: package tracking information will not be tested.

1. **Test Strategy**
   * Our testing strategy evaluation of system quality and functionality. It includes tests to uncover critical defects and functional tests to validate key application functions. Documentation testing ensures that the software implementation aligns with specified requirements. User acceptance testing and system testing are conducted to ensure that user expectations are met and to verify overall system functionality. tests such as security, stress and volume, and recovery testing may not be unnecessary at this point. Overall, our strategy aims to deliver the best solution that meets requirements and provides high-quality experience.

* **System Test**
  + - Verify package allocation based on starting point to destinations and available truck capacity.
    - Package is suitable size and weight and available space of truck.
    - Ensure proper management of truck capacity to avoid overloading.
    - Test routing optimization algorithm for accurate calculation of shortest paths.
    - Validate error handling for scenarios where trucks cannot reach destinations or reach capacity.
  + **Automated Test**
    - Develop automated test scripts to validate key functionalities such as package allocation, routing, and error handling.
    - Execute automated tests to ensure efficient regression testing.
  + **Documentation Test**
    - Review system documentation including user manuals, technical specifications, and API documentation.
    - Ensure documentation is accurate, comprehensive, and up to date.
  + **User Acceptance Test**
    - Validate that the system meets user requirements and expectations.
    - Ensure usability, functionality, and performance align with user needs.
    - You could describe the test design process and give an overview of how it will be conducted. You could provide a broad overview of
  + **Test Design Process:**
    - Understanding Requirements:
      * review project requirements and specifications to clear understanding of expected system behavior and functionalities.
    - Building Traceability Matrix:
      * Create a traceability matrix to map requirements to test cases, ensuring comprehensive test coverage.
    - Test Case Preparation:
      * Test cases will include steps to verify package allocation, routing, capacity management, error handling,
    - Test Case Review:
      * Review test cases by group meeting to ensure accuracy and relevance.
      * feedback and make it necessary to improve test case quality.

1. **Environment Requirements**

Accurately assessing the performance and reliability of the software depends on the testing environment. The necessity of defining the hardware and software requirements is referred to all over the text. The following details need to be included:

Software Framework:

**Operating System**: Indicate which Windows, Linux, or macOS versions are needed for compatibility testing as few outdated versions are not supported by the testing softwares.

**Tools for Development**: List the GCC compiler, Visual Studio versions, and any additional development or testing tools that are required like visual studio debugging tools, log files and assertions which can help us to test every part of code.

**Version Control**: Verify that Git is being used and indicate the branching plan and repository configuration for testing.

**Hardware Setting**: Processor, RAM, and Storage: Establish minimum (which should be about 8 gb of ram and 512 gb of rom) and required values to ensure the testing setup can accommodate the application requirements.

**Network**: high speed internet connection for fluent testing, appropriate bandwidth along with security configuration for better security.

**Test Harness:**

Detail any specific test harnesses that will be built or pre-existing testing tools used, and continuous integration tools and visual studio unit testing should be set up.

1. **Execution Strategy**

a. The execution of the tests, including the entry and exit criteria, are described in this section. Once every test case has been established, examined, and the system is stable enough for testing, the tests will start.

The test type will determine the exit criteria. For instance, a perfect score is necessary for vital functionality tests like for, which are crucial to the operation of the system and should pass 100% of the test scripts. Less important tests may have exit criteria of passing 95% of test scripts, provided that failing tests do not materially affect overall system integrity or user experience. As soon as there are no more serious or critical flaws, a test is deemed finished. This helps guarantee that, even though user safety and overall system functionality are not compromised, a certain level of useful flexibility has been added where appropriate, while perfection is looked after on key system components.

b. In this section, you can characterize the flaws' severity and categorize them into severity categories of:

**Critical**: Flaws that result in a system crash or give false directions. These flaws basically stop the application from carrying out its essential tasks, so they need to be fixed right away.

**High**: Flaws that significantly impair system performance and might prevent users from finishing activities. Although there can be short-term solutions, these problems have a big impact on system dependability and user productivity.

**Medium**: Bugs that require a solution but impair system performance or quality. They won't stop the essential functions, but they will make users less productive or have a worse experience.

**Low**: Small errors that have minor impact on the system, including unclear error messages. They usually do not cause any disruptions and can be easily fixed by users.

**Cosmetic**: Bugs that influence a user interface's look but have no effect on the system itself.

While not directly related to operational capacity, these flaws may worsen the user experience.

1. **Test Schedule**

**1. The first step in test planning is to understand the project requirements and objectives. This includes reviewing the project scope, functional and non-functional requirements, and any constraints or limitations.**

**2. The second step is to Define the Test Objectives: The test objectives should align with the project objectives and should clearly define what the testing is intended to achieve. For example, the test objectives for a delivery management system might include verifying package allocation, routing, capacity management, and error handling.**

**3. In this we get to know about different stages of testing such as unit testing, integration testing, system testing, and acceptance testing.**

**4. Another is to define the test approach. The test approach should define the testing techniques and methods to be used, such as black box testing, white box testing, or grey box testing. It should also define the test environment, test data, and test tools required for each test level.**

1. **Control Procedures**

**1.Reviews:** Code reviews should be conducted after every major stage of the project. The review should be conducted by the developers and testers and need access to test data and relevant information regarding any bugs discovered. The review should encapsulate program functionality, avoiding redundancies, overall efficiency, and robustness of the project

**2. Bug Review Meetings:** Bug review meetings should be conducted weekly and with a thorough analysis. The tester needs to present the test data that was used, along with the data selection which caused the bug to occur. Bug meetings should be conducted with potential solutions discussed and demonstration of the fix in the code.

**3. Change Request:** A change request should be presented during the early development of a stage in the project, and not during the end part of the stage that is close to the launch. This is to avoid delays and to provide enough time for the change to get implemented after being approved by the rest of the team. To accept a change request, the developer must demonstrate that their change is accomplishing an improvement in the software, without introducing new bugs or redundancies. The change will be approved by the rest of the development team if it satisfies functionality criteria.

**4. Defect Reporting:** Defects should be reported immediately to the lead developer. defects include bugs that seriously impede the functionality of the program and can potentially cause fatal errors and/or crashes. Development needs to be suspended so that the defect can be addressed immediately. Create a backup of the project before fixing the defect.

1. **Functions To Be Tested**

1)Package Allocation Ensure packages are assigned based on suitable size, weight, and available space. Importance: Critical for efficient distribution, preventing overloading and ensuring customer orders are fulfilled.

2) Proper management of truck capacity to avoid overloading.

3) Capacity Management: Allocate packages to trucks with sufficient space and weight capacity, preventing overloading.

4) Error Handling: Handle scenarios where a truck cannot reach a destination or all trucks reach capacity, providing meaningful error messages.

1. **Resources and Responsibilities**

C Language Developers:

Task: Craft functional C language code.

Outcome: Code must meet the outlined specifications.

Software Testing Personnel:

Task: Execute thorough debugging and ensure quality.

Outcome: Deliver test modules that exhibit extensive coverage.

Data Analysis Experts:

Task: Examine app data concerning package distribution and routing.

Outcome: Provide analytical findings for the scrum report.

Customer Relations Division:

Task: Manage interactions regarding customer order tracking and completion.

Outcome: Input towards the collective agreement.

Quality Assurance Team:

Task: Create and uphold a detailed testing framework.

Outcome: Deliver exhaustive testing records.

1. **Deliverables**

**a. Functional code in c language in files data.h , main.c, mapping.c, mapping.h**

**b. Testing Modules**

**c. scrum report**

**d. Group contract**

**e. Testing template with detailed information about 17 subtiles**

1. **Suspension / Exit Criteria**

1. It's really important to make sure that the system can assign packages to trucks considering space availability, distance and any necessary detours. If it messes up it could result in orders not being fulfilled and customers getting upset. In situations quick action needs to be taken to fix things.

2. The systems capability to figure out the route and avoid any navigation errors is crucial, for ensuring package deliveries. Any issues with this could cause delays and extra expenses. That's why it's so important to deal with any problems

3. If the system can't handle scenarios where trucks can't reach their destination due to obstacles it may lead to delays and missed deliveries. This could harm the companys reputation. Cause losses. So making sure the system can manage situations properly is essential.

4. Ensuring printing of information is key, for getting packages to the right place on time. If the system prints details it might cause confusion, delays and extra expenses. Therefore correcting any inaccuracies promptly is crucial.

5. The designated weight restriction of 1000 kg is enforced to protect the well being of drivers and fellow road travelers. Allowing packages exceeding this limit to be loaded onto trucks by the algorithm may result in accidents and legal consequences. Hence it is crucial to deactivate the algorithm under circumstances.

1. **Resumption Criteria**

1.Cases of testing suspension: Testing suspension may occur in the case of fatal error in the project, memory and/or data leaks, or suspension of program due to critical debugging stage.

2. Resumption Criteria: Testing can resume when project error has been identified, data leaks have been addressed. Testing can resume when the project has been properly suspended from servers and safe back-up and GIT branches have been created. Testing can resume once integration testing framework has been created and test data selection has been completed.

1. **Dependencies**

1.Personnel dependencies: All personnel must have the required software and hardware to run the project. Personnel must consist of Developers using C language, Software Testers to perform debugging and quality assurance, Data Analysts for the application data, and a customer relations unit to interact with customer order tracking and

fulfilment.

2. Software dependencies: All development personnel must have Visual Studio IDE and access to project files to run the project, git version control for development and on project. Data Base software is Oracle SQL. All personnel must have Windows 11 operating system.

3. Hardware dependencies: x86 architecture, minimum intel i5 processor, 8 gb of ram on all systems. 512 gb storage for database.

4. Test Data and Data base: Test data must consist of black box and white box tests which demonstrate robust functioning of the application and all cases accounted for. Data base: Oracle SQL relational database with C++ application.

1. **Risks**

In the field of software testing it's important to recognize and deal with risks that could affect how well the process works. These risks can be categorized into different types, including schedule, technical, management, personnel, and requirements risks.

When it comes to schedule risks these are, about facing difficulties in meeting testing deadlines. This can happen due to reasons like delays in development changes in scope, limited resources, availability of test data, reliance on factors and not enough time allocated. To handle these risks well it's key to plan projects communicate clearly with stakeholders prioritize effectively and manage risks proactively.

Technical risks can impact the quality of testing. Might come from issues with infrastructure like network problems and hardware limitations. They could also stem from challenges in setting up test environments limitations of tools used for testing and concerns about managing data. Taking care of these risks involves making sure everything is compatible setting up environments correctly and managing data well. Working closely with experts and keeping up to date on tools and security measures can also help manage these risks and enhance testing efficiency.

Management risks are all about problems in project management and coordination. It's crucial to understand how poor project management could affect the progress of a project and take steps to address these risks effectively.

Personal risks involve challenges within a team such as turnover rates, lack of motivation, communication issues, skill gaps, among team members and personal conflicts. In order to reduce these risks it's crucial to nurture a team atmosphere promote open dialogue offer assistance and promptly address any conflicts that arise. Holding meetings and providing support can help keep the testing team unified and motivated fostering personal development and job satisfaction.

Requirements risks pertain to challenges stemming from incomplete or evolving project requirements that could significantly affect the projects scope and outcomes. Properly evaluating the implications of these requirement issues and taking measures to address them is vital, for ensuring the success of the testing process.

1. **Tools**

1. Visual Studio 2022 community edition

2. C programming language. GCC Compiler.

3. Git Version Control

4. Oracle SQL Database

1. **Documentation**

In the field of software testing numerous documents are created to guarantee the excellence and dependability of the software under examination. These documents have purposes. Offer essential details that assist in the process of testing software. Some examples of documents include test cases, test scripts, test reports, defect logs and user manuals. It's important to recognize that the style and organization of these documents can vary based on the type of software being tested. Furthermore, to maintain consistency and standardization specific templates or guidelines may be provided for reference.

1. **Approvals**

In order to ensure a smooth and successful testing phase, it is crucial to identify the responsible individuals or stakeholders who will be approving the test plan and associated documents. This involves defining a process and timeline for obtaining their approval, ensuring that all necessary parties have reviewed and given consent before commencing the testing activities. Such a comprehensive approach to stakeholder identification and approval can help mitigate risks and promote accountability within the testing process.