LO2 support document (Test Plan)

Test Plan Document

In a drone delivery system, the most important part is whether the drone can deliver the food to the designated place as required and return it safely. Here we might consider only three requirements related above but they are good enough to illustrate. **R1:** The first is the validation of the drone route; **R2:** the second is test if the drone could go back to charging place when its battery is not enough to run all the orders; **R3:** the third is the drone's load testing.

- **R1:** To ensure that the drone delivery system's route validation functionality is working correctly and the drone can safely navigate to its destination in real scene. (Make it valid in real life)
- **R2:** The drone should check the battery after every move to ensure there are enough battery to go back to charging place if the remaining battery cannot delivery all the orders.
- **R3:** To ensure that the drone delivery system can handle a high volume of requests and maintain its performance under heavy load.

Priority and Pre-requisites

R1: A high level of quality for validation of drone route is necessary since:

- ✧ **Safety**: The validation of drone routes is crucial for ensuring the safety of the drone, as well as the safety of people and property in the vicinity of the drone's flight path. A high level of quality in route validation ensures that the drone is able to safely navigate to its destination, avoid obstacles, and handle emergency situations.
- ✧ **Compliance**: Drone delivery systems need to comply with various regulations and safety standards set by authorities. A high level of quality in route validation ensures that the system complies with these regulations and standards, which in turn ensures that the drone is operating within legal and safe parameters.
- ✧ **Reliability**: A high level of quality in route validation ensures that the drone is able to reach its destination reliably and efficiently. This is important for ensuring that the drone can deliver packages on time and in good condition, which is critical for maintaining customer satisfaction and trust.

- If the demand has a high priority, according to the principles of Chapter 3 we should consider at least two potential T&A procedures.
  1. Sensitivity (better to fail every time than sometimes): Sensitivity refers to the ability of the system to detect and respond to changes in its environment or inputs. For example, when testing a drone delivery system, sensitivity analysis would involve testing the system's ability to detect and respond to changes in wind speed, temperature, and other weather conditions that could affect the drone's flight. This would help to ensure that the system is able to operate safely and effectively in a variety of different conditions, and that it can adapt to changes in the environment in real-time.
  2. Partition (Divided and Conquer): This approach involves testing the system's ability to handle different types of packages, such as different weight and size. This can help to ensure that the system is able to handle different types of packages and navigate safely to its destination. For example, a drone route might be divided into several legs, and each leg would be tested individually.
- **Identifying the inputs and outputs of the drone route planning algorithm:**
  - **Inputs**:
    - Order information: It includes restaurants information (each order may contain products from multiple restaurants), customer information (delivery address)
    - No-fly zones: A geojson file that records the boundary points of the no-fly zone in the form of points.
    - Confinement Area: A geojson file that records the boundary points of the confinement area in the form of points.
  - **Outputs**:
    - The drone route is valid or not.
  - **Specification**: The drone route is valid when the route does not cross the no-fly zones and the confinement area.
- The principles of CH03 of Y&P suggest we decompose the requirement, and it can be divided into two tasks based on theory and reality.
  - **Task 1**: The first part tests the integrity and validation of the drone route in a drone delivery software according to the restrictions of the no-fly zones and confinement areas.
  - **Task 2**: The second part of the test is based on the calculated valid drone route. During the actual food delivery process, whether the drone can ensure that the food delivery route is valid and safe.
- In task 1, we need some specific code (**functional testing**) to check whether the drone route through the no-fly zone or confinement area can be captured (invalid). After that, **Combinatorial testing** would be used to validate the interactions between different factors that affect the drone route, such as the no-fly zones, confinement area, correct delivery order, etc. **Model-based testing** will be used to generate as many test cases as possible, by using mathematical models to generate test cases and verify the output of the algorithm.
- In task 2, functional testing is needed to ensure whether the drone can follow the calculated route within the specified range error during the actual delivery process. **Simulation testing** involves creating a virtual environment that closely replicates the real-world scenario in which the drone will operate. In this scenario, the drone route validation can be tested by simulating different conditions such as weather, obstacles, and delivery location. The simulation environment can be used to verify that the drone is able to find a valid route, avoid obstacles, and reach the destination safely. **Real-world testing** involves deploying the drone in an actual environment and testing its ability to find a valid route and reach the destination. This type of testing is essential in validating the performance of the drone under real-world conditions. It is important to conduct real-world testing in a controlled environment, such as a closed testing area or a mock-up delivery scenario, before deploying the drone in an actual delivery scenario. By combining simulation and real-world testing, we can ensure that the drone route validation has been thoroughly tested and is functioning correctly in both virtual and real-world environments. Additionally, it is important to perform ongoing testing and monitoring to ensure that the drone continues to function as expected and to identify and resolve any issues that may arise.

R2: A high level of quality for validation of drone route is necessary since:

- ✧ **Safety**: Ensuring that the drone has enough battery to return to the charging place is crucial for the safety of the drone and those around it. If the drone runs out of battery while in flight, it could crash, causing damage to property or injury to people.

- **Reliability**: Checking the battery after every move helps ensure that the drone can complete its deliveries as intended. This level of reliability is important for building trust in the drone delivery system and maintaining customer satisfaction.
- **Cost**: Avoiding battery-related issues helps reduce the costs associated with repairing or replacing damaged drones, or rescheduling deliveries that could not be completed.
- **Efficiency**: By checking the battery after every move, the drone can avoid wasting time and energy by continuing to make deliveries when it does not have enough battery to return to the charging place. This helps optimize the drone's operations and increase overall efficiency.

- If the demand has a high priority, according to the principles of Chapter 3 we should consider at least two potential T&A procedures.
  1. Feedback: This approach involves collecting data from the drone during its flight, such as battery level, and using it to adjust the route. For example, the drone could collect data on the battery level after each move and use this information to decide if it needs to return to the charging place or continue with its deliveries.
  2. Restriction: This approach involves limiting the drone's movements to conserve battery power. For example, the drone might be restricted to flying at a lower altitude or avoiding certain areas that are more battery intensive.
- **Identifying the inputs and outputs of R2**
  - **Inputs:**
    - Remaining battery: The remaining battery will be checked after each move.
    - Battery Limitation: Defines the minimum battery power required before the drone returns in the current scenario (fixed no-fly zone and confinement area)
    - No-Fly zones and confinement area.
  - **Outputs:**
    - A route that can support the drone return to charging place before the power is exhausted.
  - Specification: In calculating the return route of the drone, we don't need to consider whether the drone will carry heavy objects during the return flight, but in actual situations, we need to consider that the battery power consumption speed will be proportional to the weight of the items currently carried by the drone.
- Similarly, we can divide the requirement test into **two tasks** as in R1. The first is to test the return algorithm, and the second is to test it in real scenarios.
  1. **Task 1: Functional testing** would be used to verify that the drone's battery checking mechanism is working correctly and that it accurately determines the remaining battery level after each move. This type of testing would ensure that the drone is able to make the correct decision about whether to continue with the delivery or return to the charging place. **Model-based testing** would be used to verify that the algorithm for calculating the remaining battery level and determining whether to return to the charging place is correct. This type of testing would leverage mathematical models to generate test cases and validate the outputs of the algorithm.
  2. **Task2: Simulation testing** would be used to test the drone's battery management in different scenarios, such as long-distance deliveries, the drone carried with heavy food that will increase the consumption of the drone battery, deliveries in areas with strong wind, and deliveries in areas with challenging weather conditions. This type of testing would help to validate the performance of the drone under different conditions and to uncover any issues that may arise.

R3: This is a lower priority requirement so we will devote less effort to this requirement. Since this requirement has to be tested in a real scene, so we need the means to measure this:

- First use the **functional test** to test the maximum weight limit of the drone. Then use **combinatorial testing** to test for different types of loads.
- The test environment should be set up to closely mimic the production environment and should include the necessary hardware and software configurations.
- The test environment should be populated with a large volume of data to simulate a heavy load scenario.
- **Test approach:**
  - **Load Testing**: This type of testing involves generating a high volume of requests to the system and measuring its response time and throughput.
  - **Stress Testing**: This type of testing involves increasing the load on the system beyond normal operating conditions to identify performance bottlenecks and to determine the system's maximum capacity.
  - **Endurance Testing**: This type of testing involves running the system under heavy load for an extended period to identify any performance degradation over time.
- The result of the load test will affect the efficiency of drone delivery, and the result will lead to changes in the delivery order and delivery efficiency. This requires integration testing with drone performance testing to ensure that the drone can work properly in real scenarios. The results will also lead to whether it is necessary to consider updating the hardware of the drone to ensure its work efficiency.
- **Test cases examples:**
  - Response time of the system under heavy load
  - Throughput of the system under heavy load
  - Drone hardware stress testing under heavy load
  - Drone performance under heavy load

Scaffolding and Instrumentation

Here we describe what scaffolding and implementation are needed in order to carry out the given tasks (and this may give result in more tasks to build scaffolding and instrumentation). For our requirements:

- **Since R1 and R2 have similar functionality to test, we will discuss them together:**
  - Instrumentation:
    - Logging: Adding log statements to the code to record important events and provide information about the drone's behavior and performance.

        - Metrics collection: Adding code to measure key performance indicators, such as the battery level, the drone's position and altitude, and the routing algorithm's performance.
        - Debugging tools: Tools, such as debuggers, that allow developers to inspect the state of the application at runtime and identify issues.
    - Scaffolding:
        - Java frameworks, such as JUnit or TestNG, that provide an infrastructure for writing and executing tests.
        - Mocking frameworks: Java libraries, such as Mockito or EasyMock, that allow developers to create mock objects that simulate the behavior of other components in the system. This allows developers to isolate and test individual components in isolation.
        - For system testing: Test automation frameworks: Java frameworks, such as Selenium or Appium, that allow developers to automate the testing process.
        - Performance testing: Java libraries, such as Jmeter or Gatling, that allow developers to test the performance of the system under different conditions and load levels.
- R3:
    - Instrumentation:
        - **Performance monitoring tools**: Tools can help monitor the performance of the drone delivery system, including response time, CPU usage, and memory utilization.
        - **Logging**: Adding log statements to the code to record important events and provide information about the drone delivery system's behavior and performance. This information can be used to identify performance bottlenecks and improve the system's scalability.
        - **Tracing**: Adding code to track the flow of execution through the drone delivery system and measure the performance of key components, such as the routing algorithm, under heavy load conditions.
    - Scaffolding:
        - Load testing tools: Tools, such as Jmeter or Gatling, that allow developers to simulate large amounts of traffic or requests to the drone delivery system and measure its performance under load.
        - Real scenario tools: real food package and other weights.

Process and Risk

- **R1 and R2**
    - **Process:**

1. R1 and R2 should be placed in the verify stage. In DevOps lifecycle, the verify stage refers to Verify the functionality and performance of the drone delivery system, including the drone route validation, by performing various tests such as integration tests, system tests, and performance tests.
2. During the verify stage, the drone route validation can be thoroughly tested to ensure that it meets the requirements and functions as expected. Any issues found during this stage can be addressed and resolved before deployment to production. This stage also helps ensure that the drone delivery system is reliable, scalable, and secure.

    - **Risk:**

1. Late detection of issues: If there are any issues with the drone route validation, they may not be detected until the verify stage, which could result in significant delays and costs in fixing the problems.
2. Increased testing effort: The drone route validation will need to be thoroughly tested in the verify stage, which will require a significant effort from the testing team. This could result in increased testing time and costs.

- **R3:**
    - **Process:**

1. The task "loading test" should be placed in the "verify" stage. This is because the "verify" stage involves testing and evaluating the application or system to ensure it meets specified requirements and operates as intended. The "loading test" would be a part of this evaluation, specifically testing the drone's ability to load packages and determine if it can handle the weight and other factors involved in the delivery process.

    - **Risk:**

1. It has potential risk of discovering issues with the drone's loading capabilities later in the development process. This can result in delays, increased costs, and the need for rework if the loading issues are not addressed early on. Additionally, if the loading issues are severe enough, they could impact the overall functionality of the drone delivery system, resulting in a lower quality product that may not meet customer expectations. It's important to identify and address potential issues as early as possible in the development process to minimize the risk of negative impacts on the project.