Requirements:

R1: the drone completes most of the deliveries in a day and has a fast delivery time,

R2: the system performs efficiently (the code should run fast and under 60 seconds)

R3: the drone should be able to handle deviations and obstacles( mapping the flightpath is data-driven)

R4: the drone should have accurate navigation and the flight is safe for the public

R5: the code is easy to read, maintain and upgrade.

Lifecycle chosen : Extreme programming (XP)

Reasons for not choosing others:

Cleanroom: clean room focusses on mathematical modelling and the reduction of errors. While the reduction of error is good, this project requires a more flexible and adaptive approach

SRET: while this lifecycle emphasizes detail and accuracy, which is what is needed for the flightpath and navigation a more flexible and customer centric approach was needed for the software

Dev ops: stated in the original document.

Priority and pre-requisites:

R1: This is a high priority requirement so we will have to devote more effort for this requirement. It is also a measurable requirement ,hence we require the means to measure it:

- This a system level requirement so it will occur late. In the XP lifecycle, this would occur before the acceptance test.
- The principles of Chapter 3 indicate we should consider at least two different A&T approaches if the requirement has high priority.
- The ability to complete most deliveries in a day and have a fast delivery time depends on the heuristic and travelling salesman approach chosen.
- The restriction principle states that we should reduce hard problems to simpler problems. In this case, it would be easier to test the performance of delivering just 1 or 2 orders and comparing the average time compared to running it for a whole day or month.
- The feedback principle states that we would have to learn from experiences, hence if customers complain that deliveries are taking too long or not all deliveries can be completed then I would have to change the heuristics and approach and create new acceptance tests to release a better version.
- To verify this requirement we would need some synthetic data and to validate it, would require the checking of the performance
- This involves:
    - Creating synthetic data for that represent restaurants and the expected orders per day
    - Adding performance logging code that will be able to track the performance and give it statistics.
    - Feeding the collected data back into the unit tests stage

R2: This is a low priority requirement that is requires system level testing but can be tested early on:

- Although the performance can only be tested at a system level, testing the performance of other components of the program early on would give a more favourable result for this test
- When at the unit test stage of the lifecycle, the performance of the other components should also be tested as they would affect the main performance of the program. If the helper functions are optimised, the program in general would be optimised hence would perform better and under the threshold (60 seconds).

R3: This is an integration level requirement that has a high priority. Testing for this would take place at the unit test and acceptance test stage:

- The unit test stage would test if the flightpath plotted changes according to the latitude and longitude of the locations given. This would involve creating some synthetic data and scaffolding for the mock locations.
- At acceptance stage, the test would check if the program is able to communicate with the database and the webserver to get the orders and the locations. If it is able to receive those then it can be guaranteed that the flightpath is able to handle deviations in locations since we already know that the flightpath that will be plotted is data driven as shown by the unit tests.

R4: for this requirement, it would involve testing at the unit level and the system level and it is of high priority since it has safety requirements. So:

- At the testing stage, specifically at the unit testing stage, some tasks would need to be scheduled first:
  - Generate synthetic data to test the software on
  - The synthetic data should consists of mock orders, restaurants, pick up locations and no-fly zones
  - Design a logging system to capture the statistics relating to the times the drone's flightpath failed to cross a certain location, the times it got lost or stuck, the number of times the battery ran out before returning to Appleton and the number of times it crossed into a no-fly zone.
- Based on the propose flightpath algorithm and data received from testing the algorithm, the data may need to be fed back into the testing stage to change the approach(if it fails too often)
- The requirement would require the most amount of resources as it is the main component of the entire project (planning an optimized and good flightpath) and because of the safety regulations
- Although most of these tests would occur at a system level, some can be done at the unit test level such as ensuring a movement by the drone does not cross a no fly zone, ensuring the drone is at most at a length of length 0.00015 degrees from a pick up or drop location where it hovers and making sure the drone always stops delivering when on low battery and returns to apppleton to avoid falling from the sky.

R5: this is not a measurable quality but would be done at the pair programming stage.

- The feedback a&t principle would apply here where the developer would have to learn from experience which in this case would be feedback from a peer during the pair programming

- Readability , maintainability and upgradability of code cannot be measured and instead can only be judged hence there are not many tests that can be done.
- During the peer programming stage, the developer would have to take feedback from a peer and feed it back to the generate user stories stage, this would allow the programmer to rewrite and refactor the code so that it is easier to read, maintain and upgrade.