LO1 - Analyze requirements to determine appropriate testing strategies

This document outlines the requirements for the software and the how some of them will be carried out during the testing stage.

Functional Requirements:

Validation requirements

- Credit card must be exactly 16 digits long, and
- Credit card must be a valid Visa/MasterCard number.
- CVV must be 3 digits long.
- Credit card expiry date must be after the date of the order.
- The total cost of the order must be equal to the total sum of all the pizzas together and the delivery charge.
- The order must contain at least 1 pizza
- The order mustn't contain more than 4 pizzas.
- Pizzas must come a singular same restaurant.

Application

- Produces a file with a list of coordinates for the flight path of the drone in GeoJSON format.
- Produces a file that contains the records in JSON format for all orders on any given date.
    - Each of these records must contain...
    - A unique order number.
    - The outcome of the order.
    - The total cost of the order.
- Produces a file that records containing every move the drone makes on a date in JSON format.
- Every order should have an outcome (Delivered, No Delivered, Invalid) along with a code that states the reason.

Drone Operation and data

- Drones must fly an optimal and valid path specified by the flight path algorithm.
- Drones must avoid no-fly zones.
- Battery permitted, the drone is to return as close as Appleton Tower
- The drone starts from Appleton tower every day.
- Drones cannot leave the central area it cannot leave until it has delivered a pizza.

- The drone must land within no more than 0.00015 degrees.

Data required from REST server.

- The central zone.
- No-fly zones.
- A list of orders for any given date.
- An array of participating restaurants.

Non-Functional Requirements:

- The program will continue if non-essential data is missing.
- The program must terminate if essential data is missing.
- The application should have a run time less than 60 seconds.
- The number of orders completed every day should be maximised(efficiency).
- The application should be designed so it can be extended easily.

In the following section, we will look at the level of requirements across the project and justify the testing approaches used for each one.

Unit-Level Requirements:

Unit testing involves us testing the individual functionalities of the software. This would include but not limited to, order date checks, credit card validation and expiry dates, and pizza/restaurant ordering rules. A key principle of unit testing is that these functions will be tested in insolation of each other to guarantee that each function operates exactly as intended.

System-Level Requirements:

This covers the application's main functions, for example, maximizing the efficiency of the pizza delivery and keeping the runtime to within 60 seconds. This will be evaluated by system and performance testing to ensure that the application runs efficiently across the whole codebase.

Integration-Level Requirements:

This project has requirements that insist on inter-component interaction, such as interacting with the REST server for data retrieval and ensuring that drone delivery operates within the parameters set.

Non-Functional Requirements:

Lastly, we will test the non-functional requirements (as they are the least important when testing). This includes testing the extensibility, robustness, and performance of our entire codebase. This will be achieved by performance and resilience testing, as well as code reviews.