# LO1

## 1. Analysis of Requirements

I will perform an analysis on the client's requirements, and the level of their requirements in order to identify and justify a certain test approach.

### My Task:

I have been approached by the School of Informatics to implement the REST-based service part of their new "PizzaDronz" delivery service, where pizzas will be delivered by drone to the top of Appleton tower.

### Stakeholder's Requests:

- The School of Informatics has requested that I ensure only valid orders are processed.
- They have insisted that all information which the system needs is provided by a centralized REST-service, consisting of no-fly zones, restaurants and the co-ordinates of the University's Central Area.
- They have given me a verifiable specification that the runtime of the flight-path calculation should be less than 60 seconds, as longer runtimes would delay the launch of the drone.

The School of Informatics has provided detailed specifications of the requirements for the service (ILP_2024.pdf), as well as 2 technical requirements which contain the required endpoints (ILP CW1 Spec.pdf and ILP CW2 Spec.pdf).

## 2. Requirements

The requirements for the PizzaDronz application can be split into different sections:

- Functional requirements
- Non-functional requirements

## i. Functional Requirements

1. **F1: The flight path calculated by the application must be valid according to the following:**
   - The drone should not enter any of the no-fly zones [Unit Level].
   - Once the drone has entered the Central Area it cannot leave it again until it has delivered the pizzas to Appleton Tower [Unit Level].
   - Flight paths should only be calculated for valid orders [Integration Level].
2. **F2: The application should validate orders with the criteria as defined on pg. 17-18 in the ILP Spec [Unit Level].**
3. **F3: The application must be reachable at the following endpoints [System Level]:**
   - `http://localhost:8080/validateOrder`
   - `http://localhost:8080/calcDeliveryPath`
   - `http://localhost:8080/calcDeliveryPathAsGeoJson`
4. **F4: The application must use the rest server (https://ilp-rest-2024.azurewebsites.net/) to retrieve no-fly zones, restaurant details, and the Central Area [Integration Level].**
5. **F5: The application should return the flight path calculation in GeoJson format [Unit Level].**

## ii. Non-Functional Requirements

The non-functional requirements can be split up into measurable quality attributes and qualitative requirements.

- **Measurable Quality Attributes:**
  - **N1: Performance** – The flight path calculation must complete within 60 seconds, as specified by the client [System Level].
- **Qualitative Requirements:**
  - **N2: Maintainability** – The code should be modular, with good structures, commenting, and documentation so that future changes can be made efficiently [System Level].

- **N3: Robustness** – The code should be designed to handle errors in a way that doesn't disrupt the overall operation or cause a crash [System Level].

## 3. Test Approach for My Requirements

### F1: Flight Path

- **No-fly zones:** I will use unit tests to simulate various no-fly zones. I will check that the calculated path circumvents various no-fly zones.
- **Staying in central area:** I will use unit tests to ensure that the drone never leaves the central area once it has entered. I will check that every coordinate is inside the central area once it has entered the area.
- **Valid orders only:** I will use integration testing to ensure that only valid orders have an associated flight calculation. I will create valid and invalid orders and verify path calculations are ignored for the latter.

### F2: Order Validation

I will create unit tests for each validation criteria from the ILP Spec. I will simulate orders that fail specific criteria, and verify they are rejected with the relevant validation code.

### F3: Endpoint Reachability

Using system-level tests, I will be able to check each endpoint's availability and response. I will use the Postman tool to hit each endpoint and verify the response.

### F4: REST Server Retrieval

I will use an integration-level test to ensure the application fetches the data (e.g. no-fly zones, restaurants, etc.) from the REST server. I will simulate server downtime to ensure that errors are handled.

### F5: GeoJson Format

I will use unit tests on the path calculation function to see if it outputs the path in the requested format. I will validate the structure against a GeoJson schema to make sure it is correct.

## N1: Performance

I will conduct system-wide performance tests to measure the time taken for path calculations. I will simulate high-load scenarios and check that each path is computed within 60 seconds. There are tools available to measure runtime.

## N2: Maintainability

I can perform code reviews to evaluate my code structure, readability, and modularity. I can use web resources to assess code complexity and identify areas for improvement.

## N3: Robustness

I will test the robustness by introducing invalid inputs and simulating unexpected events and edge cases. I will make sure that the program behaves as specified, without crashing.