Functional requirements:

- Order requirements:
    - The system should be able to collect orders from restaurants and deliver it to Appleton Tower
    - The system should only collect and deliver valid orders.
    - The system should be able to classify orders as valid or invalid.
        - The system should not validate an order if…
            - The card details are wrong.
            - The order contains 0 or more than 4 pizzas.
            - The pizzas do not come from the same restaurant
            - Any of the pizzas do not exist in any restaurants
            - The total price of the order does not equal the pre-calculated price plus a £1 delivery fee.
        - If the order is invalid, the system should set its distance heuristic to infinity (or a corresponding NaN value).
        - If the order is valid, the system should calculate the Euclidean straight-line distance from Appleton Tower to the restaurant and store it in a data structure.
    - The system should only accept cards from certain companies (AMEX, Visa, Mastercard).
    - The system should be able to deliver orders such that it maximises the number of orders delivered in a day.
    - The system should ensure that the drone only picks up one order at a time.
    - The system should sort the priority of all the orders based on the straight-line distance heuristic to the respective restaurants.
- Navigational requirements:
    - The system should be able to fly the drone in 16 compass directions.
    - The system should not send the drone out if the drone will not be able to fly the complete path due to a low battery level.
    - When the drone is collecting or delivering an order, the system should have it hover for one move above Appleton Tower or the restaurant.
    - The system should be able to find the set of nodes that the drone must visit to ensure a shortest path from Appleton Tower to the restaurant.
    - The system should be able to generate a visibility graph of the provided restaurants, no-fly zones, and Appleton Tower.
- File writing requirements:
    - The system should write out three files and label them with dates in YYYY-MM-dd format to differentiate between the days:
        - deliveries.json
            - The system should write all the orders for that day, along with its delivery state and its existing attributes.
        - drone.geojson
            - The system should write the drone's moves in GeoJSON format such that this file can be imported to geojson.io to visualise its flightpath.
        - flightpath.json

- The system should write out information regarding the drone's flightpath, containing the order being delivered or collected, its previous and current location and its bearing.

Measurable attributes:

- MTBF
- Availability
- Wait time
- Drone utilisation
- Execution time

Non-functional requirements:

- The system should be secure and keep the users' data private and encrypted.
- The system should aim to have the wait time between placing an order and receiving the order be no less than 45 minutes.
- The system should not take less than 60 seconds to find the optimal paths for all orders
- The drone should be always available, in other words, it should have 100% availability. This means that outside of the drone recharging at Appleton Tower, the drone should always be delivering or collecting orders.
- The MTBF (Mean Time Between Failures) should not be lower than 3 months.
- The drone should not give off excess amounts of greenhouse gas pollution and should not cause noise disturbances to citizens.
- The drone should fly above buildings.