1.1.

**Functional Requirements**

1. Order Reception:
   * The system should receive orders via an external source (e.g., a third-party service or API).
   * Orders must include the restaurant's details, a list of pizzas to be delivered, and any necessary card payment information.
2. Order Validation:
   * The system must perform validation checks on incoming orders to ensure accuracy and processability.
   * The system must implement input validation on JSON to help prevent vulnerabilities.
   * Validation criteria include:
     + The number of pizzas does not exceed allowable limits.
     + An order may only contain pizzas from one restaurant.
     + The total price of the order should be the sum of the cost of each pizza plus 100 pence as delivery fee.
     + The restaurant must be open on the day of delivery.
     + Restaurant details match a known restaurant.
     + Pizza names match a known menu item.
     + Payment details are complete and valid.
3. Flight Path Calculation:
   * The system must calculate the optimal flight path for the drone to travel from the restaurant to Appleton Tower.
   * Flight paths must be computed using straight-line segments of 0.00015 degrees, restricted to 16-point compass directions (N, NNE, NE, etc.).
   * The calculated flight path must avoid designated no-fly zones to ensure safety and compliance with regulations.
   * Flight path calculations must consider that once the drone enters the central area, it is not allowed to leave.
4. REST Service Endpoints:
   * The REST service should provide endpoints for:
     + Retrieving calculated flight paths as JSON data.
     + Validating the feasibility of an order before acceptance.
     + Handling errors gracefully, with appropriate HTTP status codes and descriptive error messages.
5. Deployment Requirements:
   * The entire system must be packaged as a Docker container to facilitate testing and deployment.

**Measurable Requirements**

1. API Performance:
   * Each REST API call must complete within 5 seconds.
2. Delivery Accuracy:
   * The drone must deliver pizzas to coordinates with a tolerance of 0.00015 degrees from the intended delivery location.
3. Validation Speed:
   * The order validation service must process and reject invalid orders within 300 milliseconds.
4. Error Handling:
   * Invalid orders must return clear and specific error codes with descriptive messages, such as:
     + "Restaurant Closed"

**Qualitative Requirements**

1. Fastest Route:
   * The system must prioritize the fastest route between the restaurant and Appleton Tower while adhering to operational constraints.
2. System Reliability:
   * The system must handle order processing and flight path calculations without crashing, even when encountering invalid inputs.
3. Data Consistency:
   * Order data, including IDs, payment validation, and flight path information, must remain consistent and accurate throughout the process.
4. Code Maintainability:
   * The system should follow clean coding principles, making it easy for developers to add new features or fix bugs.
   * Proper documentation must accompany all modules to facilitate onboarding and maintenance.