2.1. Test Plan

**1. Introduction**

* **Objective**: To ensure the system reliably processes orders, calculates flight paths, and meets performance requirements while handling input validation and no-fly zone restrictions.
* **Scope**: This test plan covers API endpoints, order validation, flight path calculations, performance metrics, and error handling.

**2. Test Strategy**

* **Approach**: A Test-Driven Design (TDD) approach is used. Unit tests are written for core functions before implementation, followed by integration and system tests.
* **Tools and Frameworks**:
  + Unit Testing: JUnit
  + REST API Testing: SpringBoot Test Framework, JUnit
  + Mocking: Mockito
  + Docker Testing: TestContainers

**3. Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Requirement | Description | Expected Outcome |
| TC001 | Order Reception | Verify that the system accepts orders from an external API via POST requests with a JSON body. | Orders with valid JSON data are successfully received and processed, returning a 200 status code |
| TC002 | Order JSON Validation | Validate the JSON structure for an order request | Invalid JSON formats return a 400 status code with an error message describing the issue |
| TC003 | Order Validation (Pizza Limits) | Validate that the number of pizzas does not exceed limits | Orders exceeding limits are rejected with an error message that indicates the pizza quantity violation and 400. |
| TC004 | Order Validation (Restaurant) | Ensure the system rejects orders from restaurants that are closed | Inactive restaurants return a “Restaurant Closed” error message and 400. |
| TC005 | Order Validation (Restaurant) | Ensure the system rejects orders that contain pizzas from multiple restaurants | For different restaurants, return appropriate error message and 400. |
| TC006 | Order Validation (Price) | Ensure the total price is correct and includes delivery fee. | The total price should be the sum of each pizza’s cost plus 100 pence delivery. |
| TC007 | Payment Details Validation | Validate complete and valid payments | Incomplete or invalid payment information returns a payment error with a specific message |
| TC008 | Flight Path Calculation | Compute the optimal flight path | Correct path coordinates are returned in JSON format |
| TC009 | Flight Path avoids No-Fly Zones | Ensure flight paths do not intersect no-fly zones | Paths avoid specified restricted coordinates |
| TC010 | REST Endpoint for Flight Paths | Test that the flight path retrieval endpoint returns correct data | JSON output includes all required fields with valid data |
| TC011 | REST API Performance | Measure API response time for flight path calculation | API responds within 5 seconds. |
| TC012 | Order Validation Speed | Ensure invalid orders are rejected within 400ms. | Validation response time is less than 400ms. |
| TC013 | Delivery Accuracy | Check that delivery coordinates are within 0.00015 degrees of the target | Delivery within the specified tolerance range |
| TC014 | Central Area | Check the flight path does not deviate from the central area after entering | Flight path does not leave central area as required. |
| TC015 | Code Maintainability | Review code for modularity and clean coding practices | Code is modular, with clear documentation, comments, and low complexity |

**4. Test Data**

|  |  |
| --- | --- |
| Test Data Description | Example Values |
| Order JSON Example | {  "orderNo": "010ECB65",  "orderDate": "2025-01-13",  "priceTotalInPence": 2600,  "pizzasInOrder": [  {  "name": "R2: Meat Lover",  "priceInPence": 1400  },  {  "name": "R2: Vegan Delight",  "priceInPence": 1100  }  ],  "creditCardInformation": {  "creditCardNumber": "5586275969937464",  "creditCardExpiry": "06/25",  "cvv": "133"  }  } |
| No-Fly Zone Coordinates  (retrieved from [ilp-rest-2024.azurewebsites.net/noFlyZones](https://ilp-rest-2024.azurewebsites.net/noFlyZones)) | {  "name": "Dr Elsie Inglis Quadrangle",  "vertices": [  {  "lng": -3.1907182931900024,  "lat": 55.94519570234043  },  {  "lng": -3.1906163692474365,  "lat": 55.94498241796357  },  {  "lng": -3.1900262832641597,  "lat": 55.94507554227258  },  {  "lng": -3.190133571624756,  "lat": 55.94529783810495  },  {  "lng": -3.1907182931900024,  "lat": 55.94519570234043  }  ]  } |
| Central Area Coordinates  (retrieved from [ilp-rest-2024.azurewebsites.net/centralArea](https://ilp-rest-2024.azurewebsites.net/centralArea)) | {  "name": "central",  "vertices": [  {  "lng": -3.192473,  "lat": 55.946233  },  {  "lng": -3.192473,  "lat": 55.942617  },  {  "lng": -3.184319,  "lat": 55.942617  },  {  "lng": -3.184319,  "lat": 55.946233  },  {  "lng": -3.192473,  "lat": 55.946233  }  ]  } |
| Restaurant JSON Definition  (retrieved from [ilp-rest-2024.azurewebsites.net/restaurants](https://ilp-rest-2024.azurewebsites.net/restaurants)) | {  "name": "Civerinos Slice",  "location": {  "lng": -3.1912869215011597,  "lat": 55.945535152517735  },  "openingDays": [  "MONDAY",  "TUESDAY",  "FRIDAY",  "SATURDAY",  "SUNDAY"  ],  "menu": [  {  "name": "R1: Margarita",  "priceInPence": 1000  },  {  "name": "R1: Calzone",  "priceInPence": 1400  }  ]  } |

**5. Performance Testing**

1. Objective: Ensure the system can handle errors gracefully and maintain performance even when invalid input or unexpected conditions occur.

Metrics:

* + Flight Path API Response Time:  
    The response time should be within 300ms when the error is due to invalid input.
  + Error Handling:
    - The system should return a 400 status code for invalid input or processing errors.
    - The error message should be clear and descriptive, specifying the reason for the failure (e.g., "Invalid Restaurant ID" or "No Fly Zone Breached").

1. Objective: Test how the system performs when calculating flight paths with a large number of no-fly zones.

Metrics:

* The flight path calculation should still complete within 5 seconds even with complex scenarios.

**6. Test Coverage**

I aim to cover 80-90% of my code with unit tests.