B218172

# **Software Testing 2025/2026 – Requirements** (LO 1.1)

Functional Requirements

- R1.1 – The REST service must validate all orders submitted through POST requests and return a result
    - R1.1.1 – Empty, missing or invalid JSON order bodies must return a 400 (BAD REQUEST) HTTP code
    - R1.1.2 – If any of the following conditions occur, the service must return a result with [“orderStatus”: “INVALID”] and the corresponding OrderValidationCode:
        - The card number is not 16 numerical digits: CARD_NUMBER_INVALID
        - The card’s expiry date is before the order date: EXPIRY_DATE_INVALID
        - The card’s CVV is not 3 numerical digits: CVV_INVALID
        - The total price does not equal the price of the pizzas plus 100: TOTAL_INCORRECT
        - The price detailed for a defined pizza does not match its definition: PRICE_FOR_PIZZA_INVALID
        - A pizza in the order is not defined: PIZZA_NOT_DEFINED
        - The order contains more than 4 pizzas: MAX_PIZZA_COUNT_EXCEEDED
        - Pizzas are ordered from multiple different restaurants: PIZZA_FROM_MULTIPLE_RESTAURANTS
        - The order’s date occurs when the restaurant is not open: RESTAURANT_CLOSED
        - The order contains no pizzas: EMPTY_ORDER
    - R1.1.3 - Otherwise, return a result with [“orderStatus”: “VALID”] and [“orderValidationCode”: “NO_ERROR”]
    - R1.1.4 – Returned result must be a JSON of form {“orderStatus”: “<status>”, “orderValidationCode: “<code>”}

- R1.2 – The service must calculate and return a valid flight path between the order’s restaurant and Appleton Tower
    - R1.2.1 - The path must be composed of steps in longitude-latitude coordinates 0.00015-degree apart, and only moving in 1 of 16 compass directions
        - R1.2.1.1 – Calculated longitude values must be between 180 and -180
        - R1.2.1.2 – Calculated latitude values must be between 90 and -90
    - R1.2.2 – Path steps must never intersect with any no-fly zones
    - R1.2.3 – Once entering the central area, the path must not include steps outside the area before reaching Appleton Tower
    - R1.2.4 - The path must begin at the restaurant’s location and end within 0.00015 degrees of Appleton Tower
    - R1.2.5 – A hover step must be taken using angle 999 at the beginning and end of the path

- R1.3 – The service must fetch restaurant locations and central area and no-fly zone boundaries dynamically on startup
    - R1.3.1 – These must be fetched from the ILP website endpoints “/restaurants”, “/centralArea” and “/noFlyZones”

- R1.4 – The service must only return defined and appropriate HTTP response codes
    - R1.4.1 - Valid request must return 200 OK with valid JSON or GeoJSON
    - R1.4.2 – Invalid requests must return 400 BAD REQUEST
    - R1.4.3 – The service must not return unhandled exceptions or non-JSON responses

## Measurable Quality Attributes

- R2.1 – Route calculation must complete in fewer than 60 seconds

- R2.2 – Longitude-latitude coordinates must be accurate to $\pm 10^{-12}$ degrees

## Qualitative Requirements

- R3.1 – Code must be readable
  - R3.1.1 – Meaningful variable, class and method names must be used
  - R3.1.2 – Comments must be clear and explanatory

- R3.2 – Code must be maintainable
  - R3.2.1 – Implementation must be modular
  - R3.2.2 - Hard-coded values must be avoided

- R3.3 – The service must be robust and fault-tolerant
  - R3.3.1 - Unhandled exceptions must not occur
  - R3.3.2 - Invalid inputs must not crash the service

- R3.4 – The service must be fully data-driven
  - R3.4.1 – All zone and restaurant (menu, location) data must be fetched from ILP endpoints (see R1.3.1)

- R3.5 – The system must be secure
  - R3.5.1 – Sensitive information such as credit card information must not be stored permanently (only for use in order validation)