Software Testing (2022-2023)

Ignas Kleveckas (s2095960)

Requirements document for the PizzaDronz service

01/10/2022

1. Functional Requirements

1.1. Safety properties

1.1.1. The drone must fly high enough so that it will not crash into even very tall buildings, such as The David Hume Tower.

1.1.2. The drone must be secure against bird attacks. This includes:

1.1.2.1. The drone's pizza contents cannot be opened without key to avoid possible contamination from birds.

1.1.2.2. The drone must have appropriate measures so that it does not drop and does not release the food box carriage in dangerous speed in case of failure.

1.1.3. The drone must not operate under severe weather conditions.

1.1.4. The drone must not fly into certain areas where people are crowded together to minimize the consequences of impact in case of hardware of software failure. Examples include [1]:

1.1.4.1. Shopping areas.

1.1.4.2. Sports events.

1.1.4.3. Religious gatherings.

1.1.4.4. Music festivals and concerts.

1.1.4.5. Marches and rallies.

1.1.4.6. Crowded beaches or parks.

1.1.4.7. Parties, carnivals, and fêtes.

1.1.4.8. Tourist attractions.

1.1.4.9. Other areas specified by the user.

1.1.5. The drone should prioritise flying over the rooftops of buildings to minimize the consequences of impact in case of hardware of software failure.

1.1.6. The drone should return close to Appleton Tower (within 0.00015 degrees distance) before running out of battery energy.

1.2. Correctness properties

1.2.1. All dynamic information which the service needs to function is provided by a newly developed centralized REST-server.

1.2.2. The following data shall be retrieved from the console input:

1.2.2.1. Date for which the orders are processed

1.2.2.2. Rest server base URL address

1.2.2.3. Randomisation seed

1.2.3. Command line input should be validated before computation.

1.2.4. The orders retrieved from the Rest server for the required day should be validated for correctness. The validation shall be performed on:

1.2.4.1. Credit card number – this should match the Visa and Mastercard credit card numbers as of 2022 September.

1.2.4.2. Credit card expiry date – this should be no later than the month when the order was placed.

1.2.4.3. Credit card CVV number – this should be a 3-digit value.

1.2.4.4. Total cost of the order – this should match the total pizza cost from the restaurant data in the Rest server.

1.2.4.5. Pizza names – these should be defined in the restaurant menus in the Rest server,
1.2.4.6. Pizza count – this should be between 1 and 4 inclusive.
1.2.4.7. Pizza supplier – all pizzas must be delivered from exactly one supplier.
1.2.4.8. Orders that are invalid due to reasons mentioned in 1.2.4. shall not be delivered.

1.2.5. For valid input described in 1.2.2, the resulting software system shall output the following 3 files using the date format YYYY-MM-DD (which is the ISO 8061 format):
1.2.5.1. The first file (deliveries-YYYY-MM-DD.json) records both the deliveries and non-deliveries made by the drone.
1.2.5.2. The second file (flightpath-YYYY-MM-DD.json) records the flightpath of the drone move-by-move.
1.2.5.3. The third file (drone-YYYY-MM-DD.geojson) is the drone's flightpath in GeoJSON-format.

1.2.6. The flight of the drone is subject to the following stipulations:
1.2.6.1. The drone cannot move more than 2000 moves per day before it runs out of battery.
1.2.6.2. The drone can only move in one of the 16 major compass directions at a time. These are the primary directions North, South, East and West, and the secondary directions between those of North East, North West, South East and South West, and the tertiary directions between those of North North East, East North East, and so forth.
1.2.6.3. As the drone flies, it travels at a constant speed and consumes power at a constant rate.
1.2.6.4. Every move when flying is a straight line of length 0.00015 degrees. Because of unavoidable rounding errors in calculations with double-precision numbers these moves may be fractionally more or less than 0.00015 degrees. Differences of $\pm 10^{-12}$ degrees are acceptable.
1.2.6.5. The moves are of two types, the drone can either fly or hover—the drone can change its latitude and longitude when it flies, but not when it is hovering i.e., when it makes a hover move—flying and hovering use the same amount of energy.
1.2.6.6. The drone must hover for one move when collecting a pizza order from a restaurant and do the same when delivering pizzas to the roof of the Appleton Tower.
1.2.6.7. The drone is launched each day from the top of the Appleton Tower at location (−3.186874, 55.944494).

1.2.7. The drone cannot carry more than one order of a maximum of four pizzas at a time.
1.2.8. Only one drone is available for making the deliveries.
1.2.9. Once the drone has entered the Central Area, it cannot leave it again until it has delivered the ordered pizzas to the roof of the Appleton Tower.
1.2.10. Every order is subject to a fixed delivery charge, which is £1.
1.2.11. The box can contain pizzas up to 14 inches in diameter, but not larger than this.
1.2.12. The system shall ignore orders that are not available (distorted) on the REST server.

1.3. Liveness properties:

1.3.1. In case the system is started in a suitable state (i.e., valid input described in 1.2.2), the system shall terminate after writing the output files described in 1.2.5. even if the data on the Rest server is corrupted.

1.3.2. In case the system is started in an unsuitable state (i.e., invalid input described in 1.2.2), the system shall terminate with an appropriate error message without creating the output files.

2. Measurable quality attributes of the system

2.1. The system should aim to have a runtime of 60 seconds or less before terminating. This runtime should be achieved on a machine that has its system specifications similar to or better than the machine student.compute.inf.ed.ac.uk when it is lightly loaded (i.e., when the who command lists fewer than ten users using the machine).

2.2. The use of disk space, processor (CPU usage less than 40% in standard workload), memory, network on the user's device should be minimised.

2.3. The system should not have more than 1 critical failure per 200 hours of operation at any time.

2.4. The system shall provide correct output 99% of the time.

2.5. The system shall process data according to the UK General Data Protection Regulations (UK GDPR).

2.6. The system shall provide interoperability between multiple operating systems. The system should also compile and run on a Java 18 installation.

2.7. Any libraries included in the system must similarly be implemented in Java for maximum portability. This shall be checked by finding the source code of the libraries that are used.

2.8. The system code shall be readable and provide maintainability for the future developers:

2.8.1. All public interfaces shall contain Javadoc comments to provide better readability and maintainability in the future.

2.8.2. All methods shall not be longer than 60 lines of code, each no longer than 100 characters from the start.