Software Testing

Portfolio

**Outline of Software Being Tested**

The software is a drone-based pizza delivery system, where customers can order pizzas from local restaurants to the top of Appleton Tower. When a customer makes an order, it must be accurately recorded and assessed for validity. If it is valid, it is then considered amongst all valid orders for the day in order to form an optimal flightpath that delivers any many pizzas as possible within 2000 fixed-size drone movements.

While the drone is picking up and delivery pizzas, it is not allowed to enter any no-fly zones – these are areas on campus where there are higher concentrations of people and so the drone may prove a danger.

All data, including orders with customer details, restaurant information, and no-fly zones, is stored in JSON on a Rest server, and accessed by the Rest API.

The software is programmed in Java, using Maven as the build tool and JUnit for automated testing.

I have stored the software and all evidence documents on a private GitHub repository. I have sent collaboration invites to the specified markers.

**Learning Outcome 1 [25%]**

* 1. Range of requirements: functional requirements, measurable quality attributes, qualitative requirements.

A wide range of requirements were listed, spanning functional, measurable quality, and qualitative attributes.

* 1. Level of requirements: system, integration, and unit.

Detail was given on examples of system, integration, and unit requirements, covering functional and non-functional requirements.

* 1. Identifying test approach for chosen attributes.

Appropriate test approaches were chosen for the requirements of order validation and integration between flightpath calculation and no-fly zones.

1.4. Assess the appropriateness of your chosen testing approach.

Testing approaches were analysed and why each was appropriate.

**Learning Outcome 2 [25%]**

2.1. Construction of the test plan.

DevOps lifecycle was identified. Also identified stages of the lifecycle where testing of certain requirements should be prioritised.

2.2. Evaluation of the quality of the test plan.

Disadvantages were given regarding DevOps, the risks of unrepresentative test sets, and large amounts of resources required.

2.3. Instrumentation of the code.

Static instrumentation is mentioned in terms of code coverage for order validation, and performance instrumentation in mentioned in terms of flightpath and no-fly zone integration.

2.4. Evaluation of the instrumentation.

Advantages of instrumentation were given.

**Learning Outcome 3 [15%]**

I passed the LO 3 quiz and so have achieved a mark of 2/5.

**Learning Outcome 4 [15%]**

I passed the LO 4 quiz and so have achieved a mark of 2/5.

**Learning Outcome 5 [20%]**

5.1. Identify and apply review criteria to selected parts of the code and identify issues in the

code.

I did not complete this objective.

5.2. Construct an appropriate CI pipeline for the software.

Stages of CI pipeline were identified and applied to the software.

5.3. Automate some aspects of the testing.

Discussed automation of generation of synthetic data, execution of unit test, and regression testing.

5.4. Demonstrate the CI pipeline functions as expected.

Analysed the sort of errors CI pipelines will show over the course of the lifecycle.