Software Testing Portfolio

Project Description

The project I will be testing is the ILP assignment from 2021. This system aims to plan a path for a drone to deliver a list of lunch orders to requested locations from various restaurants and cafes in a specified area. It takes as input:

1) A day, month, and year triple that specifies a date.
2) A port number to connect to a database containing order information including delivery location and items ordered over several dates.
3) A port number to connect to a web server containing specified no-fly-zones for the drone, landmarks that can be used to help plan the flight path (unused in my implementation), and the menus for the cafes and restaurants used for the system.

The outputs of the program are:

1) a GeoJSON file with the computed drone path
2) Two relation updates to the database detailing the flightpath and successful deliveries.

The restrictions on the program are as follows: The drone must stop within a specified distance of the pickup and delivery points to successfully deliver the orders, the drone's battery only lasts a total of 1500 'moves' of a specified distance, the drone must embark and return to the same start location before the battery runs out and the drone must not enter any no-fly-zones.

The project, along with supporting documents for this portfolio, can be inspected at the following link: https://github.com/MajesticBevans/ilp

Stakeholders for the system

- Customers placing orders
- Shop Owners/workers
- Owners of buildings in no-fly-zones
- Entity who commissioned the project
- Members of the public in the local environment

A list of key requirements and their type can be found in the GitHub repository at Requirements.pdf within the software_testing directory.

Many of my suggested test approaches will require large quantities of sample data to be generated. This may take a significant amount of time and effort, but in my opinion will be worth it as the data will have use to test many aspects of the system, and the generated data only needs to conform to a small number of constraints, so can be pseudo-random.

Several of my approaches also require the majority of the system to be run, rather than individual units. However, were the sample data above generated properly, it could be inserted at many different points in the system execution and this would address this problem.