Software Testing 2024: Your Portfolio

# Outline of the Software Being Tested

This project implements a drone delivery system for pizza, focusing on validating orders and calculating optimised delivery paths. The system includes REST endpoints that enable interaction with various features like order validation, distance calculation, and pathfinding. There were 2 sections of this project but I will mostly be talking about the second section as it is more complex and much more testing was executed. The project can be found at <https://github.com/luketervit/ilpcw2>.

# Learning Outcomes

1. Analyze requirements to determine appropriate testing strategies [default 20%]
   1. Range of requirements, functional requirements, measurable quality attributes, qualitative requirements, …

One requirement for the coursework was for it to be modular. This was kept in mind when implementing the first section of the coursework to be able to implement new features in the second feature. Another requirement for this system was for it to analyse different orders and return either valid or why the order is invalid.

Another requirement of this coursework was to calculate the path from a restaurant to Appleton for a valid order. The success metrics for this task were an optimal path in a low runtime.

Must be robust

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

All methods must work correctly

Units must work seamlessly for integration

These integrated units must also work with each other to get the functional system requirements

robust

* 1. Identifying test approach for chosen attributes.

unit testing to verify validateorder

integration testing using postman to test endpoints work for the full system and correctly fetch API endpoints

integration testing using docker

alanlysis of runtime

runtime of unit tests and of calcdelivery?

* 1. Assess the appropriateness of your chosen testing approach.

<https://ilp-rest-2024.azurewebsites.net/orders> has a list of orders so unit testing was easy

Impossible to do unit testing for calcdeliverypath as the only way to check the path was by pasting the result into geojson.io. used postman to see runtime of calcdeliverypath.

Only able to do unit testing on orders

There was not many restaurants so tested them individually

Robust

1. Design and implement comprehensive test plans with instrumented code [default 20%]
   1. Construction of the test plan

Given a list of orders so knew I could use unit testing on orders

Knew I had to implement ordervalidation to calcdeliverypath so had to make ordervalidation

Knew paths would be graded on effieiency and runtime so had to keep that in mind

be easily integrated into the rest of the system

Validate the system against defined requirements

Ensure a low runtime

All tests pass

Accurate path

* 1. Evaluation of the quality of the test plan

All requirements are covered

Testing is measurable

* 1. Instrumentation of the code

Instrumentation was not needed

* 1. Evaluation of the instrumentation

1. Apply a wide variety of testing techniques and compute test coverage and yield according to a variety of criteria [default 20%]
   1. Range of techniques

Functional testing

Edge case testing

Performance testing

* 1. Evaluation criteria for the adequacy of the testing

Test coverage-% of code ran when doing testing

Risk coverage- verify tests try high risk tests that are on the edge of passing/failing

* 1. Results of testing

Performance- system handled 600 unit tests for validateorder and passed all 7 restaurants paths

No defects found

* 1. Evaluation of the results

Unit tests cover most of the expected outputs

Did not do robust testing on pathfinding

Would add more test cases for pathfinding

1. Evaluate the limitations of a given testing process, using statistical methods where appropriate, and summarise outcomes. [default 20%]
   1. Identifying gaps and omissions in the testing process

No robust testing on pathfinding

No testing on coverage

Did not do testing on undefined orders or orders with no data

* 1. Identifying target coverage/performance levels for the different testing procedures

Aimed to get runtime <1 second for deliverypath

Aimed to pass all tests for ordervalidation

* 1. Discussing how the testing carried out compares with the target levels

All pathfinding was below a second except for one restaurant ~2 secs and all orders passed

* 1. Discussion of what would be necessary to achieve the target levels.

To guarantee all input for orders passed I would need to invalidate orders that are of the wrong input and for calcdeliverypath optimize even more

1. Conduct reviews, inspections, and design and implement automated testing processes. [default 20%]
   1. Identify and apply review criteria to selected parts of the code and identify issues in the code. [default 20%]

Code quality

Achieves functional and non functional requirements

Code is modular for expansion

* 1. Construct an appropriate CI pipeline for the software

Github source

* 1. Automate some aspects of the testing

Automated unit tests

* 1. Demonstrate the CI pipeline functions as expected.