Software Testing Portfolio

# Introduction:

The project I have chosen to write my portfolio on is the ILP project. It considers a pizza drone delivery system, where users may order items from restaurants to be delivered to the top of the Appleton tower. We are concerned with implementation of the order validation and planning of the drone flightpath. The project specifications and details given to us are provided in the ILP coursework specifications[1]. The GitHub repo is here:

# Learning Outcomes

1. Analyze requirements to determine appropriate testing strategies [25%]

1.1 Range of requirements, functional requirements, measurable quality attributes, qualitative requirements, …

We will look at the **5 relevant stakeholders**, and talk about their possible real needs(**validation**) vs the specification requirements(**verification**). A full range of **18** of the project **functional** **requirements** will be listed as well as **7 main measurable qualities** that relate to the system. This can be found in Project Requirements[2].

1.2 Level of requirements, system, integration, unit.

The level of requirements considered were for **two stages** of the development process, the first submission and the final submission. When the code was not yet fully developed, the system and integration levels were vastly different in depth. Several examples of each **system**, **integration** and **unit** level requirements were provided in Level of requirements[3], as well as the requirements they fulfilled.

1.3 Identifying test approach for chosen attributes.

A range of **5 possible test approaches** were considered. The project implements **functional** testing with **structural** testing (**white and black box testing**), **regression** testing (**grey box testing**) and **manual and automated inspection techniques** (**Static testing**) within my project. More details about test approach for satisfying requirements can be found in Test Approaches[4].

1.4 Assess the appropriateness of your chosen testing approach.

The chosen testing approaches (**white, black, gray and static testing**) was evaluated using the **quality process**, where **completeness, timeliness and cost effectiveness** was considered for all the tests. More details are referred in Test Approaches[4]. Also refer to Level of Requirements[3], where we have examples of different testing levels verifying the specifications.

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

We considered **3 sample requirements** to demonstrate the process of planning tasks and determining their step in a **lifecycle**. The tasks were identified with their **priority** level based on their importance. More details about scheduling tasks and the plan can be found in the Planning and Evaluation Document[5].

* 1. Evaluation of the quality of the test plan

More details could be found in the Planning and Evaluation Document[5]. Where we talk about the task **time costs**, **vulnerabilities** of the plan, and the real **risks** and potential effects it may have on the stakeholders.

* 1. Instrumentation of the code

We provided the necessary **scaffolding** information and **instrumentation** required to carry out the set tasks in 2.2. for each of the **3 requirements** listed. More details can be found in Planning and Evaluation Document[5].

* 1. Evaluation of the instrumentation

We discuss the in more detail how adequate the instrumentation for the tasks were. Where a lot of synthetic data were combined with data from the REST server for testing were used. **2** ways instrumentation could be **improved** were also provided but also their new introduction of **constraints** it would have on the project too. More in Planning and Evaluation Document[5].

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

3.1 Range of techniques

The Testing Techniques Document[6] showcases the different range of techniques used, including **systematic functional testing, structural testing**, and examples of **testing measurable qualities** that includes all **3 levels of testing** (unit, integration and system). With each pointing to the referenced test class.

3.2 Evaluation criteria for the adequacy of the testing

Following an optimistic testing approach, we described the slight **lack of adequacy of tests** due to **time constraints** and the **coverage** of class tests with the use of systematic functional testing and structural tests in Testing Techniques Document[6].

3.3 Results of testing

**Two examples** of the many faults found during the development of the system was described in detail, how the **integrated testing** helped uncover the **order validation** fault and the **manual system testing** detected the **efficiency requirement fault**. This is described in more detail in Testing Techniques Document[6].

3.4 Evaluation of the results

We consider the adequacy criterion for the project test suites and tradeoffs to achieve our results. Topics here will tie in with ideas mention in 3.2. Coverage is considered and the overall confidence of the system was also determined in Techniques Document[6].

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
   2. Identifying target coverage/performance levels for the different testing procedures
   3. Discussing how the testing carried out compares with the target levels
   4. Discussion of what would be necessary to achieve the target levels.
2. Conduct reviews, inspections, and design and implement automated testing processes. [default 20%]

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

More details will be included in Inspections and automated testing process[7]. The review goes through a **system of activities** as the final part of the testing process. Mainly checks for functionality and also **general code quality** is upheld as part of the **checklist for review process**.

5.2 Construct an appropriate CI pipeline for the software

In automated testing process[7], I have stated steps on how I would set up continuous integration for the hosting site GitHub using **Jenkins**, a widely known CI tool, and how to configure jobs to allow **automated testing**.

5.3 Automate some aspects of the testing

In automated testing process[7], I have stated parts of the development process that could be automated, such as: **functional and structural tests**, **syntax checking**, **code formatting**, **dependency errors** etc.

5.4 Demonstrate the CI pipeline functions as expected.

In automated testing process[7], we showcased what kind of errors the pipeline would be able to detect, and why it fails, such as **compilation errors**, **poor performance**, or **detection of poor formatting of code**.