COMPX341-23A Assignment Two: Test Plan Specification

Due date: 11:59pm Friday 5th May, 2023
Submission: LaTeX PDF via Moodle and Testing Suite via UoW GitLab
Weight in overall grade: 20%

Abstract

You work as a Software Engineer for SoftFlux, a software engineering company. A new and emerging Smart Home development company, called Encost, has approached your company with a proposed project called The Encost Smart Graph Project (ESGP). ESGP is a software system that enables the visualisation of Encost's devices using a graph data structure.

Introduction

You still work as a Software Engineer for SoftFlux, a software engineering company. While you were busy completing your software design, a different team was working on an alternative design, which the client has accepted. Your company (SoftFlux) and the client (Encost) have agreed on the Software Requirements Specifications (SRS) document and a Software Design Specifications (SDS) document. Your task now is to continue with the Functional Software Testing. Your task is to study the SRS and SDS documents that have already been accepted by the client and propose a Functional Software Test Plan. You will then use this plan to create a testing suite for test-driven development.

Five SDS documents will be uploaded to Moodle. You must select one of these SDS documents as the 'client accepted version'. Please note, you cannot select your own SDS document. Overall, in the assignment, you are required to do the following.

1 Complete a Functional Software Test Plan

A LaTeX template has been provided. Your task is to use the template to compile an engineering report for the project's Functional Software Test Plan. Ensure you describe how each of the requirements in the SRS and SDS document will be tested. It is expected that you recommend automated testing to some extent, in these cases also recommend software tools that can perform such tests. Your functional software test plan must meet the following requirements:

- Black-box testing:

  - Write tests to cover all *High Priority* functional requirements in the SRS, based on the design laid out in the SDS. **Please note, the SRS has been updated to reflect a change in some priorities. Please ensure that you use the most up-to-date version of the SRS (Version 1.1)**

- White-box testing:
  - Provide pseudocode for one of the Summary Statistic requirements. E.g. *The system should use the information stored in the graph data structure to calculate the number of devices that exist in each device category.*
  - Branch coverage testing: Develop tests that achieve 100% branch coverage for this pseudocode

- Mutation testing:
  - Generate four mutants based on your whitebox testing pseudocode
  - Select two test sets (i.e., two sets of input-output pairs) and calculate the mutation score

For each test, specify:

- Level of test: e.g., unit vs. integration. And if integration, with what components
- Test technique: e.g., equivalence classes, boundary values, coverage, mutation testing, etc.
- Test inputs: what inputs will be tested
- You can use a test for more than one requirement. If shared tests are used to test different requirements, provide a table that maps test cases to requirements and maybe, levels and components.

2 Complete a Suite of Functional Unit Tests

Use your Functional Software Test Plan (specifically your black box testing) to write a suite of functional unit tests. All tests should be developed using JUnit. Your test suite must be submitted via GitLab (`https://courses-git.cms.waikato.ac.n`).

Notes

1. You are required to submit your test suite using UoW GitLab (`https://courses-git.cms.waikato.ac.n`). Please ensure that you commit to GitLab (providing *good* commit messages) throughout, not just at the end.

2. You are NOT required to implement your application. A Software Engineer will be tasked with implementing the software in such a way that it passes all of your unit tests.

3. As mentioned in class, five of the stronger submissions will be selected for use with Assignment Three. All selected submissions will be anonymised before being made available to other students.

4. You need to put enough information into your test plan and testing suite for someone else to develop an implementation from. Put yourself in their position (which you will be in yourself soon enough) and write with those people in mind.

5. Make sure you use a good, clear and consistent naming convention throughout. The names you chose in this document will be those used by the implementors in their work.