CITS5501 Software Testing and Quality Assurance
Semester 1, 2020
Project

Details

Version: 0.1
Date: 2021-05-14
Changelog:
- 2021-05-14 – initial version

Please check the CITS5501 website to ensure that you have the latest version.

The goal of this project is to assess your understanding of concepts covered in lectures and practised in lab/workshops.

- The assignment contributes 35% towards your final mark this semester, and is to be completed as individual work. It is marked out of 50.
- The deadline for this assignment is **23:59 pm, Sunday 23th May**. However, there is a grace period of 5 days – any projects submitted will be accepted without penalty until 23:59pm, Friday 28th May.
- The assignment is to be done individually.
- The submission procedure is given below under the heading "Submission".
- You are expected to have read and understood the University Guidelines on Academic Conduct. In accordance with this policy, you may discuss with other students the general principles required to understand this assignment, but the work you submit must be the result of your own effort.

Background

You are working for Tempestuous Software, a startup which is developing their product *Agendum*, a "to-do" list and appointments app.

Amongst other features, it:

- monitors current weather and traffic conditions in the user's city;
- uses AI to analyse items on the user's to-do list; and
- tries to determine whether weather or traffic conditions could interfere with any of the user's tasks to do or appointments, and alerts the user if so.

Tasks

1. Input Space Partitioning

One of the Java methods in the system is `assessWeatherImpact`. It takes as input the ID number of an item on the user's "to-do" list, a String representing the user's location (a city name), and a URL to Tempestuous's weather-reporting server.

Here is the documentation and signature for the method:

```
/** Return a measure of the likely impact of the current
* weather on the specified to-do item.
*
* The return value will be a number between 0.0 (no impact)
* to 1.0 (item is likely impossible to currently carry out),
* inclusive.
*
* @param taskID An ID number for an item on the user's to-do list
* @param city The city the user is located in
* @param serverURL The URL of a web-server providing weather updates
* @return A number representing the assessed impact of the
*     weather on the to-do list item.
*/
public double assessWeatherImpact(String taskID, String city, URL
    ↪ serverURL)
```

You will need to apply the *Input Space Partitioning* technique in order to develop unit tests for the `assessWeatherImpact` method. Note that you *need not write code for the unit tests*; but you should be able to state what they should do.

a. Explain whether any other parameters besides `taskID`, `city`, and `serverURL` need be considered. (2 marks)
b. Work through the steps of the Input Space Partitioning process so as to derive at least two test cases for the `assessWeatherImpact` method. You need not exhaustively list every characteristic and partition you would use in the process, but should mention at least two characteristics. (10 marks)
c. Based on your answer to (b), describe two test cases for the `assessWeatherImpact` method. If any test doubles might be needed, state what sort. (5 marks)

If you need to make any assumptions, or would need additional information to provide an answer, state the assumptions and what information you would need.

Suggested answer length: 1–2 pages.

2. Logic-based testing

One of the requirements you are given for the product is as follows:

- If (i) the weather is wet or the conditions are extreme, and (ii) the task is outdoors, flag the task as needing a weather impact assessment.

a. Suggest an appropriate predicate which could be used to represent the logic conditions in this requirement.

   You should include a list of any variables or functions you use, and explain what they represent. (5 marks)

b. Explain how you would make each clause in the predicate *active*. Choose a particular clause, and provide test cases in which that clause is made active, and set to true and false, respectively. (8 marks)

Suggested answer length: $\frac{1}{2}$ page to 2 pages.

3. Syntax-based testing

The Agendum app can be installed on desktop computers and run from the command-line.

You are given the following requirements for this version of the app:

- It should be possible to invoke the app with either of the flags `--help` or `--version`. When this is done, the app should print user documentation, or the app version, respectively.
- If neither of the above flags is given, then the app should take three arguments: the name of a tasks file, and the URL for a weather server.

a. Give a *grammar* (using the BNF-based syntax we have used in lectures) that describes all the ways the app can be invoked. You should assume every invocation of the app starts with the word "`agendum`". (10 marks)
b. Explain how many tests would be required in order to have *production coverage* for the grammar. Justify your answer. (5 marks).
c. Provide one test case that could be used to test a particular production. Clearly state any assumptions you need to make. (5 marks)

Suggested answer length: 1–2 pages.

Submission

Submit your assignment as a single PDF report using **cssubmit**. Please do not use other file formats (e.g. MS Word or RTF).

Your report should meet the following requirements:

- It should be in PDF format, and use A4 size pages.
- The font for body text should be between 9 and 12 points.
- It should contain numbered headings, with useful heading titles.

- Any diagrams, charts or tables used must be legible and large enough to read when the PDF is displayed at 100% size.
- All pages (except the cover, if you have one) should be numbered.
- If you give scholarly references, you may use any standard citation style you wish, as long as it is consistent.
- Cover sheets, diagrams, charts, tables, bibliographies and reference lists do not count towards any page-count maximums.