ASSIGNMENT 1 [40%]
Quality attributes and black box testing

Overview
For this assignment, your task is to analyse the quality attributes of a software system, and design and document appropriate tests for it using black box techniques. In doing this **you must only use concepts that have been covered in FIT2107**.

This assignment is an **individual, open book** task. Every student must complete and submit their own work. The use of AI in any way is not permitted.

Submissions will be marked out of 40, and will form 40% of your final grade in FIT2107. A late penalty of 5% per day will be applied, and after 7 days a mark of 0 will be given and no feedback will be provided on the submission.

This assignment covers FIT2107 learning outcomes 1, 2, and 3.

Deliverables
You must submit the following evidence using the Assignment 1 submission page on Moodle:

- **Every file included in the assignment template folder.**
  Please convert any docx files to PDF before submission. Leave xlsx files and py files in their original form. Do not submit the files in a zip – upload them individually. This means you need to submit:
    - task_2.pdf
    - task_3.pdf
    - task_4.xlsx
    - task_5.xlsx
    - Task_6_age_tests.py
    - Task_6_discount_tests.py
    - Task_7_categories.xlsx
    - Task_7_tests.xlsx
    - Task_7.py
    - Task_8.pdf
    - Task_9.pdf

Your grade will also be based on the commit history in your unit repository on gitlab.

SUBMISSION DUE: Friday Week 7, 11:55PM

System Under Test

Anything Anytime Library (AAL) is a community library. It offers access to books, magazines, newspapers, and public PCs connected to the internet. It also has a collection of gardening and carpentry tools, and an on-site community makerspace. Any registered member of the community can use the library as a place to work or study, to borrow books or tools, or to use the makerspace facilities.

AAL runs all of its operations with a single piece of software, *Borrowing Administration Terminal* (BAT). Unfortunately, BAT was made for more traditional libraries, so AAL have had to commission custom extensions. With limited funding for public libraries, AAL were unable to afford particularly skilled developers, nor were they able to extend the contract to include testing. So, their custom version of BAT has quite a few bugs.

The core functionality of AAL's modified BAT includes:

- Managing user registration to the library.
- Recording details about borrowed items (e.g., who borrowed it, when, when it should be returned).
- Notifying staff if an item is overdue.
- Upon return of the item, calculating a late fee.
- Prompting for payment of any late fees a patron has when they next attempt to borrow an item or use the library facilities.
- Processing payment of fees.
- Validating that patrons attempting to borrow gardening or carpentry tools are allowed to do so (i.e., have completed the required training, are of an appropriate age, and have no outstanding late fees).
- Validating that patrons attempting to access or book the makerspace facilities are allowed to do so (i.e., have completed the required training, are of an appropriate age, and have no outstanding late fees).

AAL also have a series of custom rules in BAT that have been developed and integrated haphazardly over the years. For example, specific regular patrons receive a discount on any late fees.

AAL is very keen to increase the quality of BAT, so you have been engaged to run some testing and analysis. However, bureaucracy is slow and they are not able to give you access to the software yet. So, for now your testing is restricted to black box techniques. Additionally, you are unable to run your tests in a live environment. Instead, you will need to provide documentation to a staff member from AAL who will run the tests for you and report back.

Tasks

Task 1: File Setup (Marked as part of development history modifier)

You should have the knowledge to complete this task after Week 1.

Download the assignment template (template.zip) from Moodle. Unzip the folder, and copy all of the files into the "Assignment 1" folder in your unit repository. Add, commit, and push the files.

Task 2: Quality Attributes (4 marks)

You should have the knowledge to complete this task after Week 3.

Consider the system under test. In the document "task_2.docx", provided in the assignment template folder, answer the following two questions:

1. What are the three most important quality attributes for the system under test from a user's perspective? Justify each choice.
2. What are the three most important quality attributes for the system under test from a developer's perspective? Justify each choice.

Task 3: User Stories (5 marks)

You should have the knowledge to complete this task after Week 2.

The developers originally hired by AAL to develop custom extensions to BAT did make an attempt at requirements gathering and analysis by writing a series of user stories. In the document "task_3.docx", provided in the assignment template folder, specify whether each user story is well written or poorly written, and justify your answer.

Task 4: Decision Table (5 marks)

You should have the knowledge to complete this task after Week 4.

When AAL were commissioning extensions to BAT, they thought it would be a good idea to communicate their wishes using a decision table instead of trying to write out what they wanted. Unfortunately, whoever actually wrote the decision table made some mistakes, so the information sent to the developers was not accurate. The decision table that was sent is provided in "task_4.xlsx" in the assignment template folder.

Read the following rules AAL wanted to capture, and fix the data in the decision table in "task_4.xlsx" so that it accurately reflects them:

- Any patron over the age of 90 can not use the makerspace or borrow carpentry tools.
- Any patron under the age of 18 can not use the makerspace or borrow carpentry tools.
- Patrons of any age can not use the makerspace, or borrow gardening/carpentry tools if they have not completed the associated training.
- Any person over the age of 50 but under the age of 65 gets a 10% discount on their fees, rounded to the nearest cent.

- Any person aged over 65 gets a 15% discount on their fees, rounded to the nearest cent.
- Patrons aged 90 and above do not have to pay fees.
- Fees can not be negative.
- Ivan Smith is banned from using the makerspace.
- Sandra Atkinson can borrow gardening and carpentry tools without completing the training.
- Brian Lancer does not receive any discounts on fees.

Every time you change a row in "task_4.xlsx", add a comment in column L of the same row to explain what you changed and why.

Task 5: Tests Using Category Partitioning (5 marks)

You should have the knowledge to complete this task after Week 4.

In "task_5.xlsx", provided in the assignment template folder, design a set of tests using category partitioning. For this task you are testing a method "can_borrow", which returns true if a patron can borrow a specified item, and false if they can not. The method has the following parameters:

Parameter	Data type	Description
Type of item	String	Can be book, carpentry, or gardening.
Age	Integer	Can be minor (<18), adult (18 -89), or elderly (>=90).
Length of loan	Integer (days)	Can be short (<=2), medium (<=7), or long (<=14).
Outstanding fees	Float	Can have outstanding fees, or not have outstanding fees.
Gardening tool training	Boolean	Can have completed the training, or not.
Carpentry tool training	Boolean	Can have completed the training, or not.

In determining the expected output for your tests you should follow the rules specified by AAL in task 4, except for those relating to specifically named patrons. Additionally:

- A patron can not borrow any item if they have fees to pay.

Starting from row 3, write one test per row in "task_5.xlsx". Each test should be numbered so it can be referred to, starting from 1 and incrementing by 1 each row.

Task 6: Equivalence Partitioning and Boundary Value Analysis (6 marks)

You should have the knowledge to complete this task after Week 4.

Part A:

In "task_6_age_tests.py", provided in the assignment template folder, write tests using python unittest for a method "type_of_patron", which has the following parameters:

Parameter	Data type	Description
age	Integer	A patron's current age, rounded down to the nearest whole number.

The method should return one of four strings:

- "Minor": if the age is less than 18.
- "Adult": if the age is 18 or over, but less than 90.
- "Elderly": if the age is 90 or above.
- "ERROR": if an invalid age value is used.

When writing your tests, use equivalence partitioning and boundary value analysis to determine appropriate values for "age". Do not use any other testing methods. Write a comment for each test method, identifying whether it is a test related to equivalence partitioning, boundary value analysis, or both.

Part B:

In "task_6_discount_tests.py", provided in the assignment template folder, write tests using python unittest for a method "calculate_discount", which has the following parameters:

Parameter	Data type	Description
age	Integer	A patron's current age, rounded down to the nearest whole number.

The method should return one of five values:

- 0: if the age is 50 or less.
- 10: if the age is over 50, but less than 65.
- 15: if the age is 65 or more, but less than 90.
- 100: if the age is 90 or above.
- "ERROR": if an invalid age value is used.

When writing your tests, use equivalence partitioning and boundary value analysis to determine appropriate values for "age". Do not use any other testing methods. Write a comment for each test method, identifying whether it is a test related to equivalence partitioning, boundary value analysis, or both.

Task 7: Pairwise Testing (7 marks)

You should have the knowledge to complete this task after Week 5.

Part A:

In "task_7_categories.xlsx", provided in the assignment template folder, design a set of tests using pairwise testing. For this task you are testing the same method as you did in task 5. Starting from row 3, write one test per row in, and number each test starting from 1 and incrementing by 1 each row. At this point you should not be providing actual values for each parameter, but specifying which category of value you will be testing. For example, the possible categories for "Type of item" are "book", "carpentry", or "gardening".

Part B:

In "task_7_tests.xlsx", provided in the assignment template folder, write tests that match your design. The test numbers and locations in "task_7_tests.xlsx" should match the test numbers and locations in "task_7_categories.xlsx". That is, the categories nominated for each parameter in test X in row Y should be reflected in the test data for test X in row Y.

Part C:

Once you have completed your test design, implement your tests in python using the unittest framework. Write your tests in the file "task_7.py", provided in the assignment template folder.

Task 8: Choose Your Test Strategy (6 marks)

You should have the knowledge to complete this task after Week 5.

For the following three pieces of functionality identify the most appropriate black box testing strategy. Justify your answer. Remember that you can combine techniques.

1. Whether an item is overdue. The input for this functionality would be the item's due date and the current date. The output would be true if the item is overdue, and false if it is not.
2. The calculation of fees owed. The input for this functionality would be a patron's age, and their outstanding fees. The output would be an amount of money owed, based on their fees and what discount they're eligible for.
3. Whether a person is allowed to use the makerspace facilities. The input for this functionality would be a person's age, and whether they have completed the

appropriate training or note. The output would be true if they can use the makerspace facilities, and false if they can not.

Write your answers in "task_8.docx", provided in the assignment template folder.

Task 9: Find the Bug (2 marks)

You should have the knowledge to complete this task after Week 1.

There is one inconsistency with the requirements presented by AAL in this document. In "task_9.docx", provided in the assignment template folder, answer:

1. What is the inconsistency with AAL's requirements as presented throughout the assignment specification?
2. How did you find it?

Assessment Criteria

This assignment will be marked out of 40, and will form 40% of your final grade in FIT2107. A late penalty of 5% per day will be applied, and after 7 days a mark of 0 will be given and no feedback will be provided on the submission.

- **Development history**
  After your submission is marked, a modifier will be applied to your score based on your development history. The lowest possible modifier is 0.5, and the highest possible modifier is 1.0 (i.e., no grade reduction). Your final grade will be your original grade multiplied by this modifier. To get a modifier of 1.0 you need to:
  - Have all the files from the assignment template in the "Assignment 1" folder in your unit repository, and not in a sub-folder.
  - Make at least 12 commits total.
  - Make at least 2 commits of each file in the template.
  - Use meaningful and concise commit messages.
- **Appropriate use of unit concepts**
  The marker will verify that you have used only concepts covered in FIT2107.
- **Correctness**
  The marker will verify the correctness of your answers.
- **Clarity**
  The marker will verify whether your answers use clear, specific, and appropriate examples. This is particularly important when you are writing justifications.
- **Consistency**
  The marker will verify whether related answers are consistent with each other.
- **Good coding practice**
  The marker will verify whether you have followed good coding practice for writing tests in python, as demonstrated in this unit.