Exercise 5 – Test Design

Goal:

The goal of this exercise is to train the different test design techniques presented during the lecture.

Introduction:

FocusFlow is a web-based task management system designed for individuals and small teams. As part of the FocusFlow user registration process, ensuring strong password security is critical.

In FocusFlow, a password must meet the following criteria (e.g., https://github.com/dgrewe-hse/focusflow):

- Length Requirement: The password must be between 10 and 12 characters long.
- Composition Requirements:
  - It must contain at least one uppercase letter.
  - It must contain at least one lowercase letter.
  - It must contain at least one special character (e.g., ! @ # $ % ^ & *).

If any of these rules are not met, the registration should reject the password. Conversely, a password is accepted only when it meets all the specified requirements.

Exercise 5.1 (20 Points): Black-Box Testing Techniques

You will develop test cases for the password validation feature of FocusFlow using **Blackbox Testing Technique**. In blackbox testing, you do not have any access to implementations, only documentation such as the text above. Your task is to exercise/apply two core test design techniques:

- Equivalence Class Partitioning (ECP)
- Decision Table Testing

Your goal is to use these techniques to systematically identify representative test cases that cover all crucial scenarios—including valid conditions and each type of invalid input.

Steps to perform for ECP:

1. Identify Valid and Invalid Equivalence Classes
2. For each identified class, provide a clear explanation of why the chosen test case appropriately represents that class, including considerations for boundary conditions.

Steps for Decision Table Testing:

1. Define Conditions that apply to the scenarios, e.g., “is the password length valid?”, or “does the password contain at least one uppercase letter?”, …
2. Outline the outcomes of each condition, e.g., accept: if all conditions are met, …
3. Create a Decision Table – structure your table so that each row represents a combination of conditions.
4. Explain your reasoning

Exercise Deliverables:

1. Equivalence Class Analysis Document:
    - Identification and explanation of valid and invalid input classes.
    - Detailed boundary analysis for each class.
2. Decision Table:
    - A complete table showing all combinations of key conditions along with the expected outcomes.
    - A brief discussion of the rationale behind each decision rule.
3. Test Cases List:

- A detailed list of specific passwords representing each equivalence class/decision table row.
- An explanation for why each test case was chosen and what outcome it is expected to produce.

Exercise 5.2 (20 Points): White-Box Testing Techniques

We will enhance the functionality of our FocusFlow application further towards the integration of a database. For this reason, interfaces managing persistence operations are implemented in FocusFlow.

Imagine you have a function called `createTask` in your task management application. This function is meant to create a new task based on input parameters such as the task title, a short description, a detailed description, a due date, a priority level, and information about the creator or assignee. Your goal is to ensure that the function works correctly under all conditions by writing tests that look inside the code (whitebox testing).

In the Java example repository of FocusFlow there are the implementations prepared for the repository as well as a service managing tasks:

- Repository: https://github.com/dgrewe-hse/focusflow/tree/dev/backend/src/main/java/de/hse/focusflow/repository
- Services: https://github.com/dgrewe-hse/focusflow/tree/dev/backend/src/main/java/de/hse/focusflow/service

Exercise:

- First, read through the `createTask` function of the Java implementation (TaskServiceImpl.java) to familiarize yourself with its internal logic. Notice that the function performs several checks, such as verifying that the title is provided, the title is not too long, the due date is in the future, and that only one of either an individual assignee or a team is set.
- Think about the Whitebox testing goals and create a list of test cases for each possible path through the function. Consider tests and enhance the list according to your analysis. Your analysis includes cases such as:
  - Valid Task Creation
  - Missing Title
  - Title Too long
  - Past due date
  - …

Exercise Deliverables:

- Whitebox Test Case Documentation
  - Identification and explanation of valid and invalid test cases given as list.

(Optional) Exercise 5.3 (10Points): Implementation

As an optional task, your team can start implementing similar service and database handling components in the programming language you selected within your project.