ICT3101 Lab 2 part 3

Recap:

- A **Behavior** is an operation with inputs, actions, and expected outcomes.
- A **Scenario** is the specification of a behavior using formal steps and examples.
- A **Feature** is a desired product functionality involving multiple behaviors specified as scenarios.

New:

Epics, Features, User Stories, and Gherkin Scenarios (in SpecFlow):

When we're decomposing software systems into its required functionality, there is a variety of terminology used. It's important for us to be clear about the following terms when describing our systems.

Epic

An Epic is the highest level, most generic, categorization of your system's requirements. An epic represents a business initiative to be accomplished. Here are a few examples of Epics (from Microsoft):

- Allow customers to manage their own account via the Web
- Implement new architecture to improve performance
- Engineer the application to support future growth
- Support integration with external services
- Improve and simplify the user experience
- Increase customer engagement
- Support mobile apps

Feature

Features describe functionality that can be tested with acceptance tests. Features usually describe what the software system should do, and they can be seen as composing Epics. Some examples of Features are as follows:

- Editing the customer information via the web portal
- Refresh the web portal with a new look and feel
- Adding a view options to the new work hub
- Adding a mobile shopping cart
- Support user text alerts

User story

User stories normally describe what the user wants to do. They primarily express a complete piece of functionality. One story can directly translate into a Feature or several can compose a Feature. It is often recommended that they are granular enough to fit into a single project development iteration cycle (e.g., a sprint). They are formally expressed as in the following example:

> **As a** bank clerk,
>
> **I want to** be able to modify the customer information
>
> **so that I can** keep it up to date.

Each story tends to be written from the user's perspective and follows the format: "**As** [a user persona], **I want** [to perform this action] **so that** [I can accomplish this goal]." Note, this does not describe exactly how this functionality is to be implemented, but rather what a user is hoping to achieve and why. Note that in SpecFlow, Features are written in such a way as to map one-to-one to User Stories—with minor formatting differences.

Scenarios

Scenarios are on the same level of granularity as User Stories, but they are expressed more formally and with a view to be involved in acceptance tests for a Feature, or User Story. For SpecFlow, we express Scenarios in the Gherkin language. There are various guidelines as to what constitutes a good Scenario, some of these are as follows:

- **'Given'** represents state, not actions
- **'When'** represents actions
- **'Then'** should represent a visible (to the user) change
  - ideally, not simply an internal event

- Self-explanatory
  - Have you stuck to clear and precise names?
- Must have a unique and understandable purpose
  - What makes it different from the other scenarios?
- Written from the client's perspective
  - Are they focused on business functionality (not about software design)?
  - Are they non-technical?
- Grouped by Epics or Features
  - Can somebody else read them to see if the scenario is correct within a given context?
  - Do they avoid referring to UI elements?
- Represent key examples
  - Are they precise and testable?
  - Can they be repeated?

User requirements to be translated

With our revised knowledge on terminology let's proceed to add more functionality into our system, but now beginning with raw client requirements.

Client requirements:

A group of system quality engineers wish to customize the command line calculator program to perform quality metric calculations. The engineers need to calculate the defect density of a system. Furthermore, they require a custom calculation for the new total number of Shipped Source Instructions (SSI) in the successive releases of a system (2nd release onward). The engineers also wish to be able to use the Musa Logarithmic model to calculate failure intensity and the number of expected failures, performing each of these using a single calculator command.

19. Given the requirements above. Devise an Epic description to represent this set of requirements.
20. Define some User Story examples for the Features.
21. Translate these User Stories into two Features that cover this Epic. Then generate the corresponding feature files using SpecFlow.
22. Create a set of acceptance tests for these Features as Gherkin Scenarios. Add these sets of Scenarios into the corresponding SpecFlow Features and use the BDD process to develop the necessary functions.
23. In using the BDD process to develop the new functions (and corresponding functionality) what were some of the issues you faced? How did you resolve them to get things running?

—The end of part 3 questions—