1. Part A : xyzSoft

The xyzSoft team was impressed by your work in the first phase. Ahmad (from Quality Management Department) loved how he could navigate the projects and view them using the custom timeline. However, he came back with more ideas we need to fulfill in this part.

1.1. Principles Evaluation [20 points]

During this course, we covered several principles, including the Open Closed Principle, Single Responsibility, and the Principle of Least Knowledge. **Focus on the Single Responsibility Principle**, list all classes you implemented in the previous homework, and indicate the logics/responsibilities linked to each class. If a class has several responsibilities, clarify how you can split them. You can use the template below:

Class	Responsibilities	Can be split into.. (if applicable)
Class # 1		
Class # 2		
…		

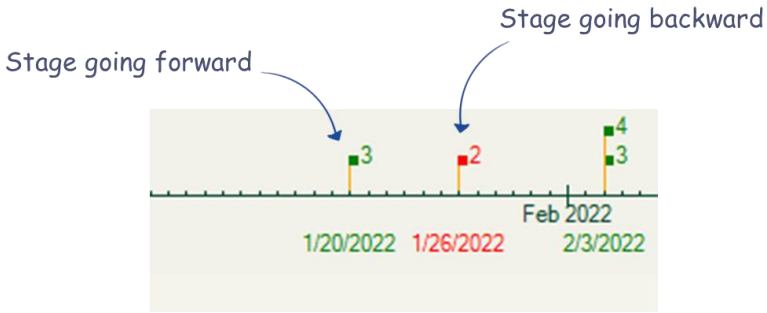
1.2. Project Re-Works [20 points]

Normally projects are moved from one stage to another higher stage. For example, a project would go from stage 1 to stage 2 (old_value=1, new_value=2) and then from 2 to 3, and so on. However, whenever a user finds that there are missing documents after changing the stage, he/ she would return it back to an older stage so that it get filled properly. For example, a user might move the project from stage 3 to 2 or stage 5 to 4. **These movements are called Re-Works**.

Ahmad wants to clearly indicate the number of reworks in the project before and after the project gets Awarded (Awarded means it reached stage # 5). So, you should count the number of reworks before the project gets awarded and also after awarding. You should display the results as follows:

1.3. Timeline Update – indicator of Stages going backward. [20 points]

You need to update the timeline that you did in the previous homework so that it clearly shows stages going forward and stages going backward in different colors (green for forward and red for backward – as shown in the image below). You should implement this as a function and host in an appropriate class and call it in the timeline component.



2. Part B: Composition Design Pattern

File/Folder combination is a typical example of the composite design pattern. A file has a name, size, and extension. A folder has a similar attribute (without an extension) plus a list of files or other folders. You are required to write a demonstration application that traverses files and folders in a selected directory.

2.1.Class diagram [10 marks]

Design a class diagram showing the above-mentioned structure using the composite design pattern.

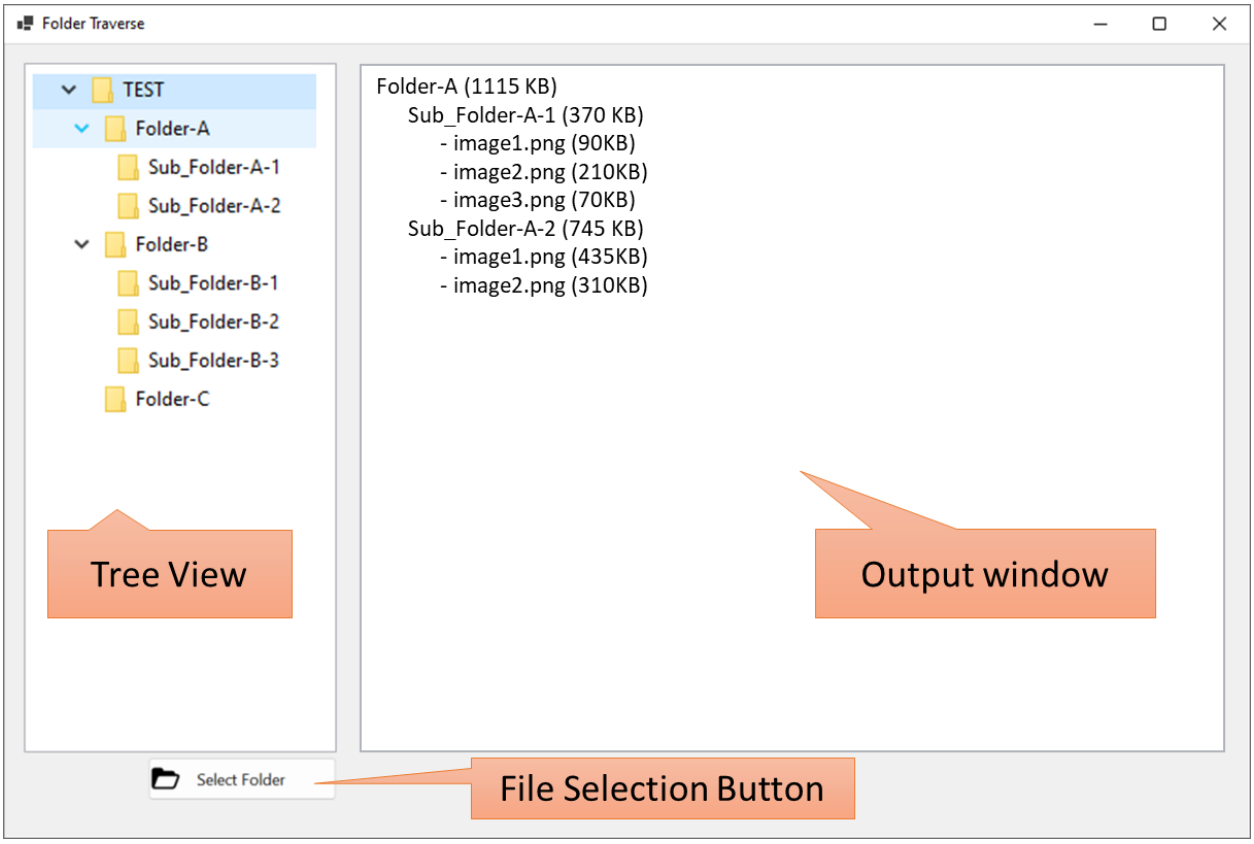
2.2.Traverse a certain folder and read the contents [30 marks]

Implement a GUI client application by which you can choose a certain folder (see the image below). Once you select a folder, you should *recursively* traverse all of its contents (files and folders) and fill in the required information as follows:

- Folder : only name
- File : name, size, extension

After that, your application should fill the tree view with the selected folder and all subfolders. Once you select one of the folders in the tree view, you should show detailed information about its contents in the output window.

After traversing, your application should traverse the created structure (your structure) again and calculate the size of all **folders by single line call**. As an output of this task, show only the code to calculate the size and sample output.



For testing purposes, use a reasonable-sized folder (small but contains subfolders)

<<<<<<<<<<<<<<<<<<<<<<<<<<<< END >>>>>>>>>>>>>>>>>>>>>>>>>>>>