Building on the prototype from Part 1, you will now add functionalities to the GUI UI .NET Core web application. The application should be able to perform the following features:

1. Lecturers can submit their claims at any time with a click of a button:

* Implement this feature in your application.
* Consider the layout, colour scheme, and user flow to make this process as straightforward as possible.
* You should design a simple and intuitive form for lecturers to input their claims.
* The form should include fields for the hours worked, hourly rate, and any additional notes.
* The 'Submit' button should be prominently displayed and easy to click.

2. Programme Coordinators and Academic Managers can easily verify and approve the claims:

* Design a separate view for coordinators and managers.
* This view should display all pending claims and provide options to verify or reject them.
* Each claim should be displayed in a clear and organised manner, showing all the necessary details for verification.
* There should be 'Approve' and 'Reject' buttons for each claim.

3. Lecturers can upload supporting documents for their claims:

* Add a feature that allows lecturers to upload documents.
* Ensure that the uploaded files are securely stored and linked to the corresponding claim.
* You should provide an 'Upload' button in the claim submission form.
* Once a file is uploaded, its name should be displayed on the form.
* Consider implementing a file size limit and restricting the file types to common formats like .pdf, .docx, and .xlsx.

4. The claim status can be tracked transparently until it is settled:

* Implement a tracking system that updates the status of each claim as it moves through the approval process.
* You could represent the status as a simple text label (e.g., 'Pending', 'Approved', 'Rejected') or as a progress bar.
* The status should be updated in real-time whenever a coordinator or manager approves or rejects a claim.

5. The system always provides consistent and reliable information:

* Unit Testing: Write unit tests for the code. These tests should cover all the key functionalities of the system.
* Ensure that your application handles errors gracefuly and displays accurate information. Implement error handling mechanisms to catch and handle exceptions. Display meaningful error messages to the user when something goes wrong.

6. Version Control: Regularly commit and push changes to the GitHub repository (5 Times) with clear and descriptive commit messages?

Remember, the goal of this part 2 is to demonstrate your ability to add functionality to a Gul application.

Focus on implementing the features as described, but also feel free to add any additional features that you think would improve the application.

Submission Guidelines:

* Add Lecturer Feedback in a Word document and show how you implemented the recommendations.
* Version Control: Push your source code and your Documentation to GitHub. Repository to be provided.