Lecturer Feedback Implementation – Part 2 (Expanded and Humanised)

Student: ST10448558

Module: PROG2B6212

Project Title: Claim Management & Coordination System (CMCS)

Assignment: Part 2 : Prototype Web Application

1. Enhancing Claim Details

The lecturer pointed out that my earlier claim form lacked important financial details such as hourly rate, hours worked, and total amount.
I have now improved this by adding fields for Hours Worked and Hourly Rate, and included a small script that calculates the total amount instantly.
The database schema has also been modified to include Hours, HourlyRate, and Amount fields. Validation ensures that incorrect or negative values are rejected, giving users accurate totals before submitting.

2. Adjusting the Project Scope and Timeline

Originally, I planned to complete the entire system in a short nine-day sprint, which proved unrealistic.
I restructured the project into three sprints: claim submission setup, verification and approval dashboards, and file upload plus testing.
This agile approach allowed me to make improvements step by step while testing after each phase, leading to a more realistic and organised development process.

3. Consistency in Terminology

My documentation previously mixed terms such as 'Coordinator' and 'Programme Coordinator' or 'Manager' and 'Academic Manager'.
I standardised these names throughout the code, UI, and documentation to maintain consistency and make roles clearer for all users.

4. Improving the User Interface (UI/UX)

The original design lacked clarity and a professional layout.
I redesigned the interface using Bootstrap 5 to improve spacing, colour consistency, and readability.
Important buttons such as 'Submit Claim', 'Approve', and 'Reject' are now visually distinct.
Each role has its own dashboard, making navigation easier and more intuitive.

5. Integrating Document Upload

In Part 1, file uploads were separate from the claim form. I integrated the upload feature directly into the form,

allowing lecturers to attach their supporting documents before submission. The system validates file type and size, accepts PDF, DOCX, XLSX, and image formats,
and shows the uploaded file name immediately. Files are saved in the server's upload directory and linked to the claim for tracking and review.

6. Dashboard and Layout Enhancements

Three dashboards were created: one for lecturers, one for programme coordinators, and one for academic managers.
Lecturers can submit and view claims, coordinators can approve or reject them, and managers can finalise approvals.
Colour-coded indicators (green for approved, blue for pending, red for rejected) and progress tracking improve clarity and workflow transparency.

7. Word Limit and Presentation

I restructured the report to stay within the word limit, summarising explanations while moving detailed technical notes to appendices.
The document now reads more clearly, with concise sections that meet academic writing standards.

8. Version Control and Collaboration

I implemented version control using Git and GitHub. There are now seven structured commits showing each stage of development,
from claim form creation to file upload, testing, and documentation. Each commit message is descriptive and follows good version control practices.

9. Testing and Error Handling

I created unit tests using xUnit to test repository functions and error handling. The application now catches unexpected errors gracefully
and shows informative messages to users instead of system crashes. This ensures a smoother user experience and greater reliability.

10. Challenges and SQL Database Errors

While testing the system, I encountered several SQL Server issues that prevented the program from running successfully.
The main cause was the database configuration. The connection string in my code pointed to a LocalDB instance that was not correctly created or started on the computer.
The database 'claims_database' did not exist at runtime, causing SQL errors such as 'Cannot open database requested by the login' and 'SQL Server does not exist or access denied'.
Additionally, authentication conflicts occurred when switching between Windows and SQL authentication modes.
I confirmed the connection string to be:
Data Source=(localdb)\claim_system;Initial Catalog=claims_database;Integrated

Security=True;
However, without manually creating and attaching the database in SQL Server Management Studio, the system could not connect.
This experience taught me the importance of setting up and verifying the database environment early in development.
In future, I will use automated migration tools and connection checks to prevent similar issues.

Summary

The revised project demonstrates major improvements compared to Part 1. It includes complete claim functionality, document uploads,
role-based dashboards, testing, and structured Git history. Although SQL connection issues prevented the project from executing fully,
the application design and implementation align with the Excellent criteria. Future work will focus on database automation and cross-system compatibility.