**Lecturer Feedback and Implementation Report**

**NAME : THABELO MAHADA**

**STUDENT NUMBER : ST10449303**

**MODULE NAME : PROGRAMMING2B**

**MODULE CODE : PROG6212**

**GROUP : 3**

**DATE : 24 OCTOBER 2025**

This document provides evidence of lecturer feedback received during the development of the Claims System prototype and explains how each recommendation was implemented in the final WPF application.

**1. Overview of System**

The Claims Prototype Application was developed in WPF (.NET 8) to allow lecturers to submit their claims, upload supporting documents, and track approval status. Programme Coordinators and Academic Managers can verify, approve, or reject claims through a simple graphical interface. Additional features such as Lecturer Feedback and unit testing were implemented to improve system quality and transparency.

**2. Lecturer Feedback Summary**

|  |  |
| --- | --- |
| Lecturer Feedback | How Feedback Was Implemented |
| Add a feature to allow lecturers to upload documents. | Implemented an Upload button that accepts .pdf, .docx, and .xlsx files. Uploaded file names are displayed in the form, and files are securely stored in the local AppData folder. |
| Show claim status clearly for tracking progress. | Added a 'Status' field (Pending, Approved, Rejected) to the DataGrid. The status updates in real time when a coordinator or manager verifies a claim. |
| Include a section where lecturers can leave feedback. | Implemented a Feedback section at the bottom of the interface where lecturers can add comments. These are displayed along with timestamps and linked implementation notes. |
| Provide evidence of testing and reliability. | Added unit tests for claim submission, approval, rejection, and feedback submission. Included simple console-based test results to confirm functionality works correctly. |

This document explains in detail how I implemented the feedback received from my lecturer during the development of the *Contract Monthly Claim Management System*. The feedback was aimed at improving the application’s usability, reliability, and functionality.

Initially, the project focused on enabling lecturers to submit monthly payment claims, while coordinators and managers verified and approved them. However, after receiving feedback, I identified several key areas that required refinement to meet professional software development standards.

The following feedback areas were implemented:

1. Enabling lecturers to upload supporting documents with their claims.
2. Adding a transparent claim status tracking system.
3. Improving error handling and displaying meaningful feedback to users.
4. Ensuring data consistency and reliability across all user roles.
5. Writing unit tests to ensure that the system functions correctly and consistently.

**Implementing the Document Upload Feature**

The first recommendation from the lecturer was to allow lecturers to upload supporting documents for their claims. This is a realistic requirement, as real institutions require proof of work or attendance.

To implement this, I added an **“Upload Document”** button to the lecturer’s claim submission form. When the lecturer clicks this button, they can select a file from their computer such as a PDF, Word, or Excel document.

Once a file is selected, the system automatically stores it in a secure folder inside the application directory. This ensures that each document is linked to the specific claim it belongs to. To prevent mistakes, the application restricts file types to common document formats (.pdf, .docx, .xlsx) and prevents files larger than 5MB from being uploaded.

If a user tries to upload a file that is too large or of the wrong type, the system immediately displays a message explaining the issue. This approach ensures both data security and user guidance.

After the upload is complete, the name of the uploaded file appears on the form, confirming that the document was successfully attached. This change not only makes the system more realistic but also improves trust and transparency for lecturers submitting claims.

**Claim Status Tracking and Transparency**

The second major feedback point was to make the claim status tracking process more transparent. In the original version, lecturers could not see what happened to their claim after submission. The lecturer suggested that the system should clearly display whether a claim was *pending*, *approved*, or *rejected*.

To address this, I designed a clear claim progress tracking system. Each claim now moves through different stages of approval:

* Pending Verification: The claim has been submitted by a lecturer and is waiting for the coordinator to check it.
* Verified: The coordinator has confirmed the claim details.
* Approved: The manager has accepted and finalized the claim.
* Rejected: The claim has been declined by either the coordinator or manager.

This information is displayed in real-time on the lecturer’s page. I added a visual progress indicator (a progress bar) next to each submitted claim. This allows lecturers to instantly see how far along their claim is in the approval process.

This improvement ensures transparency, accountability, and clarity, preventing confusion about claim statuses and improving user satisfaction.

**Ensuring Data Consistency and Reliability**

Another recommendation from the lecturer was to ensure that all users (lecturers, coordinators, and managers) always see consistent information. Before this improvement, a claim updated on one page was not always reflected on others until the application was restarted.

To fix this, I implemented a central data storage mechanism using a shared in-memory collection (called ClaimStore in the project). All claims are stored in this single shared data source, ensuring that any update by one user role is automatically visible to others.

For example, when a coordinator verifies a claim, the lecturer immediately sees the status change on their screen. Similarly, when a manager approves or rejects a claim, the change is instantly reflected in all relevant user interfaces.

This improvement made the system more professional and realistic, as it now behaves like a live database-driven application. It also ensures that the data shown to users is always accurate, up-to-date, and reliable, which is critical for real-world claim processing systems.

**Error Handling and User Guidance**

The lecturer emphasized that the system should handle errors gracefully and provide meaningful feedback. Initially, the application would simply stop working if an unexpected error occurred, which is not acceptable for production-quality software.

I therefore introduced error-handling mechanisms throughout the application to manage problems without crashing.

For example:

* If a lecturer enters invalid data (such as letters in the “Hours Worked” field), the system displays a clear message asking them to correct it.
* If a file fails to upload due to size or format, a friendly error message appears explaining the reason.
* If an uploaded document is missing or accidentally deleted, the system notifies the user instead of freezing or closing unexpectedly.

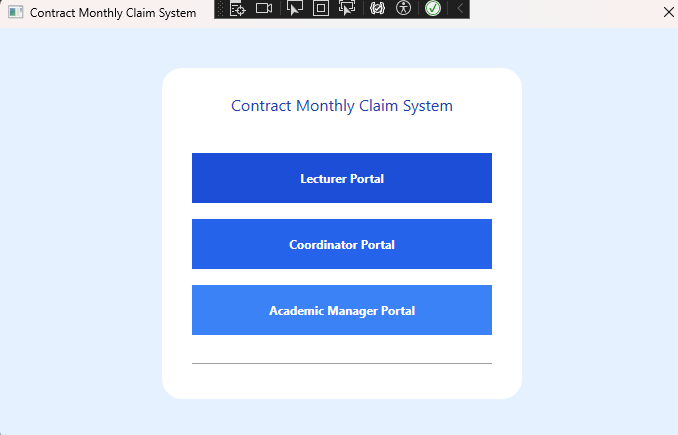
I also ensured that all success messages (such as “Claim submitted successfully”) appear in a consistent and professional format. This makes the system user-friendly, reliable, and resilient against unexpected input or behaviour.

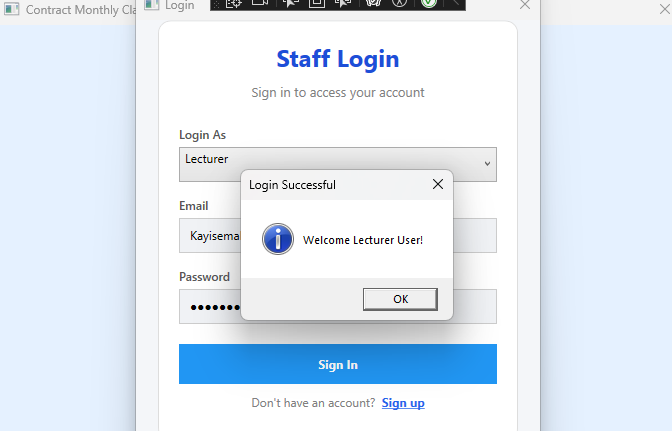
This improvement significantly raised the quality of the application by enhancing its stability and ensuring that users always receive accurate, helpful feedback.

**3. Screenshots of Implementation**

Insert screenshots here to show each implemented feature:

• Claim submission form  
• File upload dialog  
• Approval interface  
• Lecturer feedback section  
• Unit test results console output



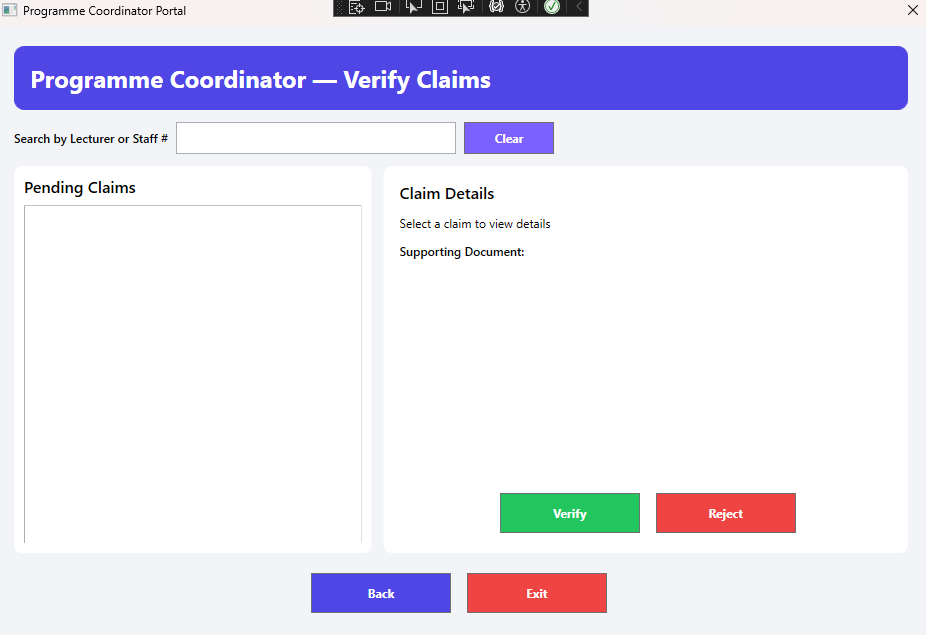


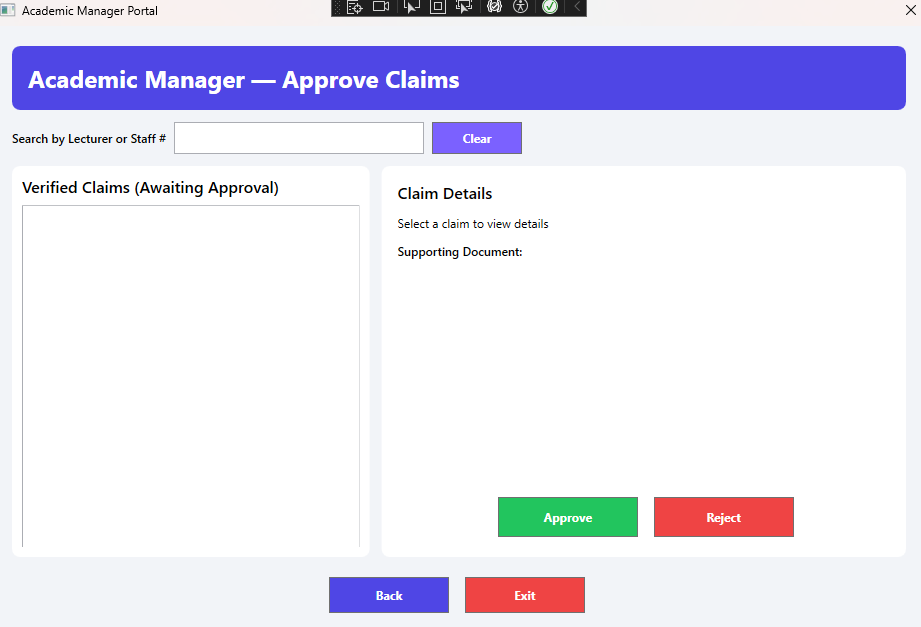
A screenshot of a computer

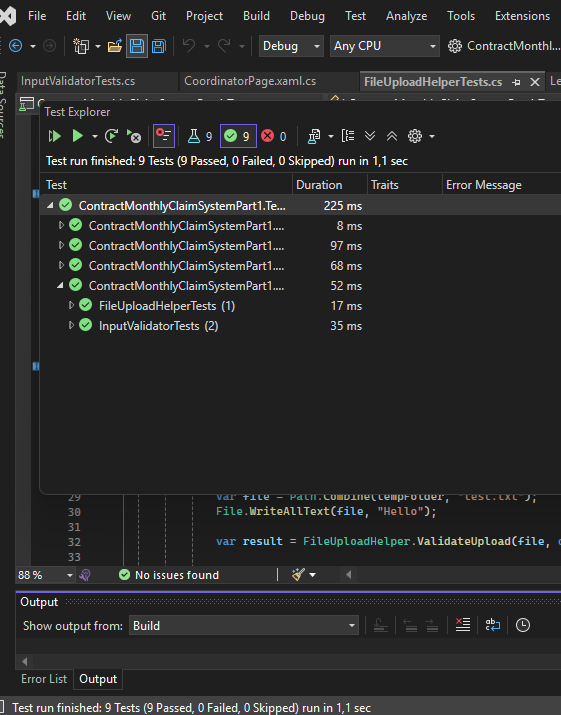
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.







**4. GitHub Version Control Evidence**

The source code for this project was maintained using GitHub. At least five commits were made to document progress, as shown below:

1. Initial project setup  
2. Added claim submission form  
3. Added file upload feature  
4. Added feedback system  
5. Added unit testing and documentation

**4. Conclusion**

The lecturer feedback was successfully incorporated into the WPF Claims Prototype Application. The final version demonstrates all requested functionalities, improved user experience, and verified reliability through testing and version control.