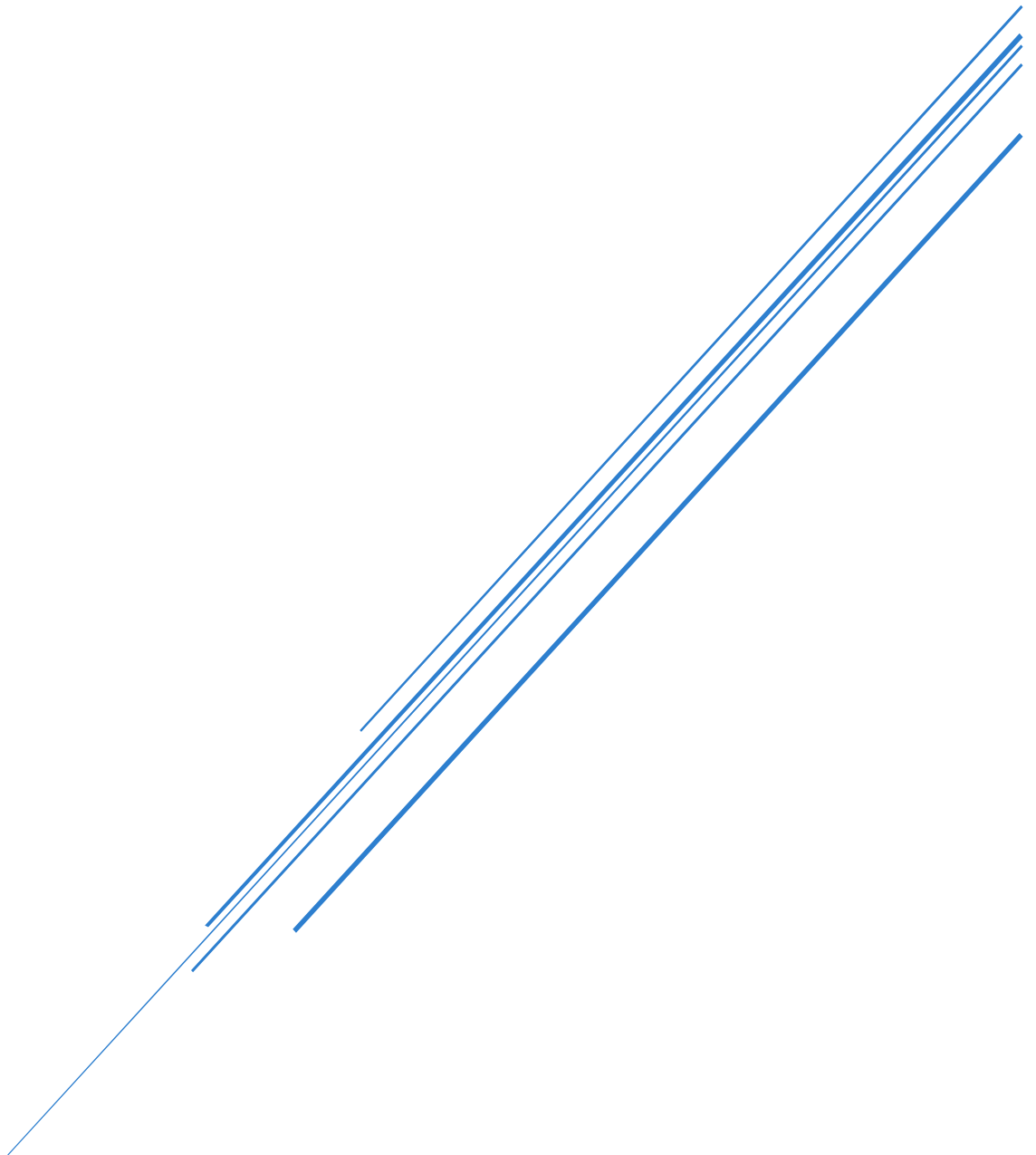


PROG6212_POE PART 1

Non-Functional Prototype



ST10371890_MALINGA_MG
Mpho Glan Malinga

Table of Contents

1. Documenting Design Choices.....	2
2. UML Class Diagram for Database.....	4
3. Breakdown of tasks for the project plan	6
4. GUI layout	7
5. ScreenShot of Github and link	11
6. In Conclusion:.....	12

Part I: Planning for the project and creating a prototype.

Summary:

The primary focus of Part 1 of the project involves the development and creation of a non-functional prototype for the Contract Monthly Claim System (CMCS). This prototype lays the groundwork for the system's functionalities in Part 2, which will incorporate dynamic elements, database integration, and live user engagements. The goal of this phase is to establish a strong base by making clear design choices, defining data structures, and creating a user-friendly interface (UI) that addresses the main functionalities of the system.

1. Documenting Design Choices

Summary of Design Choices: The CMCS was created to meet the requirements of various stakeholders, such as teachers, program coordinators, and academic managers. The decision to implement a modular and role-specific interface was driven by the need for clarity and simplicity, both crucial for users seeking quick and reliable access to claims, approvals, and tracking.

Separation of roles:

- The CMCS offers distinct dashboards tailored to each user role, guaranteeing that each user views only pertinent information and tasks.
- Lecturers can submit claims, keep track of their progress, and include additional supporting documents.
- Program Coordinators can approve or deny claims in a well-organized manner.
- Academic Managers oversee the entire process to guarantee that claims progress smoothly through the approval chain.

Limitations in design:

- The static prototype is currently nonfunctional and meant to showcase system architecture and structure without operational capabilities. The prototype lacks connectivity to a database and does not offer real-time updates. This ensures that the design is user-friendly and visually aligned with expectations before integrating functions.
- Material Design principles are utilized to ensure a consistent, professional, and responsive interface for the system. Every button, text block, and grid display a consistent color scheme and layout.

Presuppositions:

- Frequent use of the system by users such as lecturers, coordinators, and supervisors necessitates a straightforward UI design.
- Uploading Documents: Claim submissions require supporting documentation, so a user-friendly upload function is needed.
- Modularity: The system is created to allow for easy extension, ensuring that future modifications or additions (such as new approval roles or varied workflows) can be seamlessly incorporated with little to no disturbance.

2. UML Class Diagram for Database

The UML Class Diagram explains the core data structure of the system, focusing on how claims and user data will be shown. This visual is significant as it illustrates the process of mapping entities (classes) to a database in Part 2 when the features are put into action.

Key entities:

Lecturer class attributes include lecturer ID, name, surname, employee number, and contact number.

Relationships are one-to-many with claims (each professor may submit numerous claims).

Claim Class: Attributes: ClaimID, LecturerID, TotalHours, Status, and supporting documents

Relationships: Belongs to a professor; linked to one or more modules.

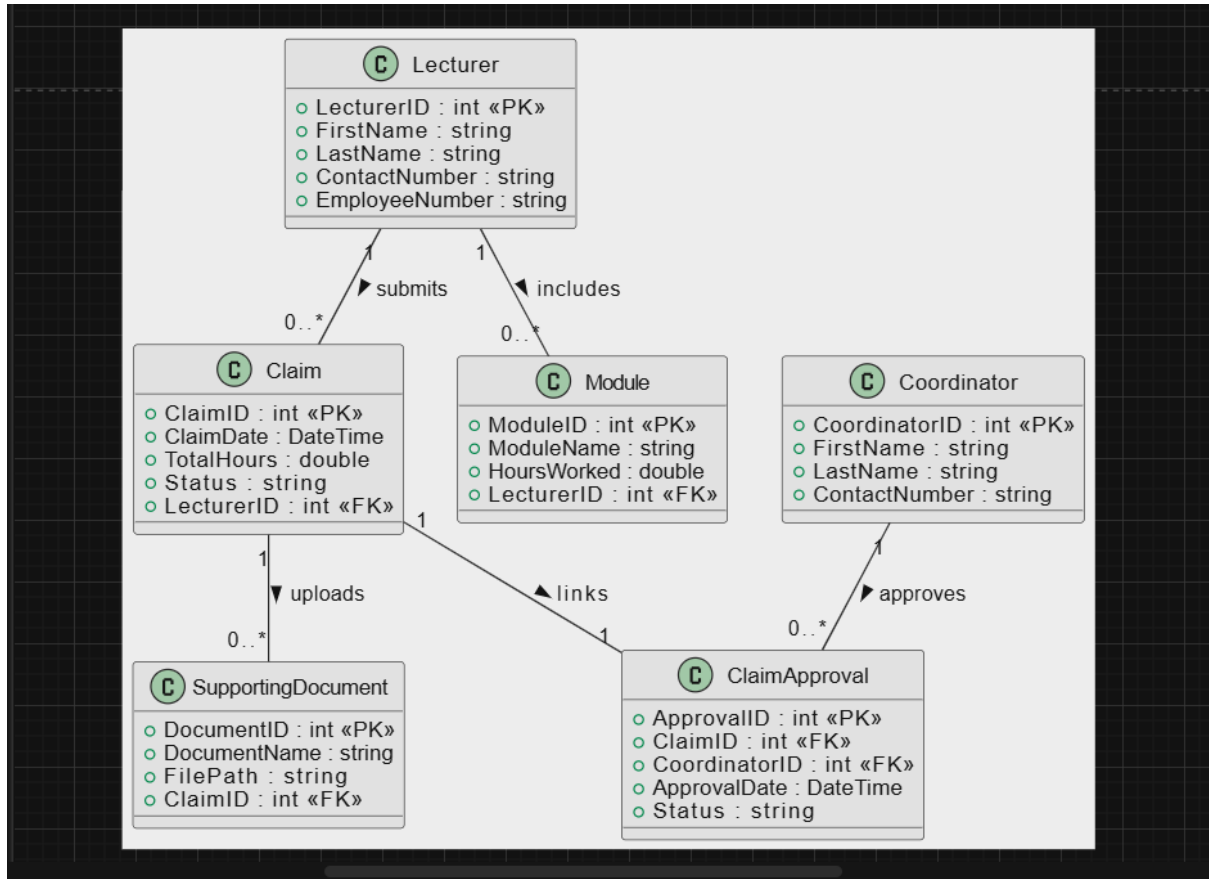
Module Class attributes include ModuleID, ClaimID, ModuleName, HoursWorked, HourlyRate, and TotalClaim.

Relationships: Each claim may include numerous modules that indicate the lecturer's instructional load.

ClaimStatus Class: Attributes: StatusID, ClaimID, Status Relationships: Shows approval status for each claim at any moment.

The structure of the UML Class Diagram.

This diagram gives a general view of the database layout and acts as the plan for creating the database schema in Part 2.



In this class diagram, every connection and attributes are explicitly specified. This arrangement will ensure a more streamlined implementation process once real data management starts in Part 2.

3. Breakdown of tasks for the project plan

Gathering and analyzing requirements (1 week):

Recognize the needs of the system and the responsibilities of each role.

Outline the layout of the user interface and the database.

Planning Stage (2 Week Duration):

Construct UML class diagrams that illustrate the database.

Develop static user interface prototypes for lecturer, coordinator, and manager user roles.

Development of the initial model (3 weeks):

Utilize WPF in combination with Material Design to develop user interface elements in XAML.

Design and organize navigation, data grids, and input fields for the user interface.

Documenting and improving (one week):

Record every design choice made.

Evaluate and enhance the user-friendliness of the static prototype.

Important achievements or stages of progress:

Finishing UML diagrams guarantees the clarity of all data connections.

Front-End Prototype Finalization: Guarantees that all graphical user interface elements are correctly organized.

Documentation completion involves recording design choices and presumptions.

4. GUI layout

The CMCS relies on the user interface (UI) as a crucial element. This non-functional prototype aims to develop a user interface that is clear and easy to navigate, serving as the basis for future interactions.

Lecturer Dashboard:

Main Features:

Ability to submit claims.

Add modules and specify hours worked.

Upload supporting documents.

The screenshot shows a web application window titled "LecturerWindow". The main heading is "Lecturer Dashboard - Submit Claim". Below this, there is a section titled "Lecturer (Personal Details)" containing four input fields: "Lecturer Name:", "Lecturer Surname:", "Employee Number:", and "Contact Number:". Below these fields is a table with four columns: "Module Name", "Hours Worked", "Hourly Rate", and "Total Claim". The table body is currently empty. Below the table, there is a section titled "Upload Supporting Documents:" with a blue button labeled "Upload Document". At the bottom of the interface, there are two buttons: "Add Another Module" and "Submit Claim".

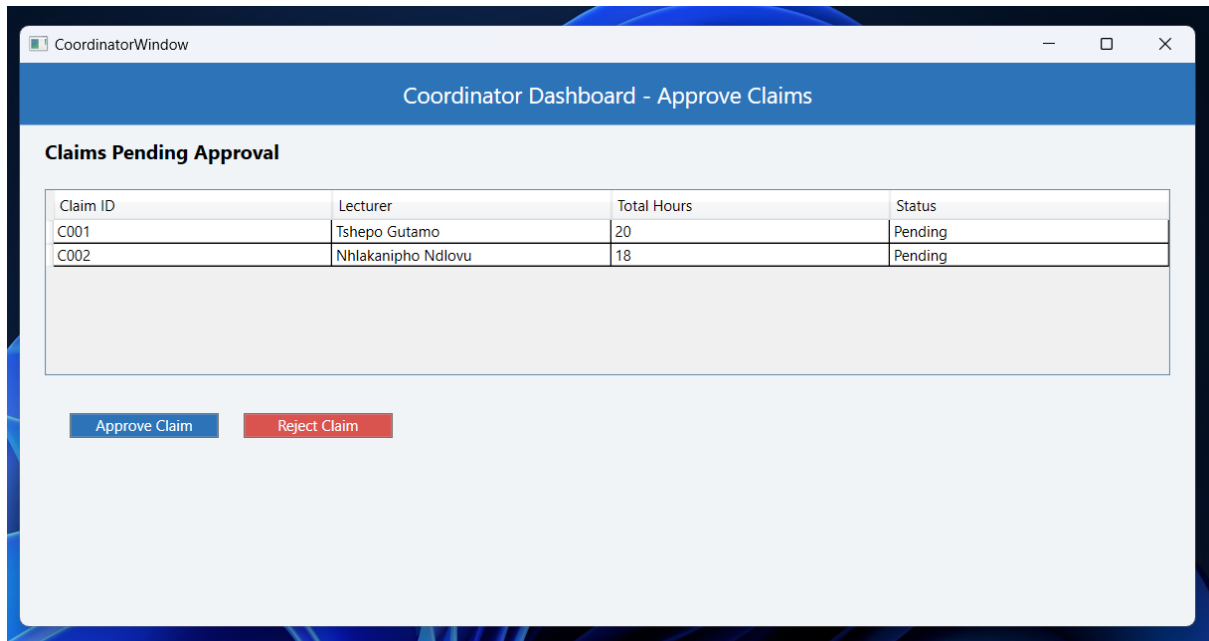
Module Name	Hours Worked	Hourly Rate	Total Claim
-------------	--------------	-------------	-------------

Coordinator Dashboard:

Main Features:

View pending claims.

Approve or reject claims.

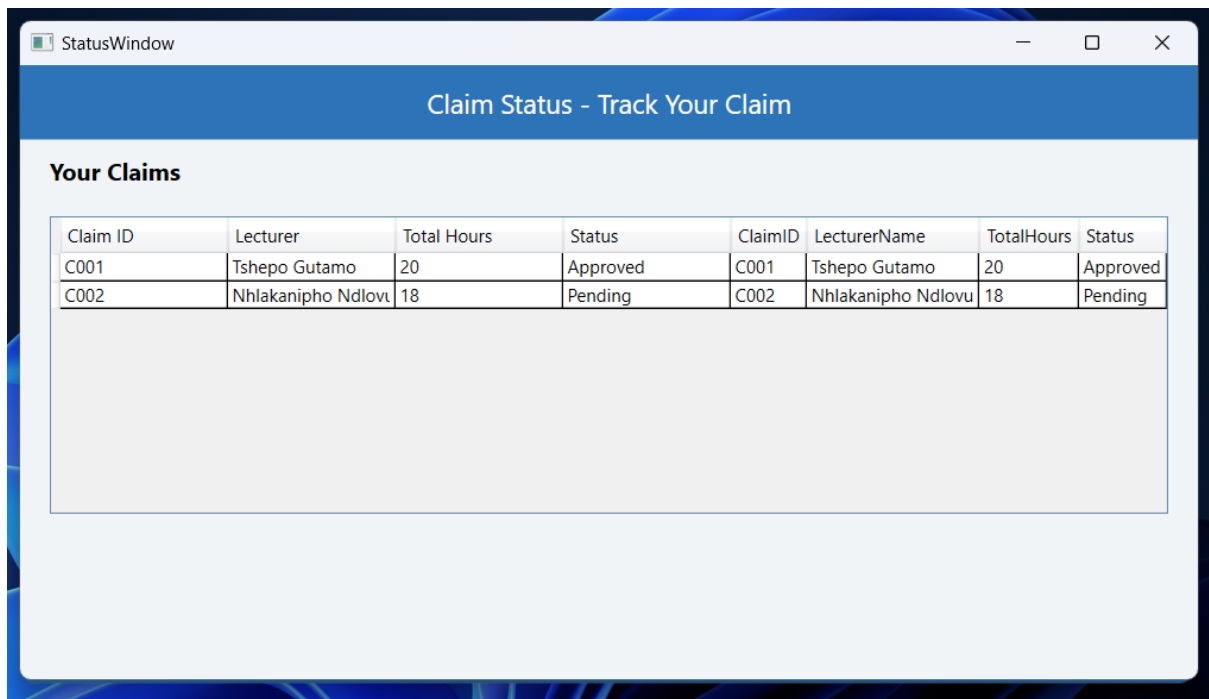


Claim Status:

Main Features:

Track claim status.

View claim history and supporting documents.



Claim Status - Track Your Claim							
Your Claims							
Claim ID	Lecturer	Total Hours	Status	ClaimID	LecturerName	TotalHours	Status
C001	Tshepo Gutamo	20	Approved	C001	Tshepo Gutamo	20	Approved
C002	Nhlakanipho Ndlovu	18	Pending	C002	Nhlakanipho Ndlovu	18	Pending

Main Navigation Window:

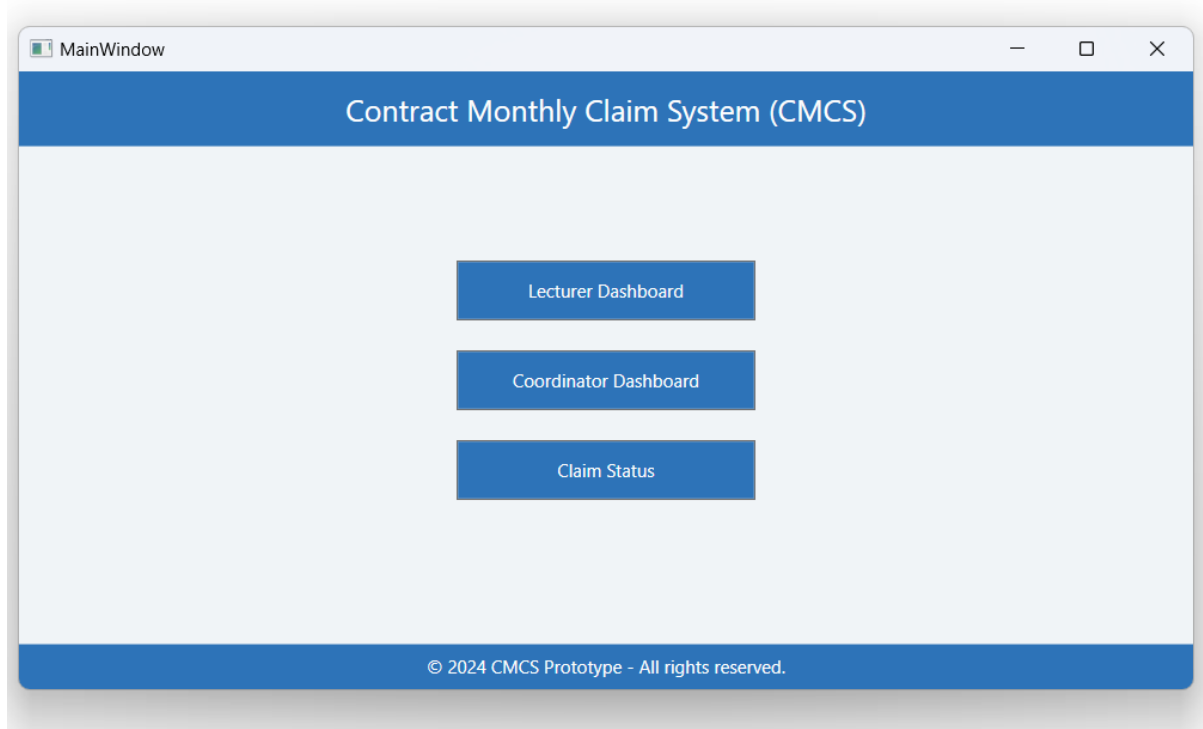
This is the entry point for all users. It contains buttons to navigate to each dashboard:

Lecturer Dashboard

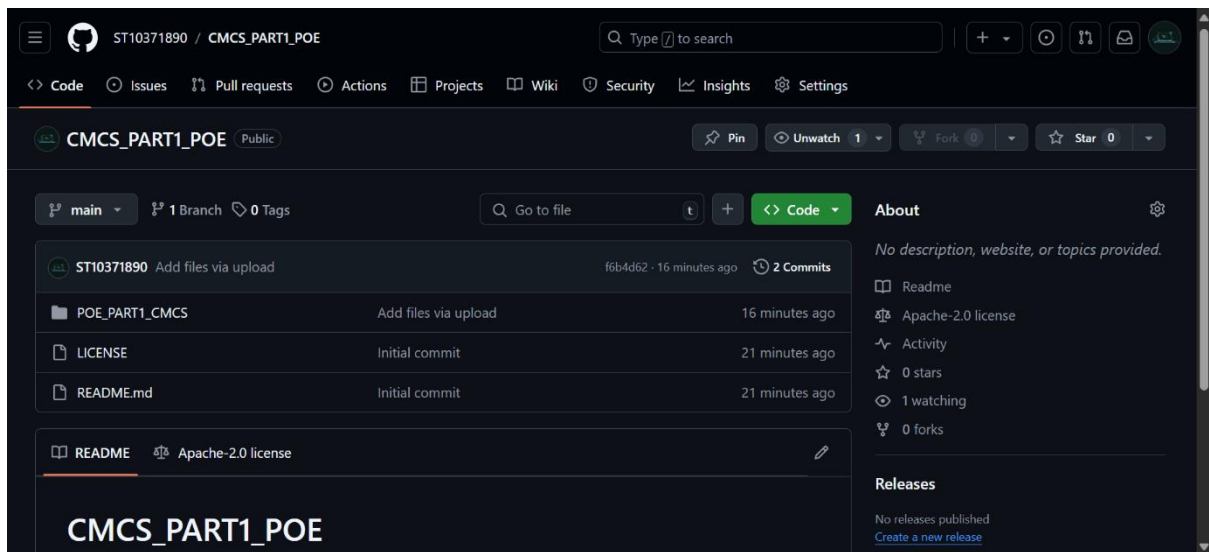
Coordinator Dashboard

Claim Status Tracker

Here's an example of the main window's screenshot:



5. ScreenShot of Github and link



[ST10371890/CMCS_PART1_POE \(github.com\)](https://github.com/ST10371890/CMCS_PART1_POE)

6. In Conclusion:

Part 1 ends with the non-working model of the CMCS, setting the groundwork for a system that works properly. The prototype clearly outlines duties, showcases a user-friendly interface, and incorporates a high-level database layout. Once all parts are assembled, this system will move forward to Part 2 to establish back-end features, link the interface with the database, and manage immediate user tasks such as submitting claims and monitoring their progress.

This careful planning guarantees that the CMCS is simple, effective, and expandable, enabling a smooth, clear process for submitting and approving claims.