Pfunzo Mulaudzi

PROG 6212 Group 2

St10442903 POE PART 1

**GITHUB REPOSITORY LINK**

<https://github.com/Mulaudzi-Pfunzo/PROG6212_POEPART1_PM.git>

**Documentation for Part 1: Project Planning and Prototype Development**

Introduction

The Contract Mbbonthly Claim System (CMCS) is developed beginning with a meticulous design phase that guarantees long-term maintainability and functional accuracy. Usability, scalability, and data integrity were taken into consideration during the design process considering the system's goal is to assist Independent Contractor lecturers, Program Coordinators, and Academic Managers in expediting the claims processing process.

Choice of Architecture and Technology

The decision to use the Model-View-Controller (MVC) architectural pattern with .NET Core was motivated by its separation of concerns namely model, view and controller, which ensures that business logic, user interface, and data access remain independent. This structure reduces the risk of tightly coupled code and makes it easier to add new functionality in later phases of development. MVC is also widely adopted in enterprise environments, making it a realistic choice that prepares students for professional practice.

Database Structure and Entity Design

The database design is centered around four primary entities: Lecturer, Claim, Document, and UserRole. Each lecturer has a unique staff number, which serves as the primary identifier. Claims are linked to lecturers and contain attributes such as hours worked, hourly rate, and claim status. Supporting documents, including timesheets or receipts, are attached to claims to provide evidence of work completed. Finally, user roles distinguish between lecturers, coordinators, and managers, ensuring that access control is both secure and role-appropriate. This structure was chosen to reflect the real-world workflow of claims submission and approval while avoiding redundancy. To extend the system beyond the minimum, additional tables such as Notification and AuditTrail were introduced. The Notification table will allow lecturers to receive updates when their claims are approved or rejected, while the AuditTrail will log all actions for accountability. Although these tables are not yet functional, their inclusion strengthens the prototype and prepares the groundwork for later automation.

User Interface Layout and Design

The design of the graphical user interface prioritizes ease of use for those who may lack technical expertise, making it professional and intuitive. The interface provides professors with an easy way to add supporting papers, submit claims using a structured form, and monitor the status of their claims. A dashboard with choices to confirm, approve, or reject pending claims is made available to academic managers and program coordinators. Even though the core functionality is not yet operational, the prototype incorporates placeholders for progress indicators and confirmation messages to give the system a more dependable feel. The design seeks to minimize errors and enhance the user experience by emphasizing clarity and cutting out superfluous complexity.

Assumptions and Constraints

Several assumptions were made during the design process. It was assumed that all lecturers would have access to a unique staff number and a valid institutional email, as these form the basis for user identification and communication. It was also assumed that the majority of claims would involve simple hourly calculations, meaning that the system does not need to handle highly complex financial formulas at this stage. In terms of constraints, the system is limited to common file formats such as PDF, DOCX, and XLSX for document uploads, as this ensures both compatibility and security. Another constraint is that the prototype does not yet connect to a live database; instead, it is intended purely as a visual representation of the final product.

Conclusion

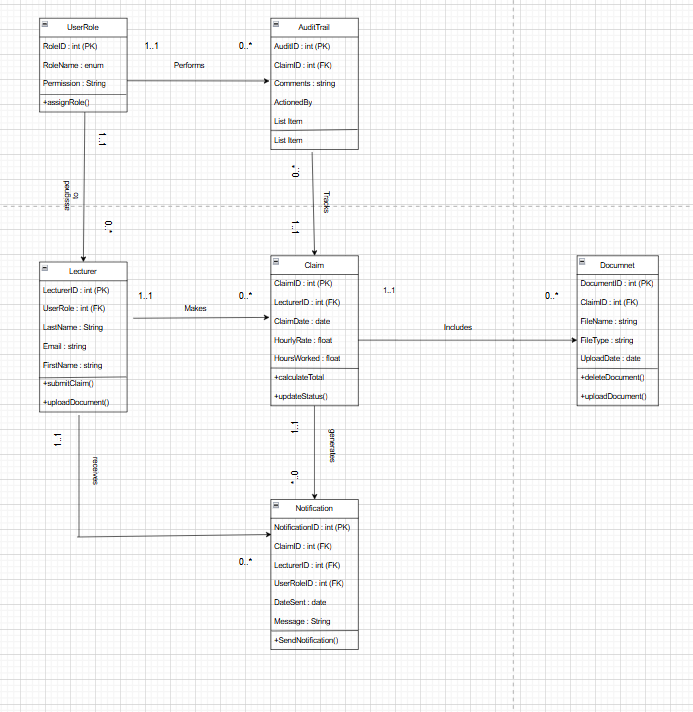
The overall objective of these decisions is to create a system that is both realistically feasible and rigorously academic. Early implementation of features like role-based access and notifications helps the project avoid the need for expensive redesigns later on. However, restricting the range of computations and file types keeps the burden reasonable and ensures that coding aspects in Part 2 stay realistic. In the end, the documentation strikes a compromise between practical execution and forward-thinking design.

**Project Plan for Part 1 – Contract Monthly Claim System (CMCS)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Phase | Task | Dependency | Duration | Timeline (Sept 2025) | Milestone |
| 1: Preparation | Review POE brief & rubric | None | 1 day | 13 Sept | Project scope confirmed |
| 1: Preparation | Set up GitHub repo & initial commit | None | 0.5 day | 13 Sept | Version control baseline |
| 2: Documentation & Design | Draft design choices (architecture, DB, GUI layout) | Rubric review | 1 day | 13–14 Sept | Draft documentation |
| 2: Documentation & Design | Define assumptions & constraints | Draft documentation | 0.5 day | 14 Sept | Constraints finalised |
| 2: Documentation & Design | Create UML class diagram | Design rationale complete | 1 day | 14–15 Sept | UML completed |
| 3: Prototype Planning | Define Lecturer view layout | UML complete | 0.5 day | 15 Sept | GUI layout approved |
| 3: Prototype Planning | Define Coordinator/Manager dashboard layout | UML complete | 0.5 day | 15 Sept | Dashboard layout approved |
| 3: Prototype Planning | Plan placeholders for claim submission, document upload, and claim tracking | UML complete | 0.5 day | 15 Sept | GUI blueprint complete |
| 4: Implementation (Prototype only) | Create MVC skeleton in Visual Studio | UML + GUI layouts | 0.5 day | 16 Sept | Project skeleton committed |
| 4: Implementation (Prototype only) | Add non-functional Lecturer view | Skeleton ready | 0.5 day | 16 Sept | Lecturer view prototype |
| 4: Implementation (Prototype only) | Add non-functional Coordinator/Manager view | Skeleton ready | 0.5 day | 16 Sept | Coordinator view prototype |
| 4: Implementation (Prototype only) | Add placeholders for file upload & claim status tracking | Skeleton ready | 0.5 day | 16 Sept | GUI placeholder complete |
| 5: Review & Refinement | Review prototype usability & design | Prototype complete | 0.5 day | 16 Sept | Internal review complete |
| 5: Review & Refinement | Insert UML & GUI screenshots into report | Prototype complete | 0.5 day | 16 Sept | Report draft ready |
| 5: Review & Refinement | Proofread & finalise Word doc submission | Report draft ready | 0.5 day | 17 Sept | Final submission |
| 5: Review & Refinement | Final GitHub push with documentation & project | All tasks complete | 0.5 day | 17 Sept | Submission complete |

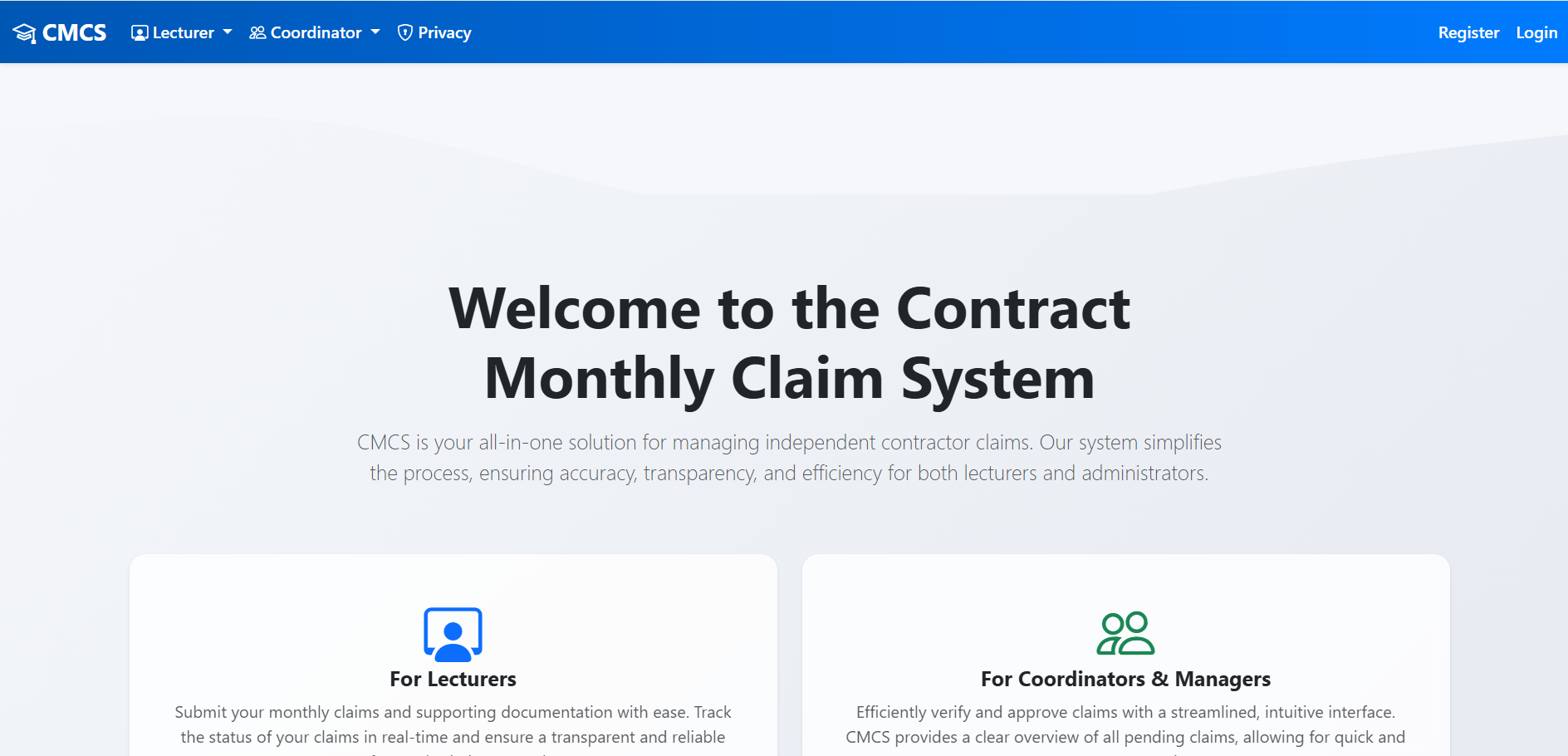
**UML Diagram**

(NOTE: Another separate document will be submitted on ARC for improved visibility for the following diagram)



**Screenshots of GUI**

**Home Page Screenshot**

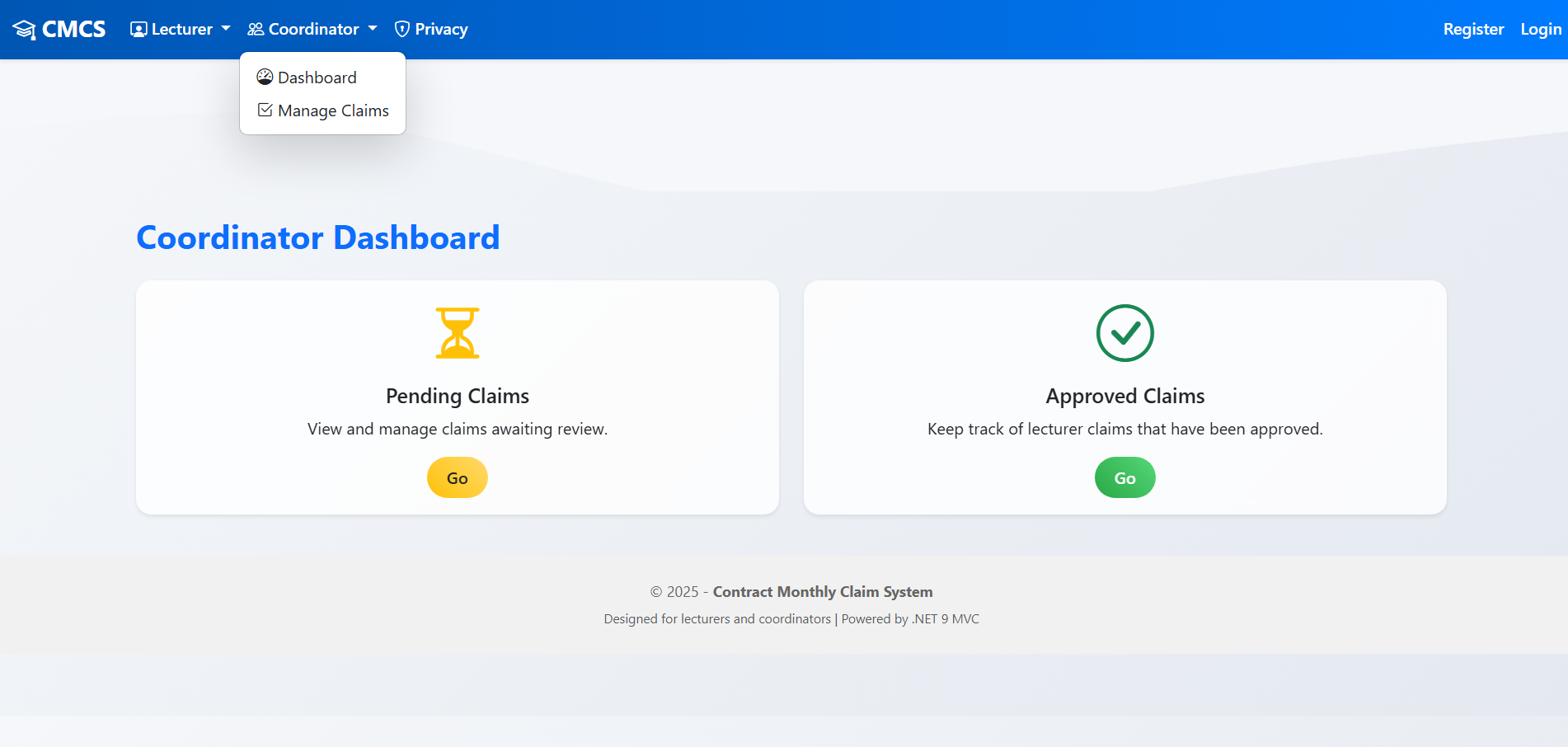


**Lecturer Dashboard screenshot**

A screenshot of a computer

AI-generated content may be incorrect.

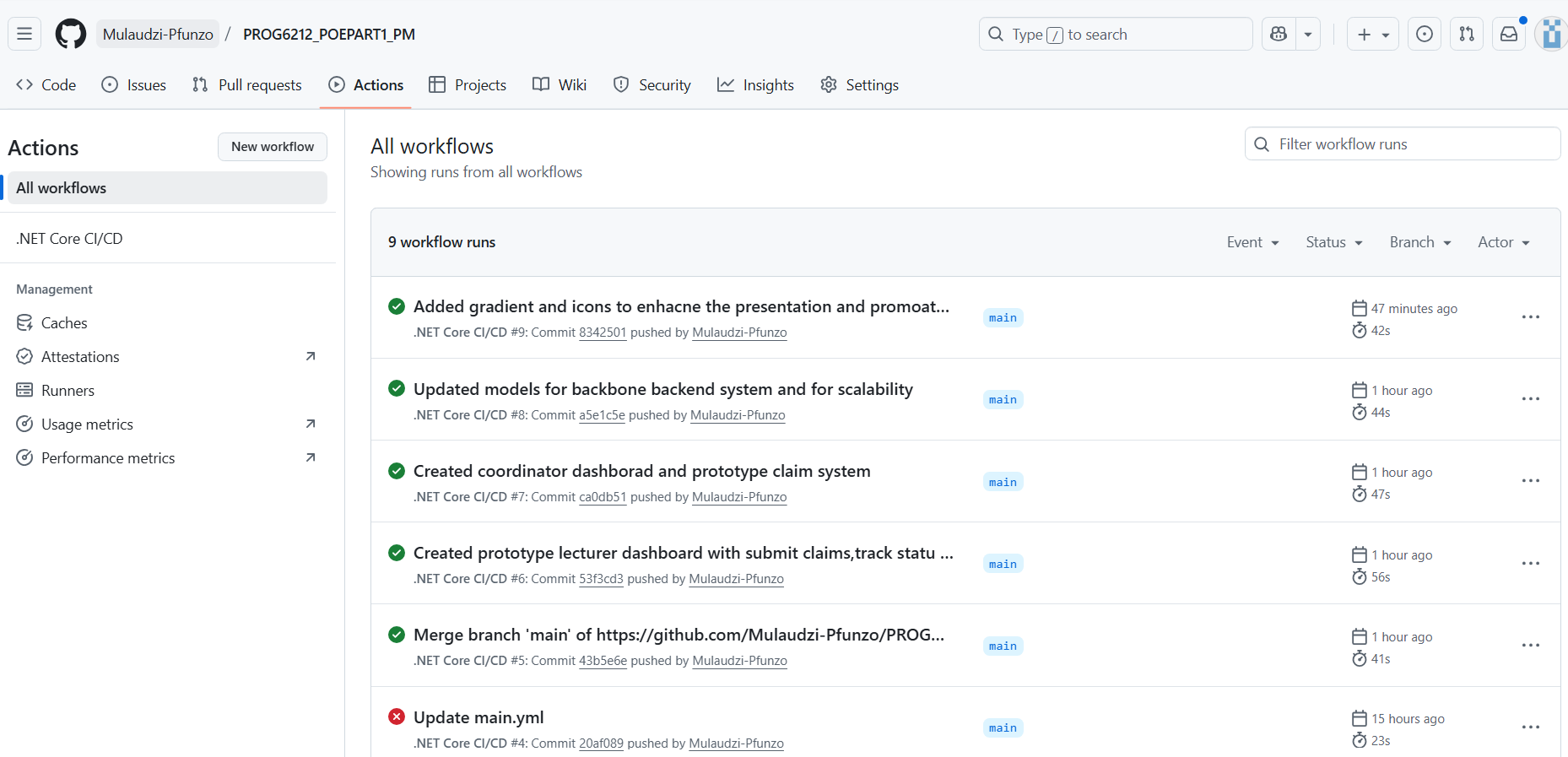
**Coordinator Dashboard screenshot**



**Privacy Policy screenshot**  
A screenshot of a computer

AI-generated content may be incorrect.

**GitHub actions screenshot**



**Reference List**

Microsoft. (2025). *ASP.NET Core Documentation*. Available at: <https://docs.microsoft.com/en-us/aspnet/core/> (Accessed: 17 September 2025).

Freeman, E., Robson, E. and Robson, P. (2025). *Head First C#*. 5th ed. O'Reilly Media.

Gemini was consulted for clarity on project plan requirements as seen below

