SStT1-049485675

[Company name]  [Company address]

Contract Monthly Claim System (CMCS) – Prototype (POE Part 1)

**Table of Contents**

1. Introduction (Background & Objectives)  
........................................................................... 1

2. Design Documentation  
..................................................................................................... 2

Design Choices and GUI Layout .......................................................................................... 2

Data Model and Database Structure ..................................................................................... 2

Assumptions and Constraints ............................................................................................... 2

Project Plan Overview ......................................................................................................... 2

Conclusion ........................................................................................................................... 2

3. UML Class Diagram (Database Model) ...........................................................................

Entities and Attributes .......................................................................................................... 3

Relationships ........................................................................................................................ 3

Diagram ................................................................................................................................ 3

Explanation .......................................................................................................................... 3

4. Project Plan......................................................................................................................... 4

5 GUI Screenshots (Prototype Evidence)........................................................................... 5

Dashboard ............................................................................................................................ 5

Submit Claim form ............................................................................................................... 5

Verify Claims (table with Approve buttons) ......................................................................... 5

Track Claims ........................................................................................................................ 5

Login & Register .................................................................................................................. 5

6. Version Control Evidence (GitHub) ................................................................................. 6

7. Conclusion ........................................................................................................................ 7

8.Reference list........................................................................................................................ 8

**1. Introduction (Background & Objectives)**

The Contract Monthly Claim System, or CMCS, that's the prototype web app they built. It came from the Portfolio of Evidence, you know, for the PROG6212 module. The whole idea is to make submitting monthly claims simpler for independent contractor lecturers. Verifying and approving them without the usual hassle. In practice, it involves lots of people, like the lecturers, program coordinators, and academic managers. You have to check hours worked, rates, and all that supporting paperwork real carefully.

This POE aims to show some understanding of C# GUI in the .NET Core environment. Applying theory to an actual project. Right now, it's a non-functional prototype. It focuses on design, layout, and the overall flow. Not the back-end or database stuff yet. Still, it gives a solid idea of how it'd work in reality. Makes the UI intuitive and user-friendly, pretty much. The scope for Part 1 of the POE includes. **Planning and documentation**, including design explanations, assumptions, and constraints.

* A **UML class diagram** that models the database structure for claims, users, and supporting documents.
* A **project plan** outlining the tasks, dependencies, and timeline for prototype development.
* A **graphical user interface (GUI) prototype**, developed in ASP.NET Core MVC, which demonstrates the key system workflows: claim submission by lecturers, verification by programme coordinators, approval by academic managers, and the ability to track claim status transparently.

Finishing up this first stage, you know, it sets things up nicely for the rest of the POE. Later on, we'll get into adding those functional features, hooking up the database, and putting in the authentication stuff. I mean, the main thing right now is just making sure the design is clear and documented pretty well. That way, the whole final system ends up being efficient, focused on the users, and it lines up with how the academic claim management process is supposed to go.

**2. Design Documentation**

The setup for this whole Contract Monthly Claim System prototype, or CMCS for short, basically boils down to keeping things simple for lecturers dealing with their claims. You know, a straightforward interface that matches whatever role the person has. Right now it's just a mock-up. Nothing actually functions yet. It's all about demonstrating the look and feel before they implement the real features in later phases.

Design Choices and GUI Layout

They picked ASP.NET Core MVC to build this. It seems like a good choice since it separates concerns well, scales nicely, and works great for web applications. There's a shared layout file called \_Layout.cshtml that ensures a consistent appearance across all pages, including navigation and branding elements. At the top, a navigation bar allows quick access to the dashboard, submitting claims, verifying them, approving, tracking progress, and account management like login or signup.

The dashboard serves as the main entry point. Users view their assigned role there, such as lecturer, programme coordinator, or academic manager, and access relevant functions. Lecturers complete a form to submit claims, entering details like the month, hours worked, hourly rate, and a placeholder for supporting documents. In the verification section, claims appear in a table for coordinators to review details and approve. Academic managers perform a similar process for final approval. The tracking feature enables lecturers to check the status of their claims, promoting transparency throughout the process.

Data Model and Database Structure

The prototype features a single Claim model with essential properties including Id, LecturerName, ClaimMonth, HoursWorked, HourlyRate, Status, and SupportingDocument. This provides the minimum structure required to represent a claim.

In the full implementation, this model maps to a Claims table in the database. Additional elements include a Users table for lecturers and administrators with role-based access control, a SupportingDocuments table to store file information and paths for evidence uploads, and a ClaimStatusHistory table to log changes for auditing purposes. The UML class diagram in the accompanying report illustrates these components and their relationships.

Assumptions and Constraints

Several assumptions underpin this prototype. Authentication and role management are absent for now but will be added in subsequent phases. Document uploads are simulated using a text input field rather than actual file handling. Approvals are indicated through display messages without any database updates.

Constraints limit the scope to front-end development in this initial phase, excluding back-end data persistence or integration. This approach allows focus on usability, layout design, and clear workflow presentation for different roles.

Project Plan Overview

The project timeline spans three weeks. Week one involved requirements gathering, UML diagramming, and establishing the MVC framework. During the second week, the dashboard and claim-related views were developed, along with sample data integration. The final week focused on UI refinements, documentation preparation, and submission readiness, including the UML diagram, project plan, and report in Word format. Version control was maintained through at least five GitHub commits, each accompanied by descriptive notes outlining the changes.

This stage sets up a good base for what's next. With the GUI clear, model in place, and roadmap sketched, the prototype shows the claim flow in a way that's structured but not overwhelming. When they layer on the functions later, it'll hit the marks for tech and ease of use.

**3. UML Class Diagram (Database Model)**

The database design for the Contract Monthly Claim System (CMCS) is represented using a **UML class diagram**, which models the core entities, their attributes, and relationships. This diagram provides a blueprint for how data will be structured and stored once database functionality is introduced in later stages of the project.

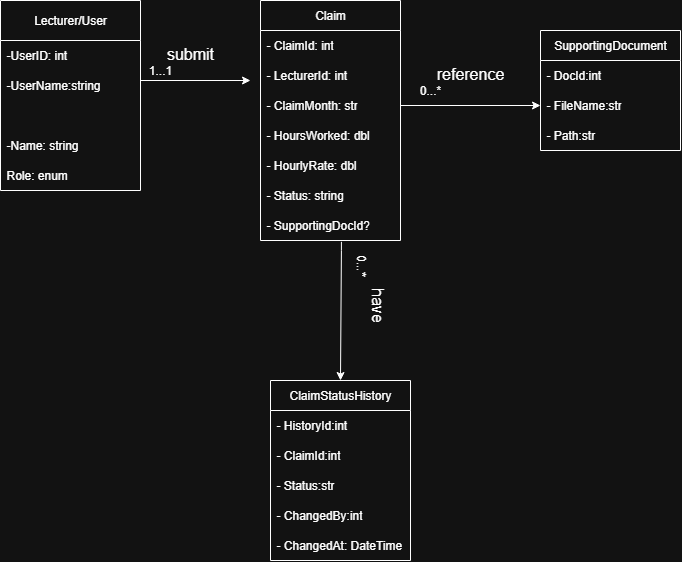
**Entities and Attributes**

* **User (Lecturer/Staff):** Represents individuals who interact with the system. Key attributes include UserId, Username, Password, FullName, and Role. Roles are defined as Lecturer, Programme Coordinator, or Academic Manager, each with specific permissions.
* **Claim:** Stores claim details submitted by lecturers. Attributes include ClaimId, LecturerId (foreign key to User), ClaimMonth, HoursWorked, HourlyRate, Status, and SupportingDocId.
* **SupportingDocument:** Stores metadata about uploaded files, such as DocId, FileName, and FilePath. Each claim may have one or more associated supporting documents.
* **ClaimStatusHistory:** Records the history of status changes for each claim. Attributes include HistoryId, ClaimId, Status, ChangedBy, and ChangedAt. This ensures an audit trail is maintained.

**Relationships**

* A **User** can submit multiple **Claims**, but each claim is linked to only one user.
* A **Claim** may reference zero or more **SupportingDocuments**, depending on whether evidence is required.
* A **Claim** can multiple entries in the **ClaimStatusHistory**, reflecting its progression through the workflow (e.g., Pending → Verified → Approved).

**Diagram**

*Figure 1: UML Class Diagram for CMCS Database Schema*

**Explanation**

The diagram lays out the logical setup for the database. It helps track and handle claims in a solid way. You know, by breaking everything down into users, claims, documents, and history. That kind of arrangement keeps the data reliable. It allows for full audits. And it scales without a ton of problems.

Take the ClaimStatusHistory entity for example. It logs every action that happens on a claim. That gives you real transparency. Oh and, putting supporting documents in their own separate area avoids all that duplication mess. File management gets a lot easier, basically.

This structure fits right in with the MVC prototype we're building now. The Claim class we have matches the Claims entity in the diagram. We'll add in things like User and SupportingDocument entities later on. You know, during those next development phases. So the prototype ends up connecting to a full working back-end.

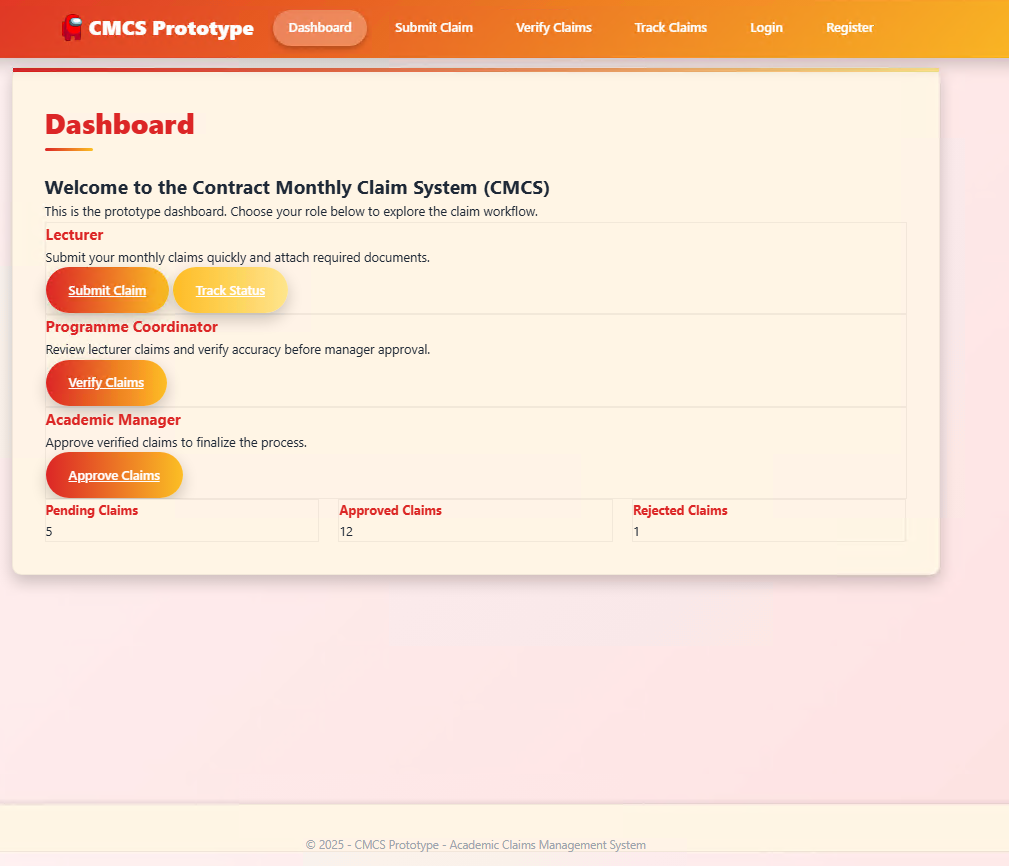
**4. Project Plan**

The project plan for the Contract Monthly Claim System (CMCS) prototype was structured over a three-week period to cover design, development, documentation, and submission requirements. Tasks were broken down into manageable units, with dependencies identified to ensure logical progression.

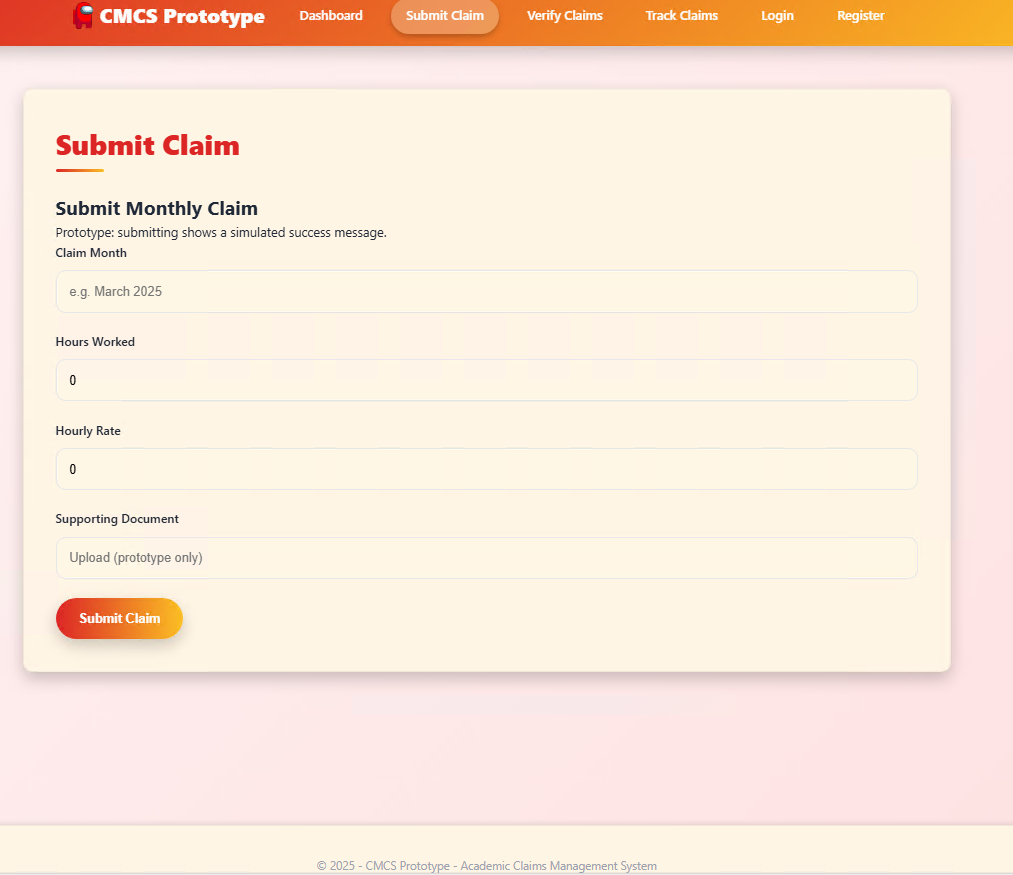
| Week | Task | Dependencies | Duration |
| --- | --- | --- | --- |
| Week 1 | Gather requirements and identify assumptions | – | 1 day |
| Week 1 | Draft UML class diagram (initial design) | Requirements identified | 1 day |
| Week 1 | Create MVC project skeleton and layout page | – | 2 days |
| Week 2 | Implement Dashboard, Submit, Verify, and Track views | Layout completed | 3 days |
| Week 2 | Define Claim model and seed sample data | UML draft completed | 2 days |
| Week 2 | Apply CSS styling and ensure responsive layout | Core views in place | 2 days |
| Week 3 | Finalise UML diagram and refine design documentation | Prototype complete | 2 days |
| Week 3 | Prepare project plan and report (Word document) | Documentation drafted | 2 days |
| Week 3 | Push at least 5 commits to GitHub with descriptive messages | Development tasks completed | 1 day |

**5. GUI Screenshots (Prototype Evidence)**

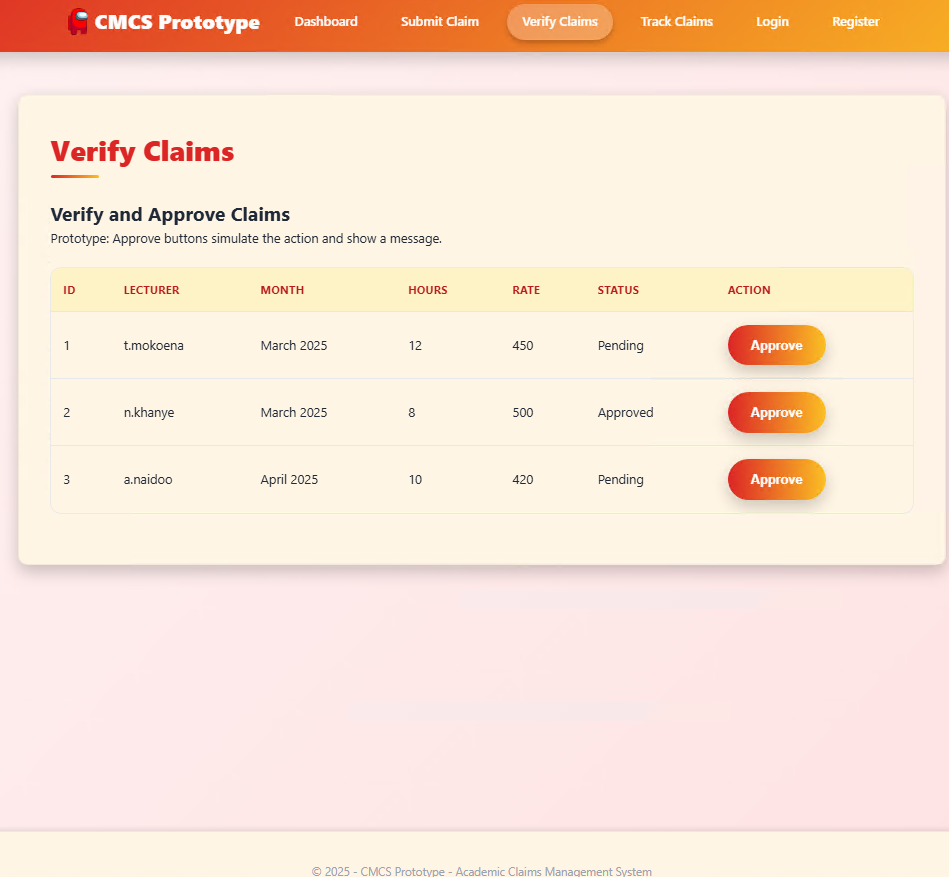
Dashboard



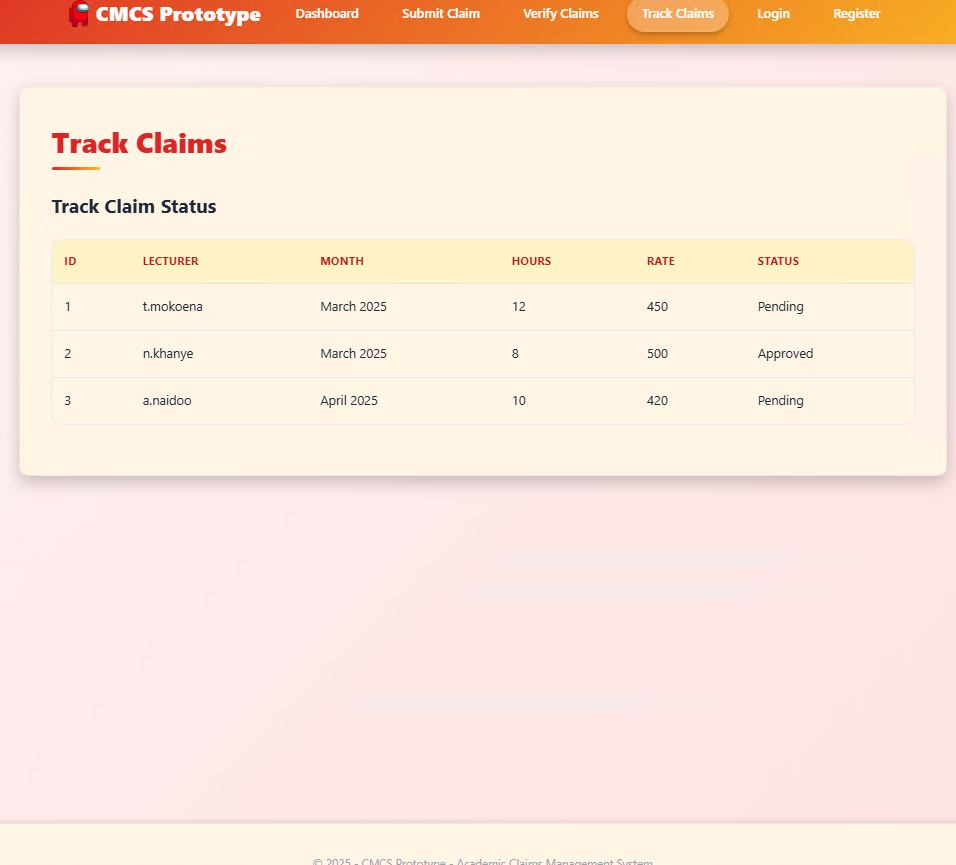
Submit Claim



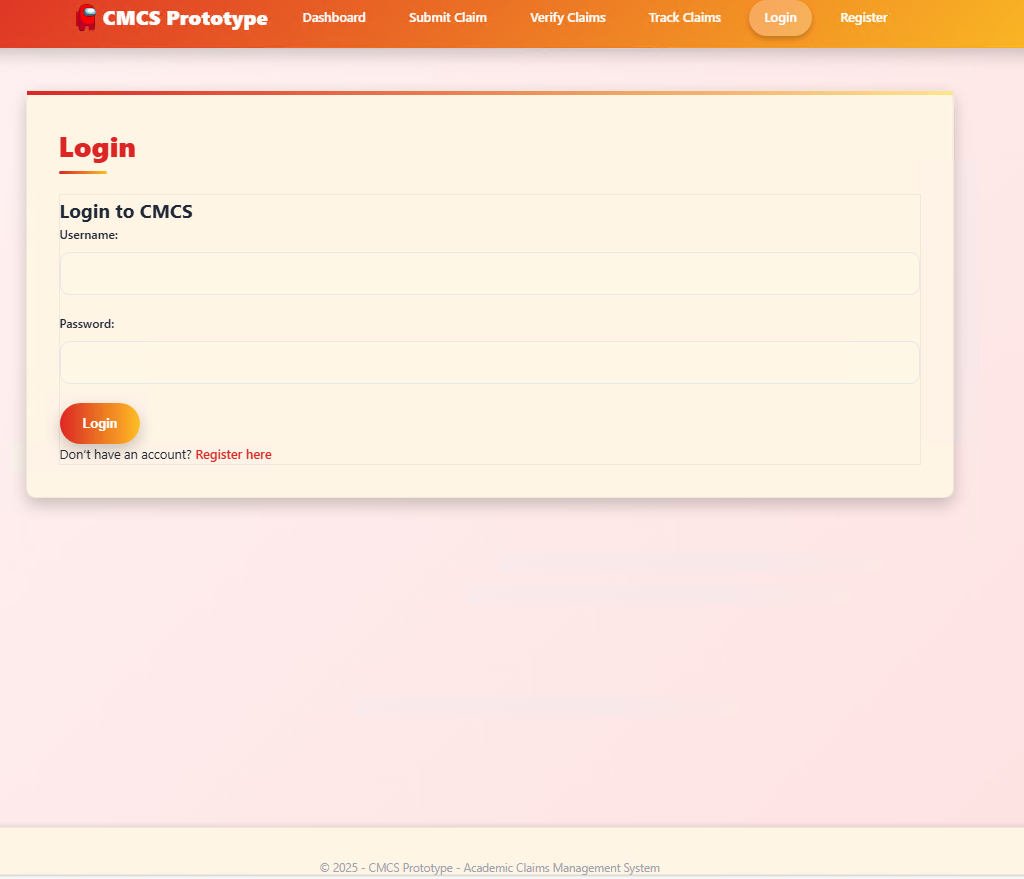
Verify Claims (table with Approve buttons)



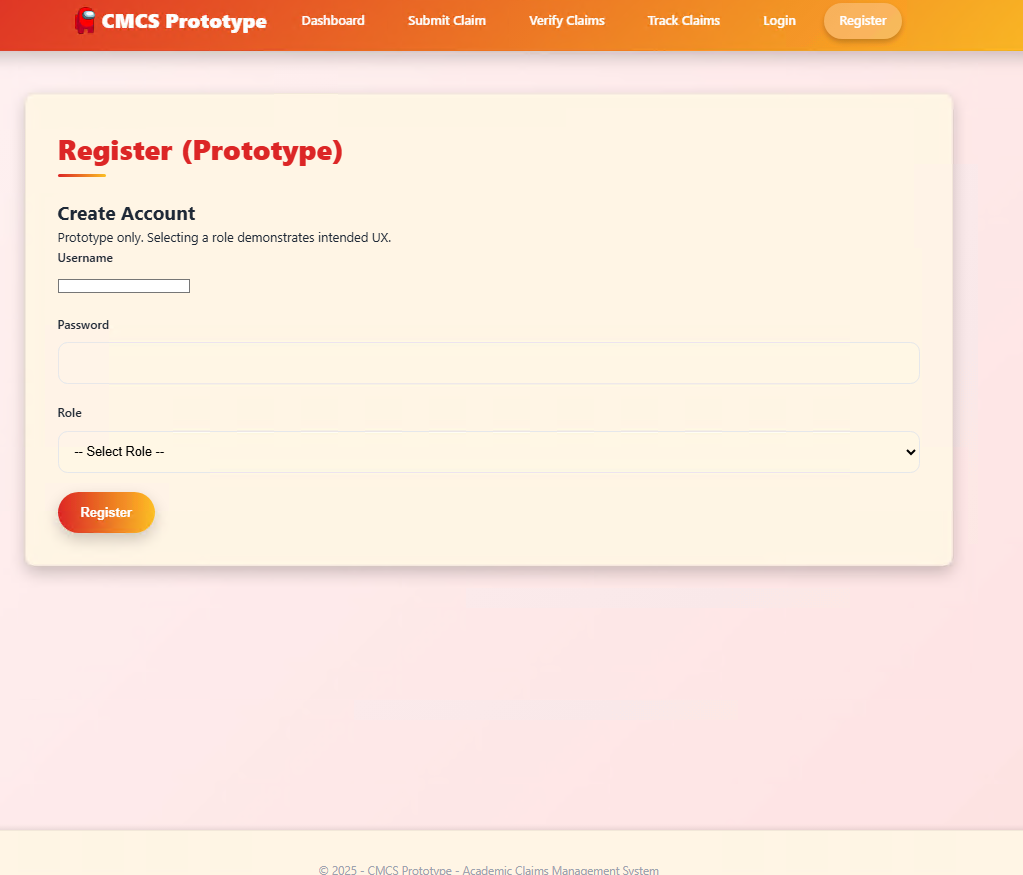
Track Claims



Login



Register



**6. Version Control Evidence (GitHub)**

* + Screenshot of commit history (at least 5 commits, with descriptive messages).

**7. Conclusion**

The Contract Monthly Claim System prototype. That's for Part 1 of the Portfolio of Evidence. It lays out the planning and design work, you know, and gets the front-end going for how the app should function. We've hit all the deliverables in here. Like, we documented the reasons behind our choices. Put together a UML class diagram for the database structure. Sketched a project plan that actually makes sense. And built a basic GUI prototype with ASP.NET Core MVC.

This setup follows the workflows just about as we planned them. Lecturers can submit claims. Coordinators review them. Managers approve or reject. Oh and there's a tracking area, so lecturers see the status of their submissions all along the way. Right now, it's not fully operational. I mean, approvals are fake. Document uploads are dummies. No real database storage happening. Still, it establishes the foundation. For when we implement the actual features in Part 2.

We've put a ton of emphasis on usability. And navigation that's clear based on user roles. The UML diagram and project plan guide everything technically. With a schedule that fits. We used version control the whole time too. Lots of commits on GitHub. They show the build-up step by step.

Design and planning are complete at this point. So yeah, the project's set for Part 2. We'll add the back-end components there. Login functionality. Real data saving. Proper file handling. That phase makes it functional for real. Aligning with the overall goal of simplifying claims and approvals for lecturers.

**8 Reference list**

Draw.io (2024). *Flowchart Maker & Online Diagram Software*. [online] app.diagrams.net. Available at: https://app.diagrams.net.

Freeman, A. (2013). *Pro ASP.NET MVC 5*. Berkeley, Ca Apress.

Ramotion (2023). *MVC Web App Architecture | Ramotion Agency*. [online] Web Design, UI/UX, Branding, and App Development Blog. Available at: https://www.ramotion.com/blog/mvc-architecture-in-web-application.

Smith, S. (2024). *Overview of ASP.NET Core MVC*. [online] Microsoft.com. Available at: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-9.0.

Tutorialsteacher (2025). *ASP.NET MVC Tutorials*. [online] www.tutorialsteacher.com. Available at: https://www.tutorialsteacher.com/mvc.