1. Documentation

**1. Design Choices**

The Contract Monthly Claim System (CMCS) is designed with a clear focus on usability, efficiency, and scalability. Below are the primary considerations for the design:

* **Modular MVC Architecture**: The Model-View-Controller (MVC) design pattern was chosen for the CMCS prototype as it allows clear separation of concerns. By separating the data handling (Model), user interface (View), and control logic (Controller), the system remains scalable and maintainable. The system is built using .NET Core MVC, which provides robust features for both small and large-scale applications.
* **User-Centric Interface**: The user interface (UI) is designed to be intuitive and easy to navigate, ensuring that users (lecturers, coordinators, and academic managers) can interact with the system without extensive training. The UI focuses on:
  + **Simplified Forms**: Forms are clean and minimalistic, with only necessary fields for claim submission and verification to avoid overwhelming users.
  + **Consistent Navigation**: A navigation bar is included on every page to facilitate easy movement between different parts of the system, such as submitting a claim, tracking status, or approving claims.

**2. Database Structure**

The database design revolves around storing lecturer claims, supporting documents, and the approval process. The key entities identified are **Lecturers**, **Claims**, **Coordinators**, **ClaimDocuments**, and **ClaimStatus**. The relationships between these entities are designed to ensure data consistency and easy retrieval.

* **Lecturers Table**: Stores information about the lecturers who will submit claims. This table includes fields for the lecturer's name, email, and hourly rate.
* **Claims Table**: This table stores the claims submitted by lecturers. Each claim includes details about the number of hours worked, the status of the claim (pending, approved, rejected), and the date the claim was submitted. Each claim is linked to a lecturer and a coordinator.
* **Coordinators Table**: Stores information about Programme Coordinators and Academic Managers responsible for approving or rejecting claims.
* **ClaimDocuments Table**: This table stores any supporting documents uploaded by lecturers when submitting a claim (e.g., payslips, contracts). The documents are linked to their respective claims.
* **ClaimStatus Table**: Tracks the history of each claim’s status, such as pending, approved, or rejected, providing transparency to the process.

**Key Considerations**:

* The **primary keys** for each table ensure the uniqueness of each record.
* **Foreign key relationships** are used to link claims to the lecturers who submitted them and the coordinators who are responsible for approval.
* **Enum values** (e.g., 'Pending', 'Approved', 'Rejected') are used to simplify the tracking of claim statuses.

**3. Layout of the GUI**

The layout for the GUI is simple yet functional, focusing on user-friendliness and accessibility.

* **Lecturer Claim Submission Form**:
  + The claim submission form consists of fields for the number of hours worked, additional notes, and the option to upload supporting documents. This ensures that lecturers can easily submit all necessary information in a single form.
  + The form includes a prominent "Submit Claim" button, designed to make the submission process straightforward.
* **Coordinator and Academic Manager Claim Approval View**:
  + This page shows a table listing all the pending claims. Each row displays the claim ID, lecturer name, hours worked, and status, with "Approve" and "Reject" buttons for each claim.
  + The layout is designed to be clear and easy to use, allowing coordinators to quickly approve or reject claims without navigating away from the table.
* **Claim Tracking Page**:
  + The claim tracking page allows lecturers to enter a claim ID to view the current status (pending, approved, rejected) of their claim.
  + The status is updated in real-time as claims move through the approval process.

**Visual Consistency**:

* A navigation bar at the top of each page provides consistent access to different parts of the system.
* The forms and tables are designed with clear labels and adequate spacing to enhance readability and usability.

**4. Assumptions and Constraints**

* **Assumptions**:
  + It is assumed that lecturers will only submit claims once a month and that all claims must be verified by a coordinator or academic manager before they can be approved.
  + It is assumed that each lecturer has a fixed hourly rate, which can be pre-filled in the claim form based on their profile in future iterations.
  + The database schema assumes that each claim is tied to a single lecturer and can have multiple supporting documents.
* **Constraints**:
  + The system is currently designed as a **non-functional prototype**, so while the layout and design are present, no backend logic or database interaction has been implemented at this stage.
  + Only the essential fields for claim submission, approval, and tracking have been included. Additional features (such as notifications or advanced filtering) may be added in future iterations.
  + The prototype is built to support **basic CRUD operations** in the next phase, and any complex workflows (e.g., multi-level approval or escalations) would need further development.

**Rationale Behind Design Decisions**

* **MVC Architecture**: MVC was chosen because it allows scalability and easy maintenance. As the system grows, more functionality can be added without reworking the core architecture.
* **User-Friendly Design**: The interface is designed to minimize complexity, especially for non-technical users like lecturers. Simplifying the claim submission and tracking processes improves overall user experience.
* **Clear Data Relationships**: The database is designed to minimize redundancy and ensure that claims, documents, and status updates can be easily tracked and referenced.
* **Assumptions and Constraints**: Assumptions about user roles and data flow guided the design, while constraints such as keeping the GUI non-functional at this stage ensure that the scope remains achievable.

1. UML Class Diagram

A screenshot of a computer

Description automatically generated

1. Project Plan:

**Project Plan for CMCS Prototype Development**

**Total Duration**: 2 weeks (14 days)

**Phase 1: Project Setup & Requirements Gathering (Day 1-2)**

**Task 1.1: Set up development environment**

* Install required software/tools (Visual Studio, GitHub, SQL Database, MySQL Workbench)
* Set up a GitHub repository for version control

**Task 1.2: Gather requirements**

* Understand the requirements for CMCS (Lecturers, Coordinators, and Claim submission flow)
* Clarify roles and features (Lecturer claims, Coordinator approval, Document upload, Claim tracking)

**Deliverable**: Environment setup and project requirements gathered  
**Dependencies**: None

**Phase 2: Database Design & UML Diagram (Day 3-4)**

**Task 2.1: Design Database Schema**

* Define entities (Lecturers, Claims, Coordinators, ClaimDocuments, ClaimStatus)
* Determine the relationships between entities

**Task 2.2: Create UML Class Diagram**

* Create the UML diagram to represent the database structure
* Include classes, attributes, and relationships

**Deliverable**: Completed UML class diagram and database schema  
**Dependencies**: Task 1.2 (Requirements gathering)

**Phase 3: User Interface Design (Day 5-7)**

**Task 3.1: Design the GUI Layout (Non-Functional Prototype)**

* Design user-friendly interfaces for the following:
  + Lecturer’s claim submission form
  + Coordinator/Academic Manager’s claim approval form
  + Claim tracking and document upload section

**Task 3.2: Create Wireframes for Review**

* Develop wireframes for each user interface using design tools (e.g., Figma, Adobe XD, or directly in Visual Studio)

**Deliverable**: GUI wireframes (non-functional) and finalized layout design  
**Dependencies**: Task 1.2 (Requirements gathering), Task 2.1 (Database Schema)

**Phase 4: Prototype Implementation (Day 8-12)**

**Task 4.1: Implement the Non-Functional Prototype**

* Set up the .NET Core MVC or Windows Presentation Forms project
* Create static pages/forms for the Lecturer and Coordinator
  + Lecturer claim submission form
  + Coordinator/Academic Manager view for claim approval
  + Upload document form
  + Track claim status page

**Task 4.2: Connect UI Components**

* Ensure that the UI forms and views are connected (without backend functionality)
* Incorporate basic navigation between pages

**Deliverable**: A fully designed non-functional front-end prototype  
**Dependencies**: Task 3.1 (GUI design)

**Phase 5: Project Review & Documentation (Day 13-14)**

**Task 5.1: Write Documentation**

* Document design choices (GUI layout, database design, assumptions, constraints)
* Describe how the database structure supports the application logic

**Task 5.2: Review and Refine**

* Review the entire prototype (UI design, navigation, and layout)
* Ensure all requirements are met and prepare for submission

**Task 5.3: Finalize Version Control**

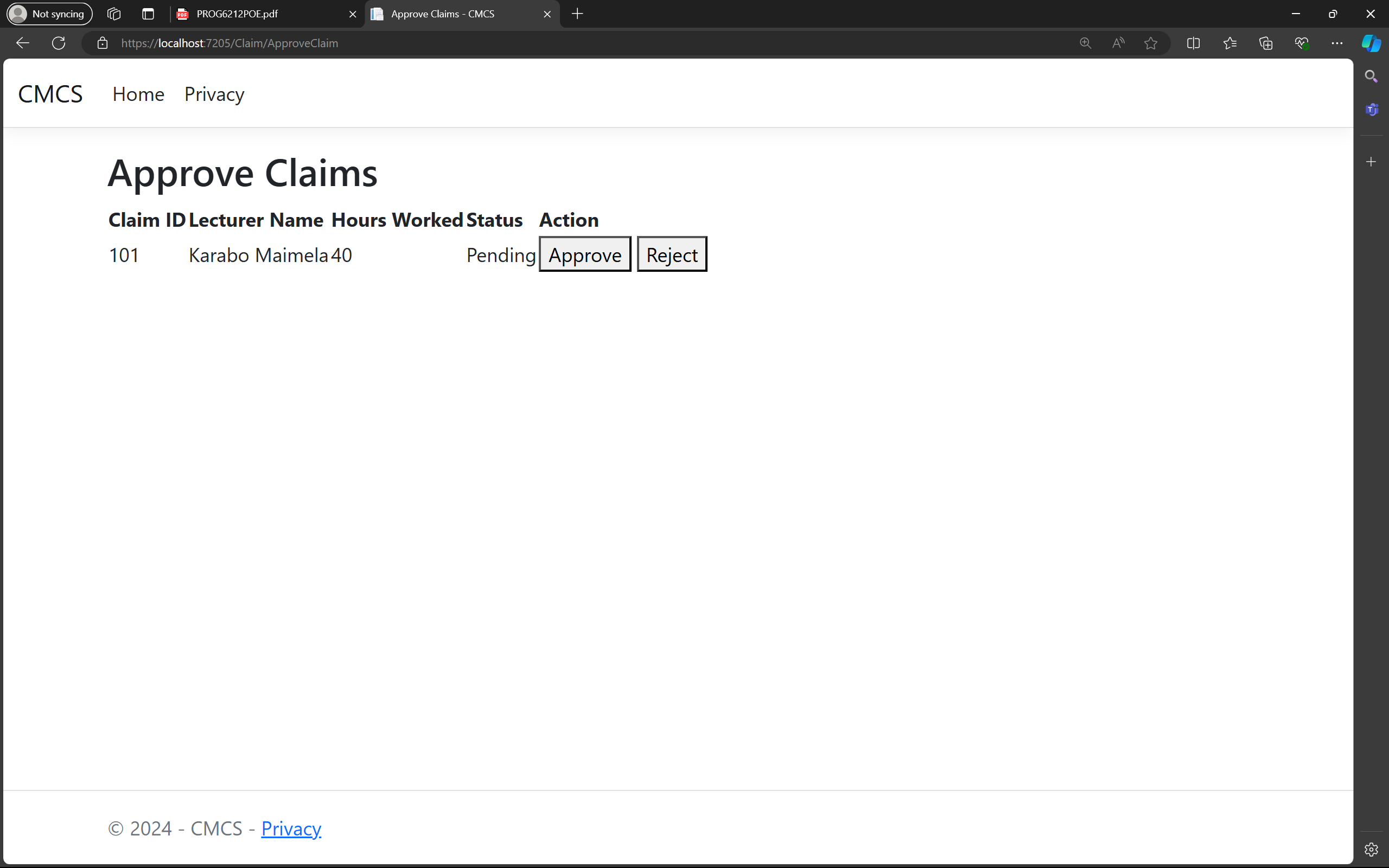
* Commit changes to GitHub (at least 5 commits with descriptive messages)

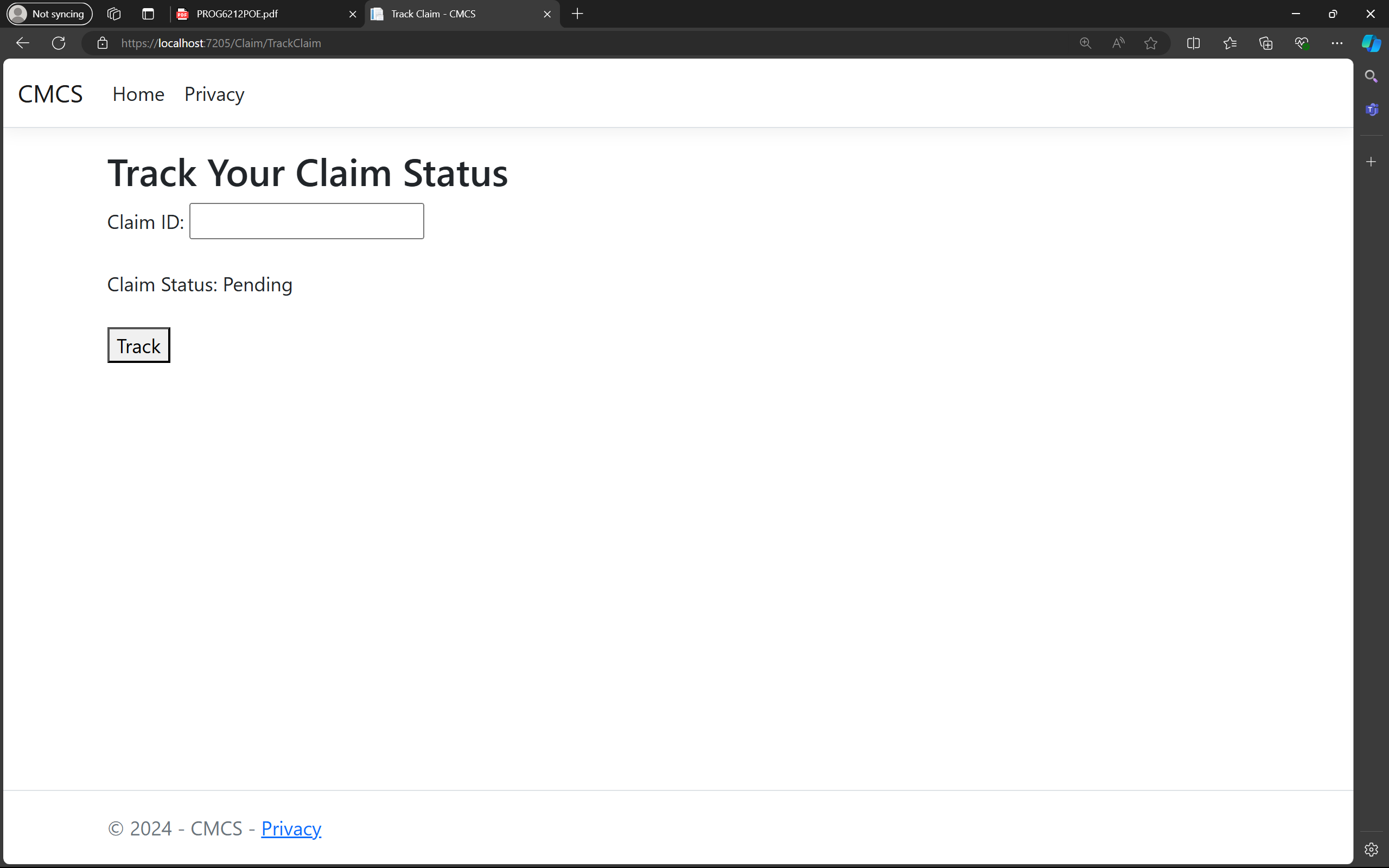
**Deliverable**: Final prototype with complete documentation and version control  
**Dependencies**: Task 4.1 (Prototype Implementation)

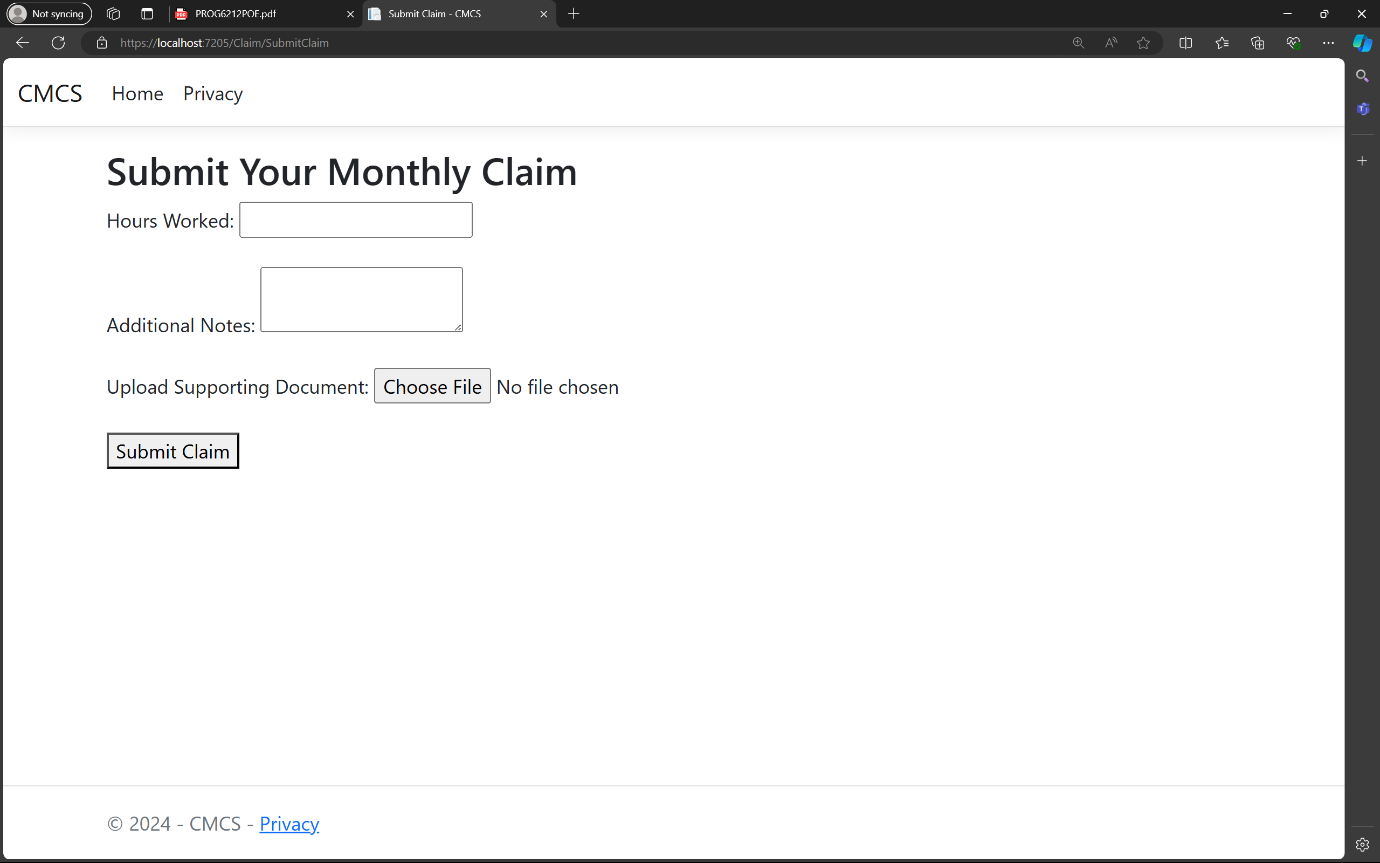
**Timeline Summary**

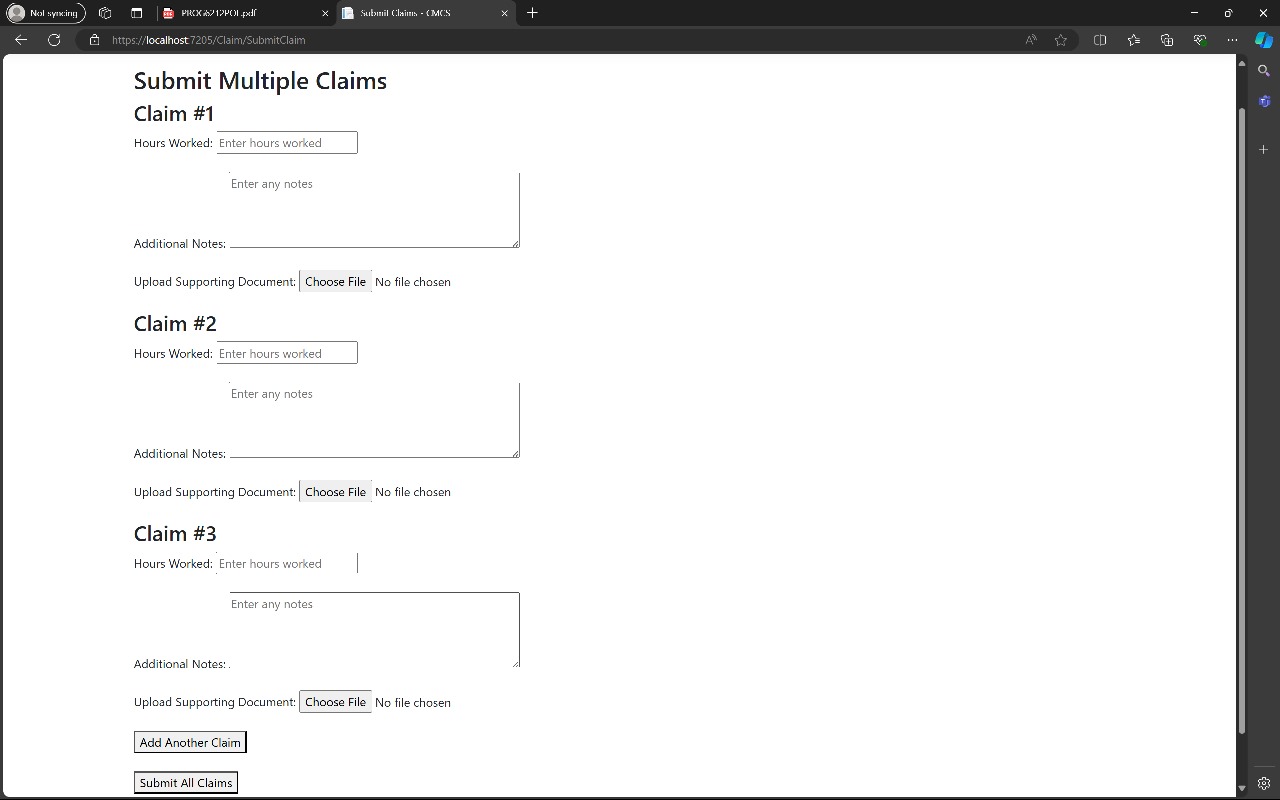
|  |  |  |  |
| --- | --- | --- | --- |
| Task | Duration | Start | End |
| Phase 1: Setup & Requirements | 2 days | Day 1 | Day 2 |
| Phase 2: Database Design & UML | 2 days | Day 3 | Day 4 |
| Phase 3: GUI Design | 3 days | Day 5 | Day 7 |
| Phase 4: Prototype Implementation | 5 days | Day 8 | Day 12 |
| Phase 5: Review & Documentation | 2 days | Day 13 | Day 14 |

1. GUI IU:









REFERENCE

* How to upload files/folders/projects on github /upload project folder on github(simple way)– ved The Master. 2009. YouTube video, added by ved The Master. [Online]. Available at: <https://youtu.be/eGaImwD8fPQ?si=OOYN8sDMqF6SbCjy> [Accessed 02 september 2024].
* C# GUI / C# GUI PROJECT / C# GUI Tutorial/ Learn C#/ C# Programming Tutorial/ simplilearn –Simplilearn. 2022. YouTube video, added by Simplilearn [Online]. Available at: <https://youtu.be/UMc1HLyBHSQ?si=s23woiyQZJ1ejOyc> [Accessed 02 september 2024.
* How to write a project plan [project planning steps that work – **Adriana Girdler**. 2020. YouTube video, added by **Adriana Girdler**. [Online]. Available at: <https://youtu.be/Lt9_4vzPdlo?si=AqDlWCF9q_PCaPGV> [Accessed 06 september 2024].
* UML tutorial how to draw uml class diagram – Wondershare Edraw. 2021. YouTube video, added by Wondershare Edraw. [Online]. Available at: <https://youtu.be/ao1ESgIy2Ws?si=W9OCt6Q0ZqdQouOQ> [Accessed 06 september 2024].