**ST10254797**

**Cristiano Naidoo**

**PROG 6212 POE Part 1**

The CCMS system is meant to help lectures submit their claims, which are then reviewed and approved or declined by the Programme Coordinators and Academic Managers. Being a prototype of the intended application, the database structure is designed to prioritise simplicity and clarity in representing important items.

**Table/Entities:**

Lecture Profile (This is the Profile for the lectures): This tables contains fields such as Lecture\_ID, Hourly\_Rate, Total\_Hours\_Worked, Total\_Claims\_Submitted, Lecturer Name, Lecturer Email and an Image.

Submit Claims (This is the table that is used to submit the claims of the lectures): This table contains fields such as Claim\_ID, Lecture\_ID, Total\_Hours\_Worked, Hourly\_Rate, Claim\_Amount and uploading documents

Track Claims (This is the table that the lectures use to track their claims): This table contains fields such as: Claim\_ID, Submitted\_Date, Status and Total Amount

Review Claims (This is where the Programme Coordinators and Academic Managers review the claims from the lectures): This table contains fields such as Lecture Name, Hours worked, Hourly Rate, Total.

Approve Claims (This is where the Programme Coordinators and Academic Managers approves the claims from the lectures): This table contains fields such as Claim\_ID, Lecture Name, Status.

**Relationships:**

Each lecturer may submit several claims, resulting in a one-to-many link between Lecturer and Claim.

Each Lecturer can track multiple claims, resulting in a one to many relationships between Lecture and Track Claims

Each claim can have their claim status individually tracked in the track table, resulting in a one-to-one link between Submit Claims and Track Claims.

Each claim can be reviewed exactly once, resulting in a one-to-one link between Submit Claims and Review Claims.

Each claim is approved or declined exactly once, resulting in a one-to-one link between Submit Claims and Approve Claims.

**Design Rationale:**

The database structure separates concern, making it easy to manage lecturer data, claim submissions, and tracking claims independently.

Reason for Field Selection:

Lecture Profile Table: Storing Total\_Hours\_Worked and Total\_Claims\_Submitted in the Lecturer Profile database optimises efficiency by eliminating the need to calculate them dynamically for each query. By saving them, the system can instantly present an overview of a lecturer's claims and work history without having to reprocess past data.

Submit Claims Table: To make sure that historical records of claims accurately reflect the rate at the time the claim was made, the Hourly\_Rate is included in the Submit Claims table. Inaccurate financial data would result from rate changes impacting all prior claims if the Hourly\_Rate and Lecturer Profile were dynamically linked.

Managing Further Growth: Because of the structure's scalability, new features like claim history and document versioning can be added without requiring a redesign of the main database. To offer a more thorough tracking procedure, new claim statuses like "Under Review" or "Rejected" could be added to the Track Claims table. A ClaimHistory database could keep track of each status change over time in order to implement claim history. Comparably, by enlarging the Submit Claims table to monitor numerous document uploads with version numbers, document versioning can be implemented.

The structure is designed to be efficiency, accuracy, and scalability, so additional features like claim history, document versioning can be added later without restructuring.

(Mortier. 2023)

**GUI Layout:**

Home Page:

Allows access to main functions for various jobs (lecturers, coordinators, and managers). The design is straightforward, with a navbar that simply leads users to their respective activities.

Lecture Profile Page:

Allows access to view profile details such as Name, Email, Total Hours worked, Hourly Rate and Total Claims submitted. This makes sure professors can easily navigate the system and have a comprehensive picture of their profile, which helps to clear up any misconceptions.

Lecturer's Submit Claim Page:

Features a clear and straightforward form for Lectures to submit their claims.

Key fields such as Total Hours Worked, Hourly Rate, and Claim Amount are disabled to display calculated numbers, indicating real-world limits on these variables. This is vital for ensuring data integrity because the system calculates these numbers based on the lecturer's profile and claim details.

A separate File contribute option is available for lecturers to contribute several supporting materials.

Buttons such as Submit Claim are disabled in this non-functional prototype, but they are included to replicate actual procedures

Lecture Track Claim Page:

Allows the Lecture to check the process of the claims that has been made, shows the Claim ID, Submitted Date, Status and Total Amount. The table style improves the user experience by making claim history simple to read and scan at a glance, particularly when dealing with several claims. It also visually organises material in an organised manner, allowing professors to rapidly examine the status of their claims without having to scroll through many pages.

Coordinator and Manager pages:

Placeholder pages are given to allow Coordinators to review claims and Managers to approve them.

Future editions will add filters and status indicators to monitor claims' development at each level.

Design rationale:

Consistency: The layout uses the same theme throughout all pages, including recognisable Bootstrap components for form inputs, navigation, and buttons. This allows users to simply browse the program.

On the Submit Claim Page, the fields Total\_Hours\_Worked and Claim\_Amount are hidden to maintain data integrity and usability. These numbers are generated using data already present in the lecturer's profile, limiting the possibility of user input errors. Disabling these variables guarantees that financial calculations are accurate and consistent. The decision to show these fields rather than allow for manual entry simplifies the user interface and reduces the likelihood of user confusion or manipulation of critical variables.

On the Track Claim Page, using a table style to present claim history is straightforward and organised. Tables provide a familiar interface for most users, allowing lecturers to compare claims, check progress, and determine the status of each submission. This strategy increases efficiency by displaying all relevant information in one location, reducing the need for users to switch between different displays. Furthermore, the table architecture is scalable, which means that future upgrades like as pagination, filtering, or sorting can be implemented without disturbing the user experience.

Accessibility: The Bootstrap framework guarantees that the app is responsive and useable on a variety of devices and screen sizes. The adoption of Bootstrap also corresponds with technological restrictions, as it lowers the need for bespoke CSS and speeds up development while preserving a high-quality user experience.

Scalability: The form is intended to be easily extensible, allowing new features (such as extra notes or supplementary documentation) to be added without requiring significant reorganisation. The separation of concerns between distinct pages (such as submission, tracking, and review) allows for the seamless integration of new features or roles as the system matures.

(Microsoft. 2024)

**Assumptions and Constraints:**

Assumptions:

Non-Functional Prototype: This version of the CMCS lacks back-end logic and database connectivity, despite the presence of forms and UI components. The purpose is to demonstrate a proof of concept for the user flow.

Simple Workflow: During this stage of development, the procedure is linear. Lecturers submit claims, which are then reviewed by Coordinators and approved by Managers. This prototype does not include any sophisticated procedures, such as rejections or appeals.

File provide Limitations: The system permits several files to be uploaded for each claim, however lecturers are expected to only provide relevant material (e.g., time sheets, contracts). At the moment, no file validation (such as format or size constraints) has been implemented.

Constraints:

Technological constraints:

The system's design and layout are built with Bootstrap, which limits customisation but provides responsiveness and simplicity of use.

This version lacks back-end logic and database connectivity, limiting its ability to show real-world claim processing, authentication, and approval routines.

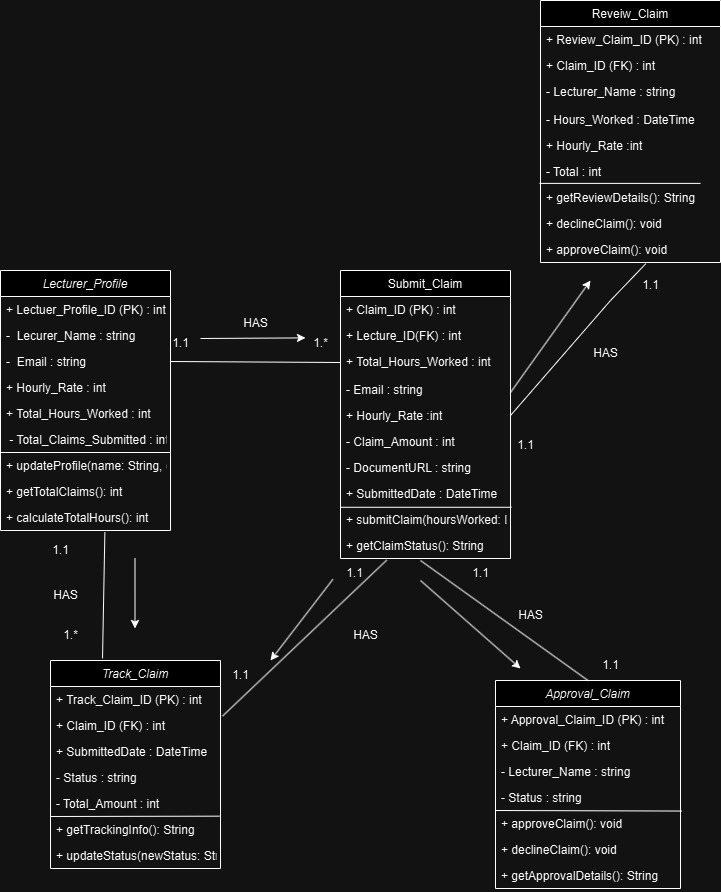
User Roles: The system has three main roles: lecturer, coordinator, and manager. Each position gets access to particular areas of the application. The limits are that this prototype does not provide user authentication or role-based access control.

Document Upload Handling: File uploads are intended to be basic, with no size or type inspection; however, future iterations may need to handle file validation (for example, accepting just PDFs or picture formats).

The present design strikes a compromise between simplicity (to provide a seamless user experience) and scalability (to allow for future improvements). The database and GUI design prioritise clarity and simplicity of navigation, which is critical for users with varied degrees of technical knowledge.

(Zight. 2019)

2) **UML DIAGRAMS**



3) **Project Planning:**

1)Project initiation

Tasks: Establish project scope and objectives.

Establish the project team's roles and responsibilities.

Set up project management tools.

Dependencies: none.

Timeline: two days (September 1-2)

2) Requirements Gathering and Analysis

Task: Gather and record comprehensive needs from stakeholders. (lecturers, program coordinators, academic managers)

Analyse and finalize both functional requirements (e.g., claim submission, approval process) and non-functional requirements (e.g., usability, performance).

Dependencies: Project initialisation is now complete.

Timeline: two days (September 3 to September 4).

3) Database design tasks include creating schema (tables and relationships) for claims, instructor profiles, and tracking purposes.

Choose the best fields, data types, and indexes for maximum performance.

Create a cloud database for development and testing (such as Azure SQL Database).

Dependencies include the gathering and analysis of requirements.

Timeframe: three days (September 5-7).

4) UI Design: Create wireframes and mockups for various screens (e.g. profile, claim filing, claim tracking).

Review user interface designs with stakeholders and iterate depending on their comments.

Dependencies: Database design is now complete.

Timeline: two days (September 8-9).

5) Frontend Development tasks include implementing UI screens according to design specifications.

Create client-side functionality (such as form validation and dynamic content).

Integrate the UI with backend services. (e.g., claim submission, file upload, status tracking).

Dependencies: The completion of the UI design.

Timeline: fiva day (September 10-14).

6) Backend Development:

6.1 API Development Tasks: Implement RESTful API endpoints for critical functionality such as claim filing, status tracking, and document upload.

Create user roles and authentication (e.g., JWT, OAuth, or session-based).

Implement CRUD operations for lecturers' profiles and claims.

Dependencies include database design.

Timeframe: 5 days (September 15–19).

6.2 Business Logic Implementation Tasks: Implement claim processing logic, including calculating claim amounts based on hours worked and rate.

Implement role-based access for claim evaluation and approval (Program Coordinators and Academic Managers).

Handle file uploads and validation (size and type) of accompanying documents.

Dependencies: API development.

Timeframe: 5 days (September 20–24).

6.3 Database Integration Tasks: Connect the API to the database to store and retrieve claim, lecturer, and document data.

Optimise database searches for efficiency.

Dependencies: API development.

7) Backend testing

Tasks: unit and integration testing on API endpoints.

Test the claim acceptance process and confirm that the business logic is correct.

Run stress tests on the database for claim submissions and retrievals.

Dependencies: Backend development.

Timeline: three days (September 28-30).

8) Frontend-Backend Integration:

Tasks: Connect the frontend UI to the backend API endpoints.

Test the data flow between frontend forms (claim submission, tracking) and backend services.

Ensure that the frontend appropriately reflects backend status modifications (for example, claim status changes).

Dependencies include backend development and frontend completion.

Timeline: three days (October 1-3).

9) Prototype Testing

Task: Test the entire system from beginning to end.

Perform usability testing on the UI/UX flow to ensure that it is easy to understand for lecturers and managers.

Test different workflows (submitting a claim, tracking claim status, and the approval procedure).

Validate file upload functionality and handle errors (such as invalid file formats).

Dependencies include frontend and backend integration.

Timeline: three days (October 4–6)

10) Security implementation

Task: Implementing user authentication for lecturers, coordinators, and managers with Azure AD or equivalent services.

Secure API endpoints (e.g., JWT tokens, HTTPS).

Perform security tests (SQL injection and authentication flaws).

Dependencies include frontend-backend integration and testing.

Timeline: two days (October 7-8).

11) Stakeholder Feedback Session

Task: Organize feedback sessions with stakeholders (lecturers, coordinators, and managers).

Collect input on the usability, features, and general system performance.

Based on feedback, iterate on design and functionality (for example, improving the user interface, addressing bugs, and refining claim tracking).

Dependencies include usability testing and security implementation.

Timeline: three days (October 9-11).

12) Final testing and deployment tasks:

Complete the last phase of testing, which includes regression and performance testing.

Deploy the prototype to a cloud platform (such as Azure App Services).

Validate the deployment environment and confirm that all functionalities work after deployment.

Dependencies include stakeholder feedback and iteration.

Timeline: two days (October 12-13).

13) Documentation and Deliverables

Tasks: Prepare and distribute detailed project documentation, including design decisions (UI, database, API structure).

Detailed explanations of assumptions, restrictions, and trade-offs made during the project.

Instruction guides for lecturers, coordinators, and managers.

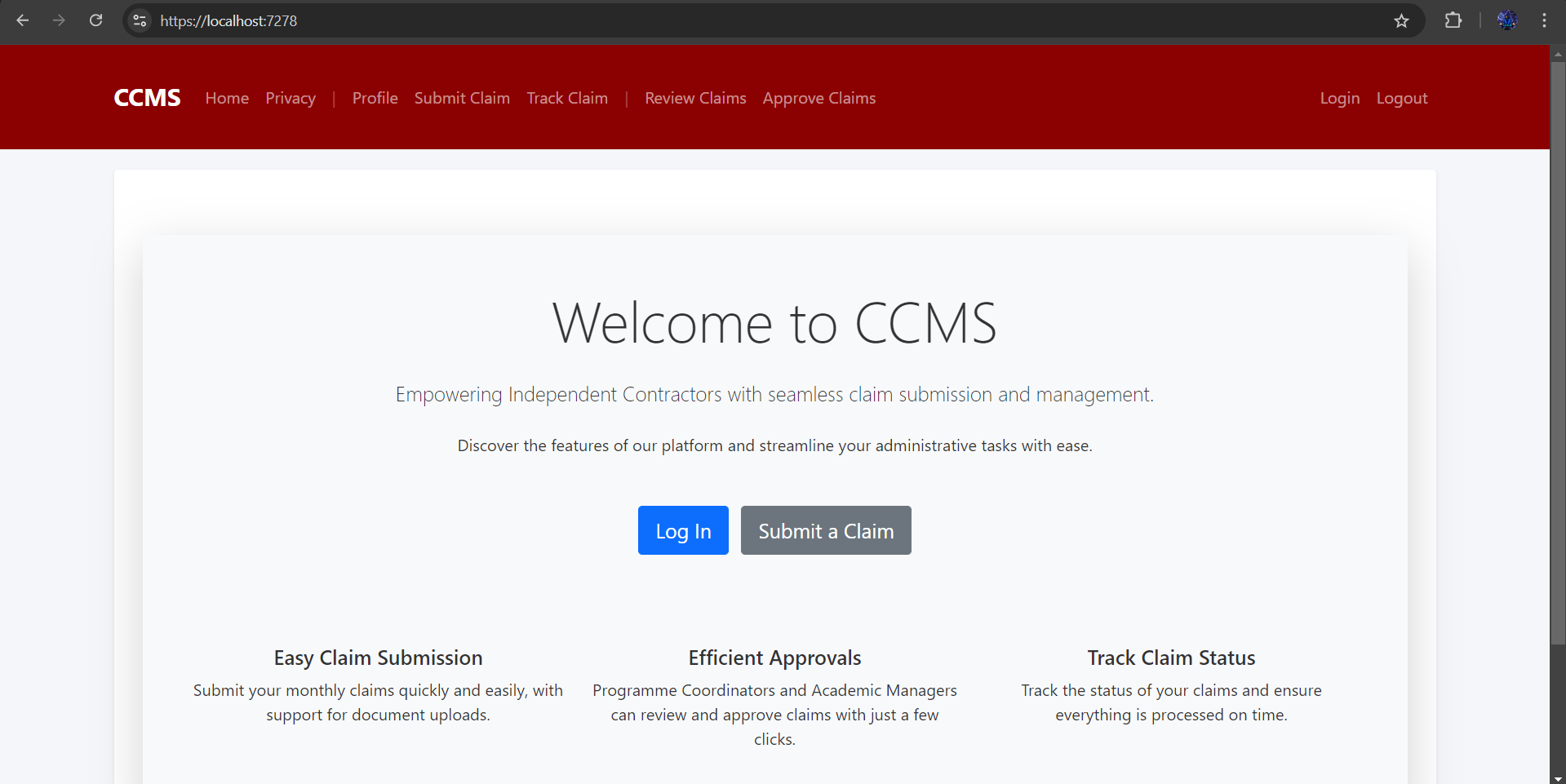
Deliver a final report that includes screenshots, URLs, and repository links (GitHub).

Dependencies: final deployment.

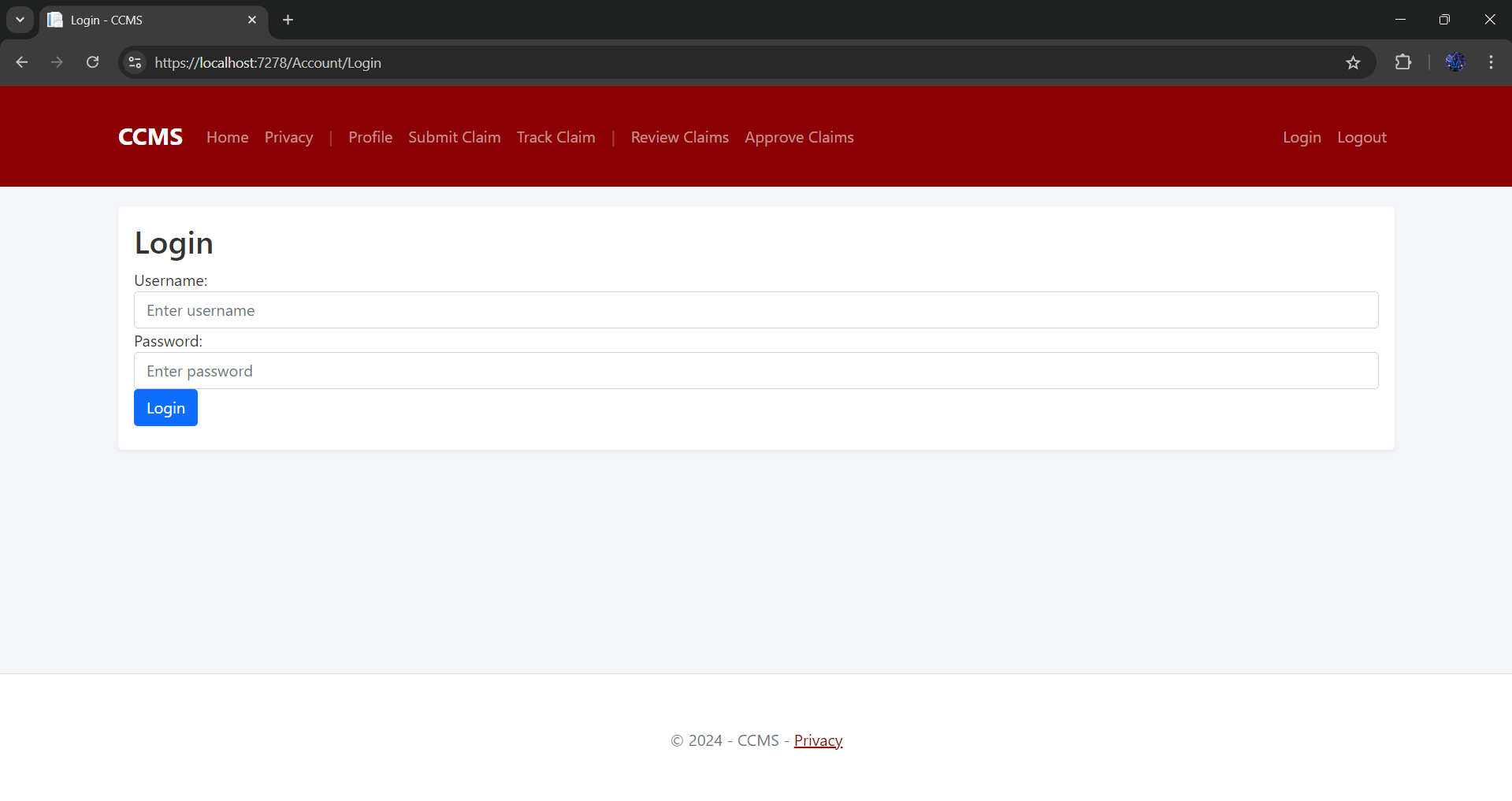
Timeline: two days (October 14-15).

(Lockhart.2021)

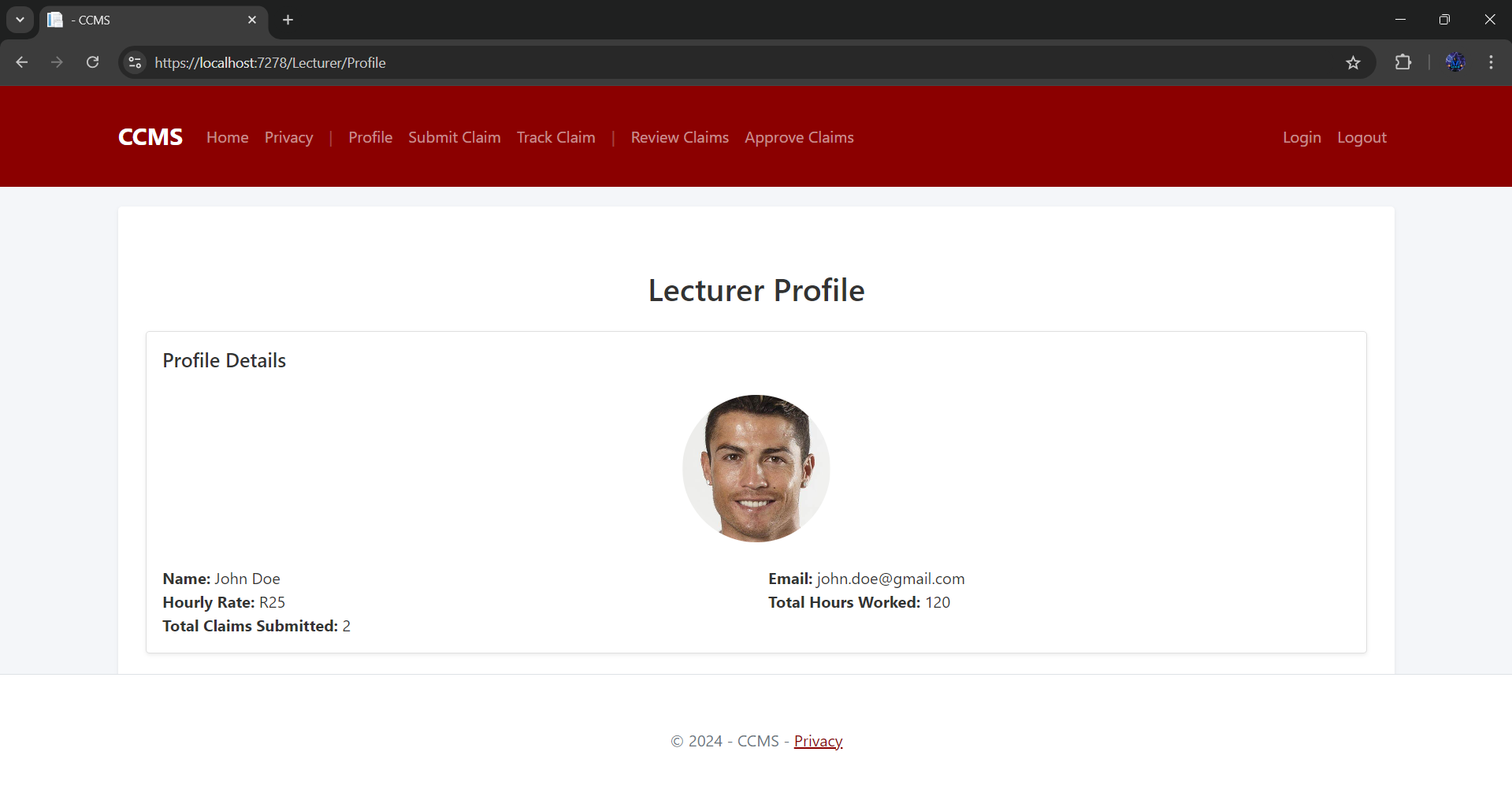
4**)GUI**

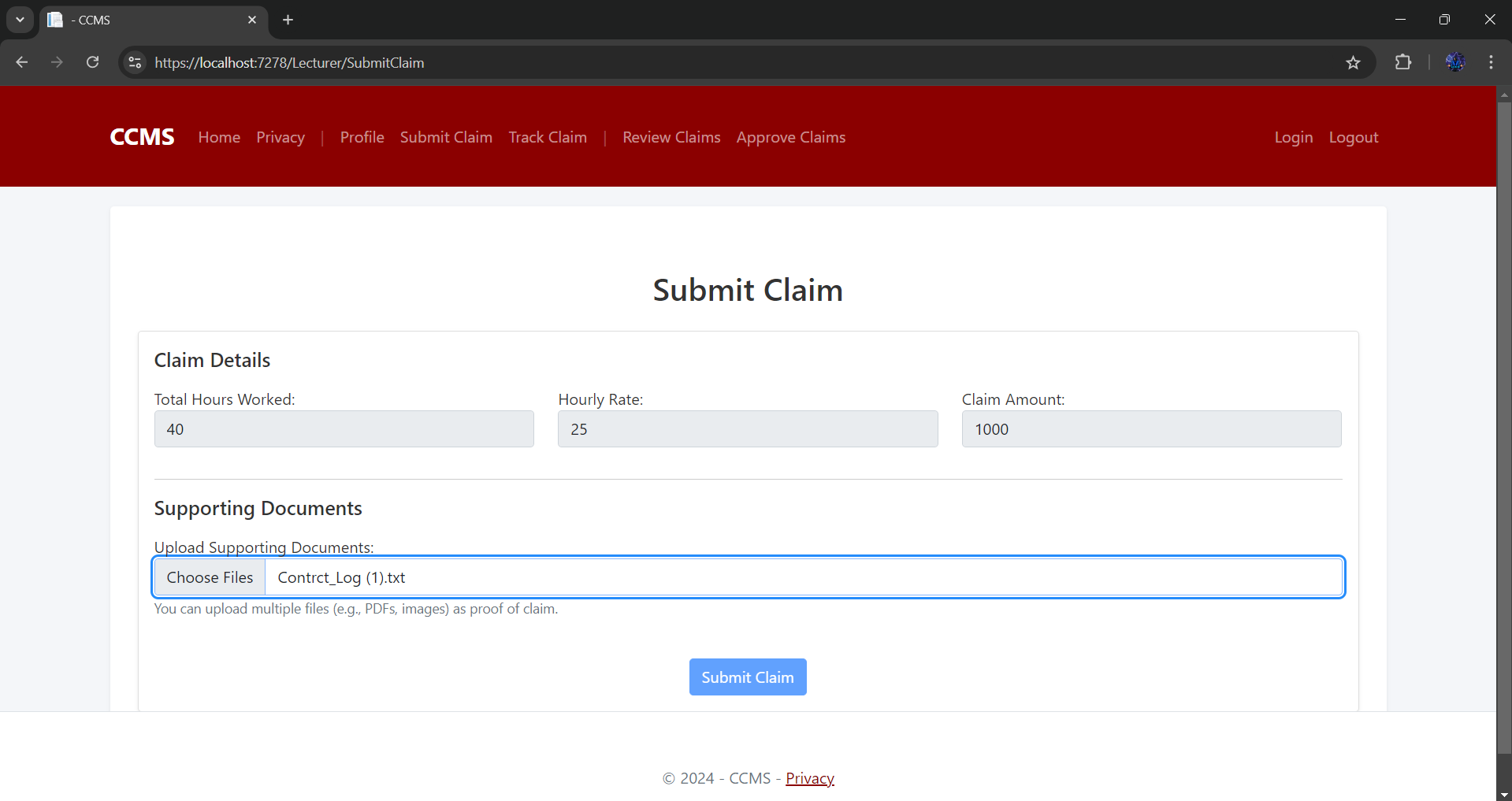


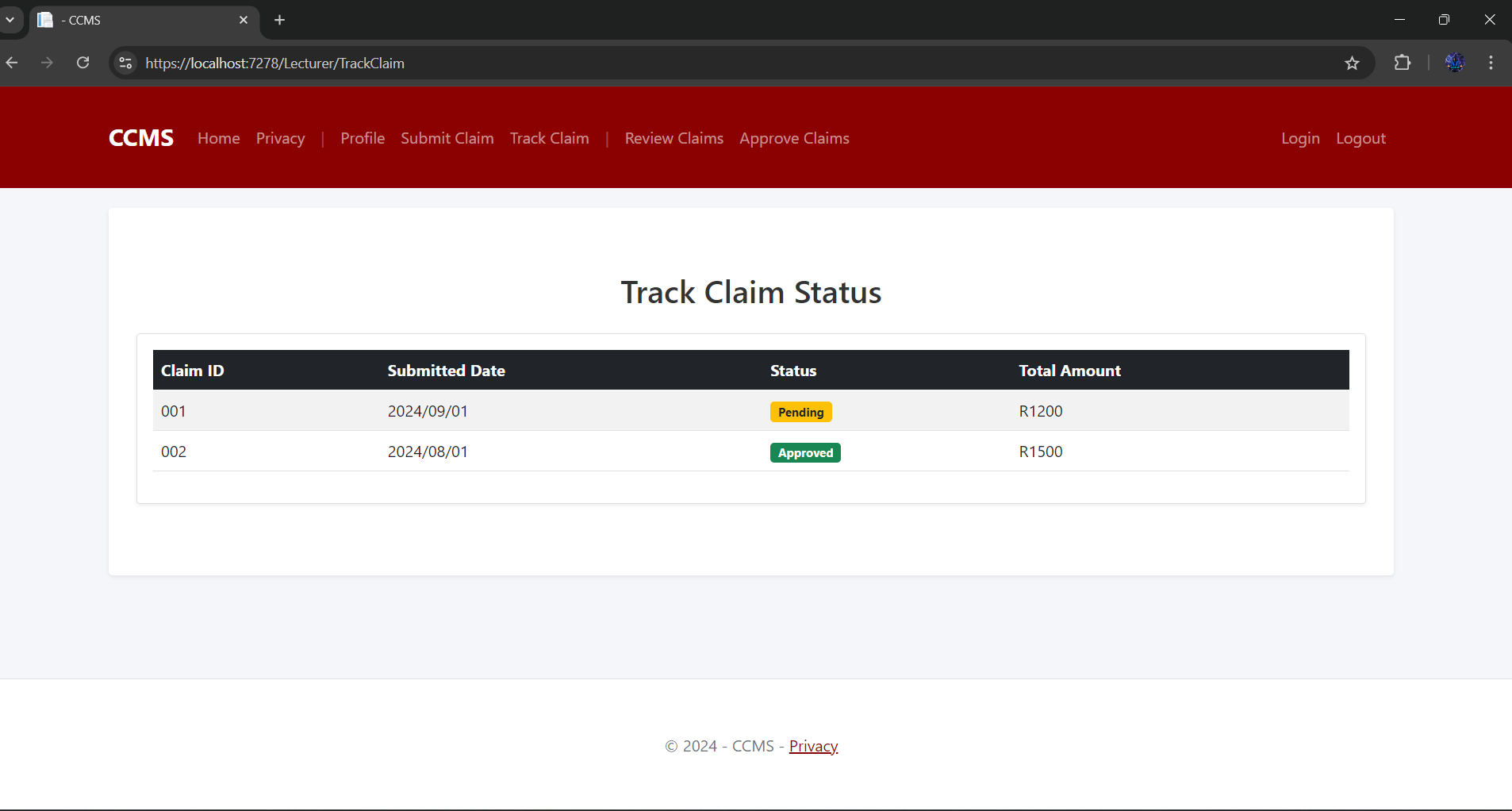
A normal user that is not logged in can only access the home and privacy pages



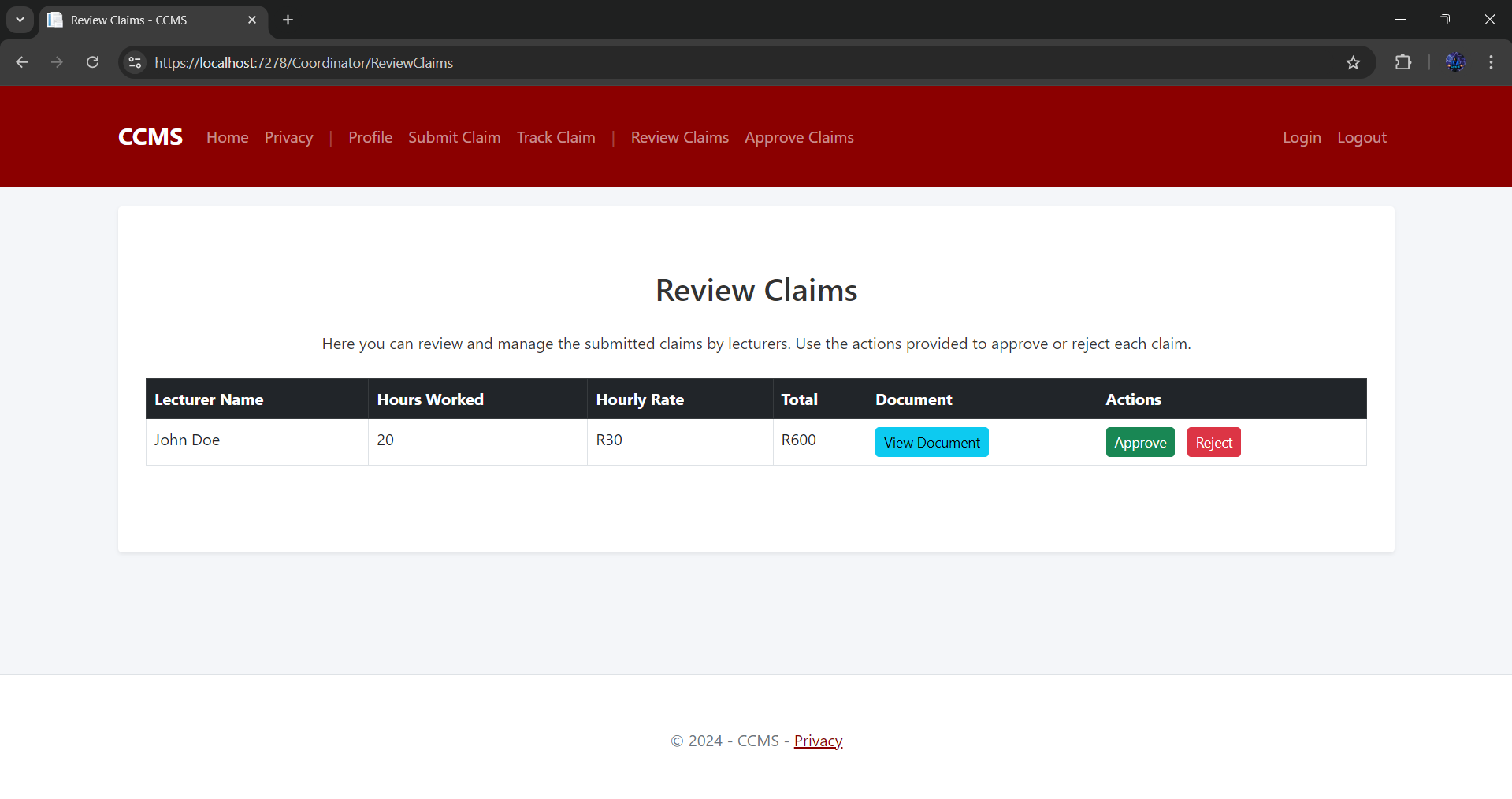
A lecture can access the Profile, submit claims and Track claims while the Programme Coordinators and Academic Managers can access everything

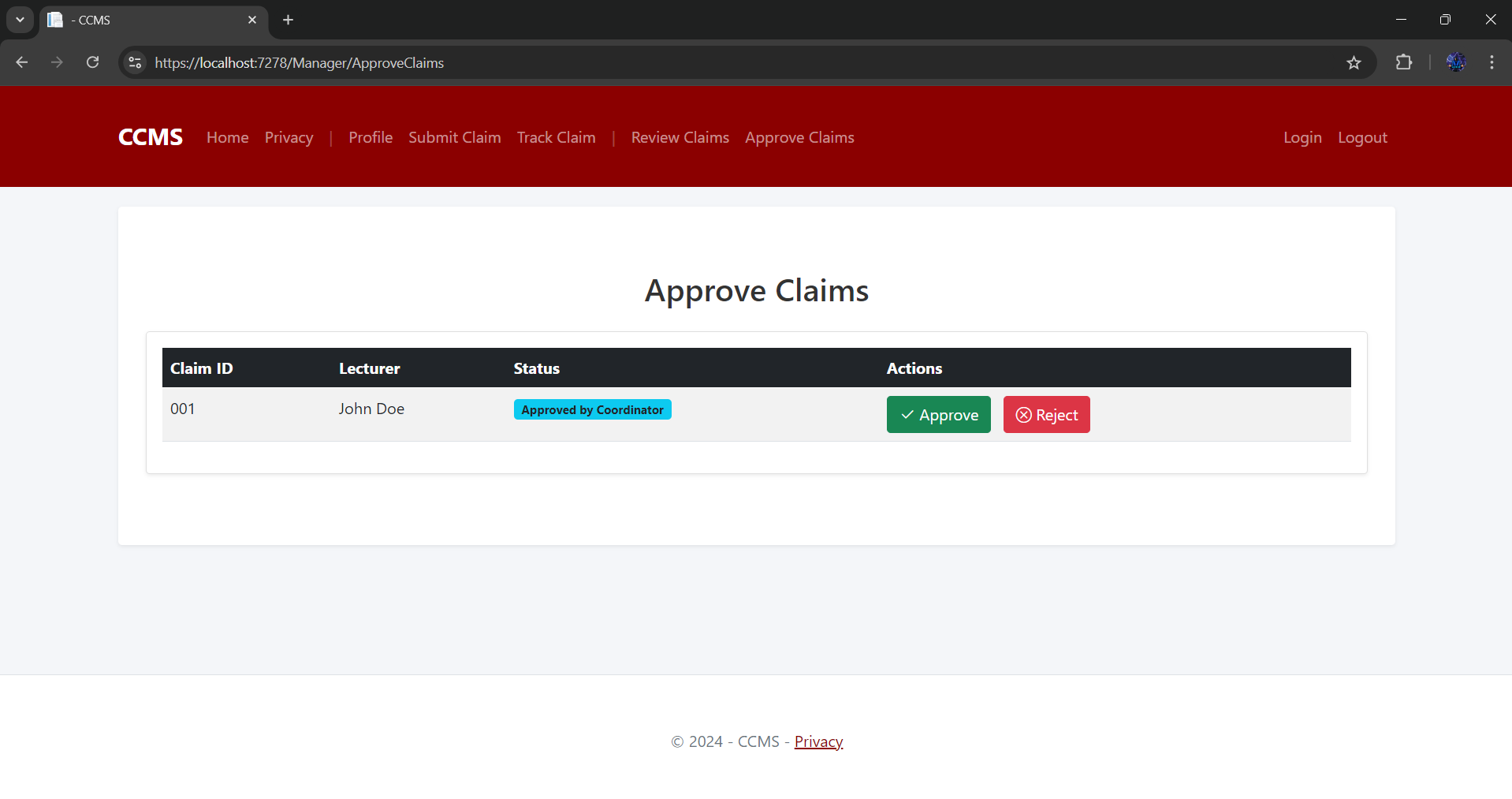






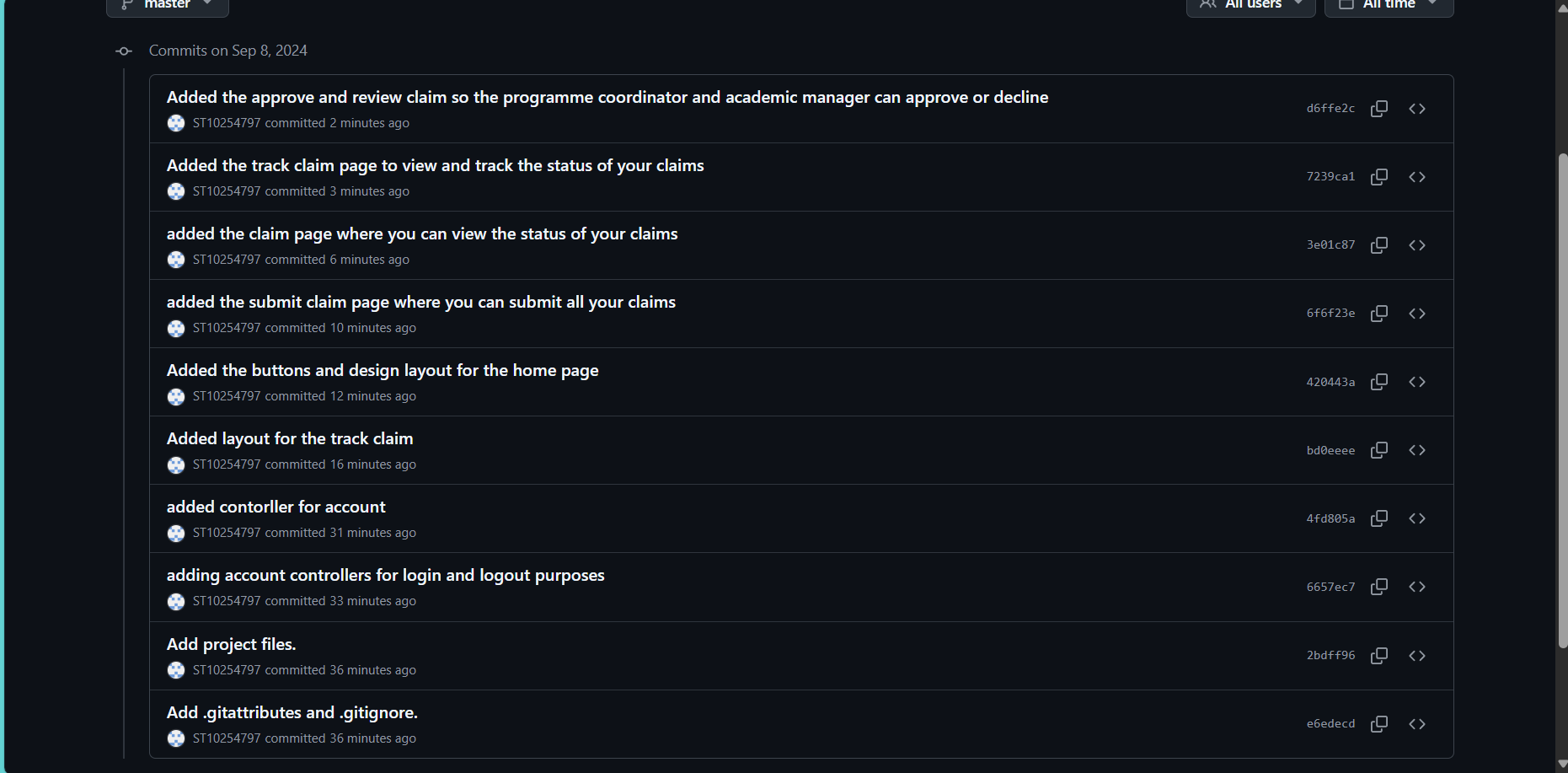
The lectures can view their profile, submit claims with a push of a button and track their claims.





This is for the Programme Coordinators and Academic Managers, they can review the claims and approve the claims

5)**Git**



<https://github.com/ST10254797/CCMS.git> (my github link)

<https://github.com/VCDN-2024/prog6212-poe-part-1-ST10254797.git> (github link provided)

**References:**

Lockhart, L. 2021. How to Develop a Project Plan: Template & Example [2024]. [Online]. Available at: <https://thedigitalprojectmanager.com/projects/managing-schedules/project-plan-guide/>

(Accessed 6 September 2024)

Microsoft. 2024. Create a UI by using XAML Designer. [Online]. Available at: <https://learn.microsoft.com/en-us/visualstudio/xaml-tools/creating-a-ui-by-using-xaml-designer-in-visual-studio?view=vs-2022>

(Accessed 5 September 2024)

Mortier, G. 2023. Top 11 Best Practices for Database Design. [Online]. Available at: <https://vertabelo.com/blog/best-practices-for-database-design/#:~:text=The%2011%20Commandments%3A%20Best%20Practices%20of%20Database%20Design,Makes%20Sense%20to%20Do%20So%20...%20More%20items>

(Accessed 5 September 2024)

Zight. 2019. Design Thinking Assumptions: Build, Measure, Learn and Ideate. [Online]. Available at: <https://zight.com/blog/design-thinking-assumptions/>

(Accessed 5 September 2024)