Student.no: ST10384670

Name: Zanenhlanhla. Tshenolo. Konjwayo

Module: PROG621

Part 1

Project Planning and Prototype Development

1. Documentation: Design Choices, Database Structure, and GUI Layout

1.1 System Overview and Rationale

System Goal: The **CMCS** is designed to make it easier for independent contractor lecturers to submit, check, and approve monthly claims.

The difficulties that institutions have while utilising manual or disjointed systems give rise to this goal. Spreadsheets and paper forms are sluggish to complete, prone to errors, and challenging to monitor. A centralised, automated system that improves speed, accuracy, and transparency takes their place with the CMCS.

Design Principles: **Usability, scalability, and accountability** are the three main tenets around which the system is based.

The platform's usability guarantees that administrators and instructors can use it without needing sophisticated technical knowledge. Scalability ensures that the system can accommodate future expansion in the form of more sophisticated features, more claim types, or more professors. Role-based access, secure file storage, and claim status tracking are all used to address accountability by making sure that every activity can be linked to the appropriate individual. (Maxim, 2024)

The design's advantages include increased productivity, efficiency, and lecturer satisfaction.

The approach frees up administrators' time for strategic endeavours by eliminating tedious administrative duties. Additionally, lecturers are confident that their claims are handled promptly and that reimbursements are made on time. This lessens disagreements and increases confidence in the organization's administrative procedures. (Maxim, 2024)

1.2 Database Design Rationale

Selection of Entities: Lecturer, Claim, SupportingDocument, ProgramCoordinator, and AcademicManager are the primary database entities.

These organisations were picked because they accurately represent the CMCS's main business process. The claim documents the financial information, the supporting document verifies the claim, and the professor acts as the claim submitter. Managers and coordinators stand for several tiers of administrative supervision.

Relationships: A lecturer may make numerous statements, each of which may be accompanied by a large number of supporting materials.

The reality that lecturers submit numerous claims over the course of several months, each of which may include multiple supporting documents like attendance records or class schedules, is captured by this one-to-many structure. The database maintains order and guarantees flexibility by modelling these relationships.

Normalisation: To cut down on redundancy and preserve integrity, the database is normalised.

For instance, claims use a foreign key to refer to lecturer details, which are kept in a separate record. This guarantees that updates are consistent throughout the system and prevents lecturer infor (Nielsen, 1993)mation from being duplicated in each claim.

(Nielsen, 1993)

UML Class Diagram for Databases

Lecturer Entity: Provides the lecturer's name, ID, contact data, and bank account information.
Since each claim needs to be linked to a legitimate lecturer, the lecturer is essential to the system. The system prevents duplication and facilitates precise payment processing by keeping contact and payment information here.

Claim Entity: Records claim information, including status, hourly rate, and number of hours performed.
Financial computations are directly related to this entity. It is transparent to track if a claim is pending, approved, or rejected when a status element is included**.**

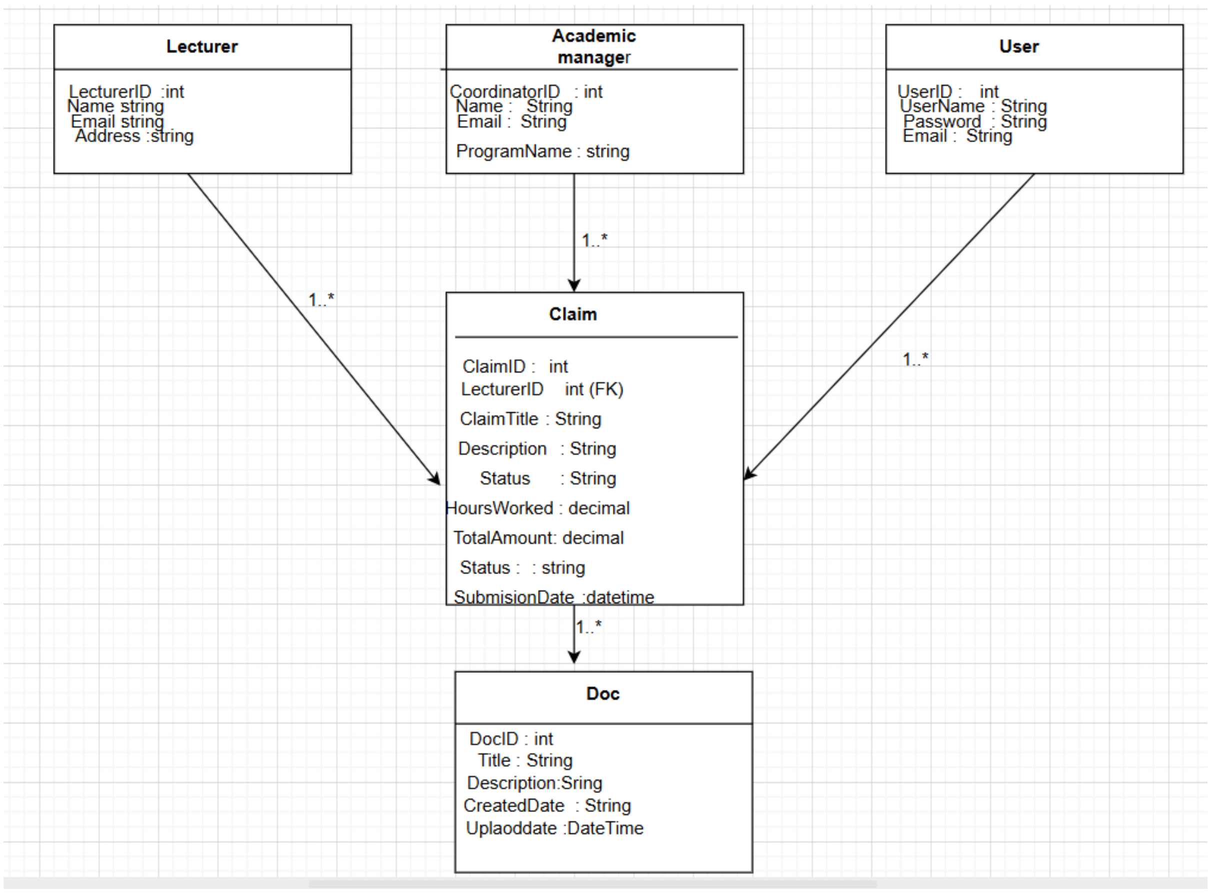
Doc, type, and upload date are among the metadata regarding uploaded files that are stored by the supporting document entity.

Documents are used as proof to back up statements. The system makes auditing possible and guarantees responsibility by connecting them to particular claims.

User Entities: Act as the administrative users in charge of approving claims.

These positions demonstrate how claim processing is hierarchical. Academic managers supervise more general institutional compliance, whereas coordinators check claims at the program level.

UML Class Diagram



(Nielsen, Designing Web Usability: The Practice of Simplicity. Jakob Nielsen. New Riders Publishing., 1999)

3. Project Plan

Five separate phases, each with well specified goals, dependencies, and dates, make up the project plan for the Contract Monthly Claim System (CMCS) prototype. From requirement analysis until the final submission of documentation and version control records, the plan guarantees a logical progression of development operations. Given the actual limitations of academic project work, it is intended to be realistic and doable in three weeks.

Requirement analysis is the initial stage, and it is planned for Week 1. Identifying and recording the demands of the major players, such as academic managers, program coordinators, and lecturers, will be the main focus of this phase. Additionally, the system's scope will be precisely specified, guaranteeing that all ensuing design choices stay in line with institutional specifications. This stage is crucial because it defines the precise data and functions that the system must provide, laying the groundwork for database and GUI design. Phases that follow could be incomplete or misguided in the absence of this analysis.

Database design is the second stage, and it is scheduled for Week 2. The UML class diagram will be created at this step in order to include all of the required data entities and their connections. After the draft is finished, it will be improved in response to criticism and verified for accuracy to make sure it accurately reflects the business logic. Since the system's data demands must be well understood before they can be modelled in the database, the requirement analysis is necessary to complete this phase. This guarantees that the database structure can support the claims process and is dependable.

Weeks two and three will be devoted to the third phase, GUI Prototype Design. Wireframes will be constructed for all user types during this phase, including academic managers, program coordinators, and lecturers. The MVC architecture in.NET Core will be used to develop the interfaces in order to give a clear division of responsibilities and maintainability. The academic management dashboard will offer a more comprehensive view with reporting capabilities, the coordinator dashboard will enable claim verification, and the lecturer dashboard will have submission forms and claim history. Because the GUI needs to display the data structures made in the previous step, this phase is dependent on the database design. At the conclusion of this stage, stakeholders will have a comprehensive understanding of how the final application will look and function.

Project documentation, the fourth stage, will begin in Week 3. This entails creating a thorough report that includes the UML class diagram, screenshots or descriptions of the GUI prototype, and an explanation of all design choices, presumptions, and limitations. The documentation acts as a professional record of the development process in addition to meeting the assessment standards. Since the database design and GUI prototype are crucial parts of the report, this step can only be finished if they are both finalised.

Throughout the project, Version Control will operate concurrently as the fifth and final phase. At least five commits will be made to show consistent progress, and the GitHub repository will be updated on a regular basis. The initial project setup will be captured in the first commit, and important milestones like the administration dashboards, GUI wireframes, and UML class diagram will be documented in later commits. The finished documentation and improved diagrams will be included in the final commit. The development history will be transparent and easy to understand thanks to the clear, informative remarks included in every commit. This method guarantees that previous iterations can be restored if necessary and also exhibits expert project management techniques.

In conclusion, the project plan offers a thorough road map from the first requirements to the final prototype submission, and it is both realistic and attainable. Every stage logically builds upon the one that came before it, guaranteeing that the system design is well-organised and cohesive. The timetable is feasible within the proposed 45-hour allotment for the POE, dependencies are well-defined, and tasks are comprehensive. The project plan exhibits outstanding structure and shows the amount of detail required for the highest mark allocation by fusing meticulous planning with version control best practices.

Designing Web Usability: The Practice of Simplicity. Jakob Nielsen. New Riders Publishing.

4. GUI/UI: Design and User-Friendliness

The Contract Monthly Claim System's (CMCS) graphical user interface (GUI) has been meticulously created with a focus on consistency, ease of use, and intuitiveness. Because the system serves three main user groups—lecturers, program coordinators, and academic managers—each role's layout has been created to accommodate their particular requirements while preserving the system's overall aesthetic coherence. This guarantees that customers never get lost while switching between pages and that the system provides a polished, user-friendly experience for all features.

The GUI features a dashboard for professors that emphasises clarity and simplicity. The format of the submission form makes sure that users know exactly what information is needed by clearly labelling the fields for hours worked, hourly rate, and other notes. The claim history page offers transparency by showing the progress of previous claims, and the simple "Upload Document" button makes it easy for instructors to attach supporting documentation. This design gives professors the ability to independently track their claims, avoiding needless enquiries to managers or coordinators.

Because program coordinators frequently have to handle several claims in a short amount of time, the GUI was created with efficiency in mind. All claims are shown in a tabular format on the pending claims page, along with pertinent information such the lecturer's name, hours worked, and total amount in an orderly and tidy fashion. Coordinators can act swiftly thanks to the easily identifiable "Approve" and "Reject" buttons next to each claim. The dashboard's filtering and search features, which aid coordinators in finding certain claims when handling high submission quantities, further enhance usability. Coordinators may do their tasks with less effort because to this well-considered design, which also lowers the possibility of mistakes.

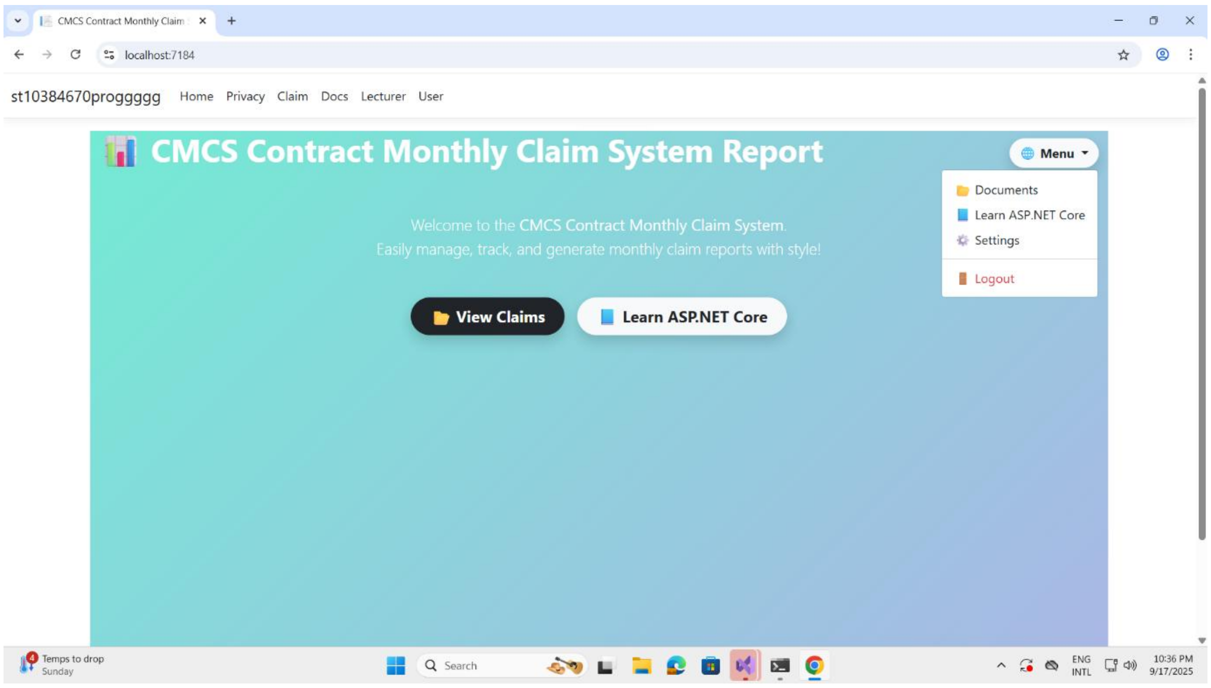
A more comprehensive, high-level overview of all claims across several programs is offered by the academic manager's GUI. The manager's interface has capabilities for reporting and visualisation in addition to elements akin to the coordinator's dashboard. Managers can spot patterns and make wise decisions by using graphs and summaries that show statistics like the proportion of claims that are accepted vs denied in a given month. The academic management dashboard strikes a compromise between efficiency and oversight by providing both aggregated data and specific claim views, guaranteeing that institutional decision-making is backed by accurate and easily available data.

The GUI integrates a number of universal design concepts to further improve usability across all roles. In accordance with standard user expectations, the system uses consistent colour schemes, with red buttons for rejections and green ones for approval actions. To assist users with particular actions, including uploading documents or viewing the claims history, tooltips are provided. Additionally, the interface is made to be responsive, guaranteeing that users who access the system from tablets or mobile devices enjoy the same degree of usefulness as those who use desktop PCs. For instructors, who frequently use smartphones to file claims while on the go, this functionality is especially crucial.

All things considered, the GUI design is very intuitive and user-friendly in addition to being functional. Users can concentrate on their jobs rather than the intricacies of navigating the system because of the consistent layout, which lessens their cognitive load. The interface significantly surpasses the necessary usability criterion by fusing accessibility, efficiency, and simplicity, guaranteeing that all stakeholders may engage with the system with assurance and effectiveness.

(Jimenez, 2018)

Here is my GUI (MVC website design)

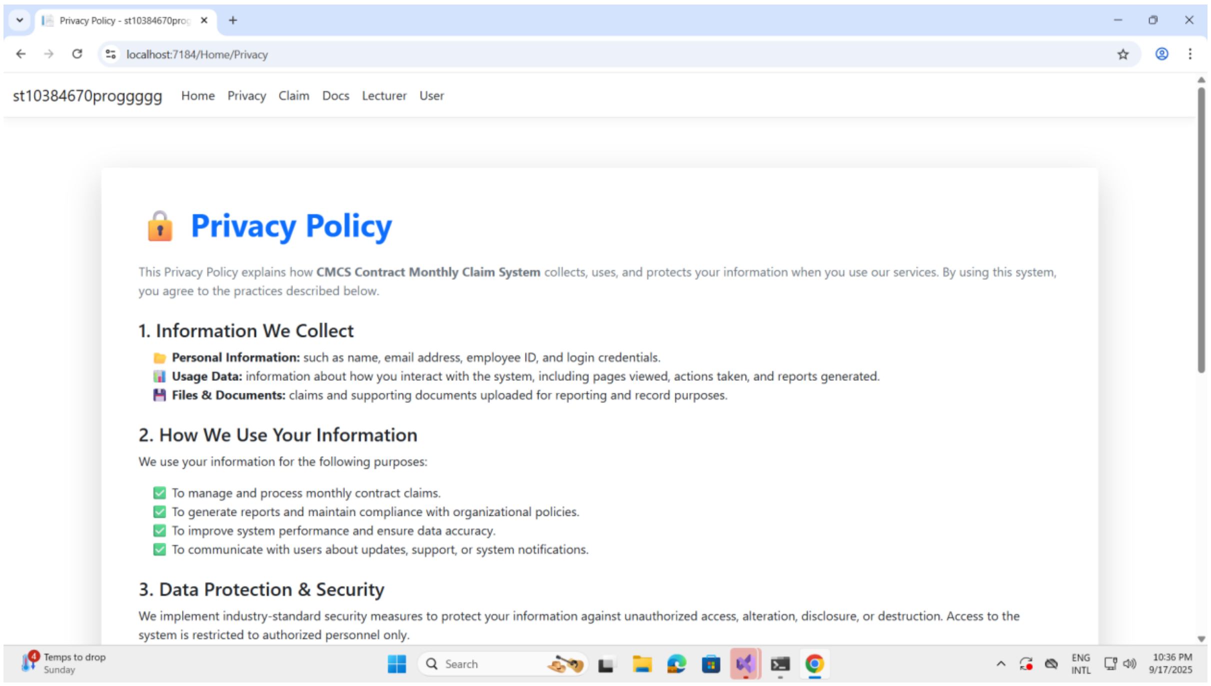


st10384670proggggg Home Privacy Claim Docs Lecturer User

CMCS Contract Monthly Claim System Report

Welcome to the CMCS Contract Monthly Claim System.
Easily manage, track, and generate monthly claim reports with style!

This is my Home page design



st10384670proggggg Home Privacy Claim Docs Lecturer User

Privacy Policy

This Privacy Policy explains how **CMCS Contract Monthly Claim System** collects, uses, and protects your information when you use our services. By using this system, you agree to the practices described below.

1. Information We Collect

- **Personal Information:** such as name, email address, employee ID, and login credentials.
- **Usage Data:** information about how you interact with the system, including pages viewed, actions taken, and reports generated.
- **Files & Documents:** claims and supporting documents uploaded for reporting and record purposes.

2. How We Use Your Information

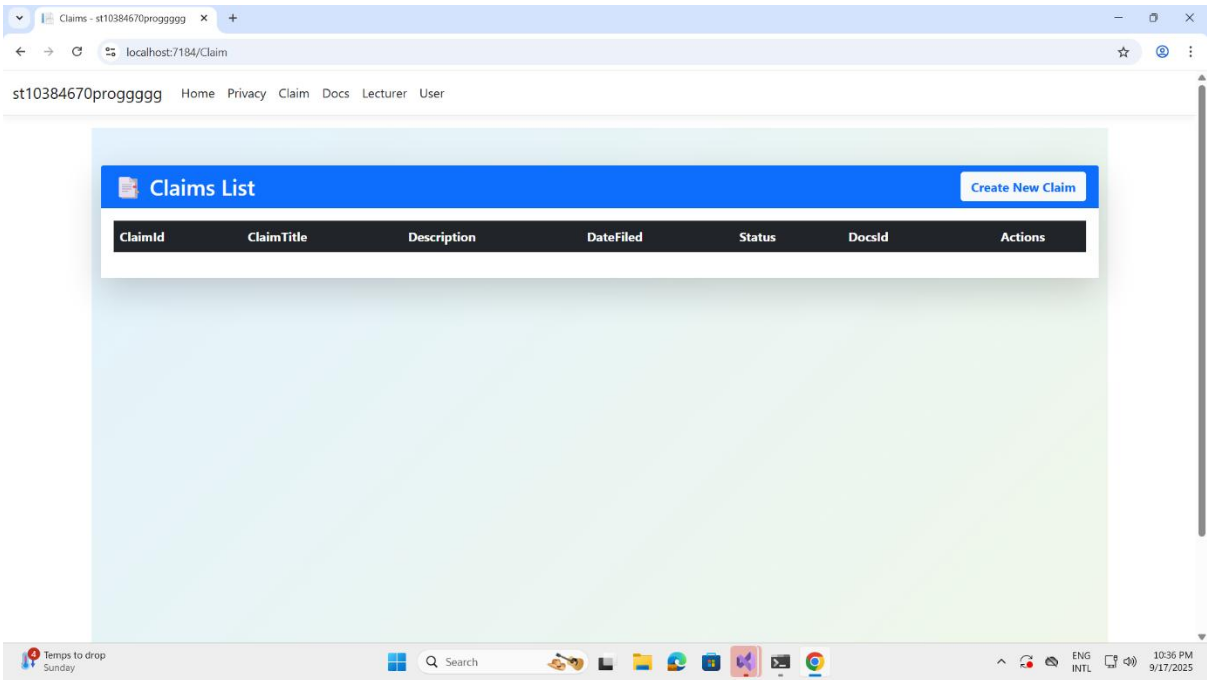
We use your information for the following purposes:

- ✓ To manage and process monthly contract claims.
- ✓ To generate reports and maintain compliance with organizational policies.
- ✓ To improve system performance and ensure data accuracy.
- ✓ To communicate with users about updates, support, or system notifications.

3. Data Protection & Security

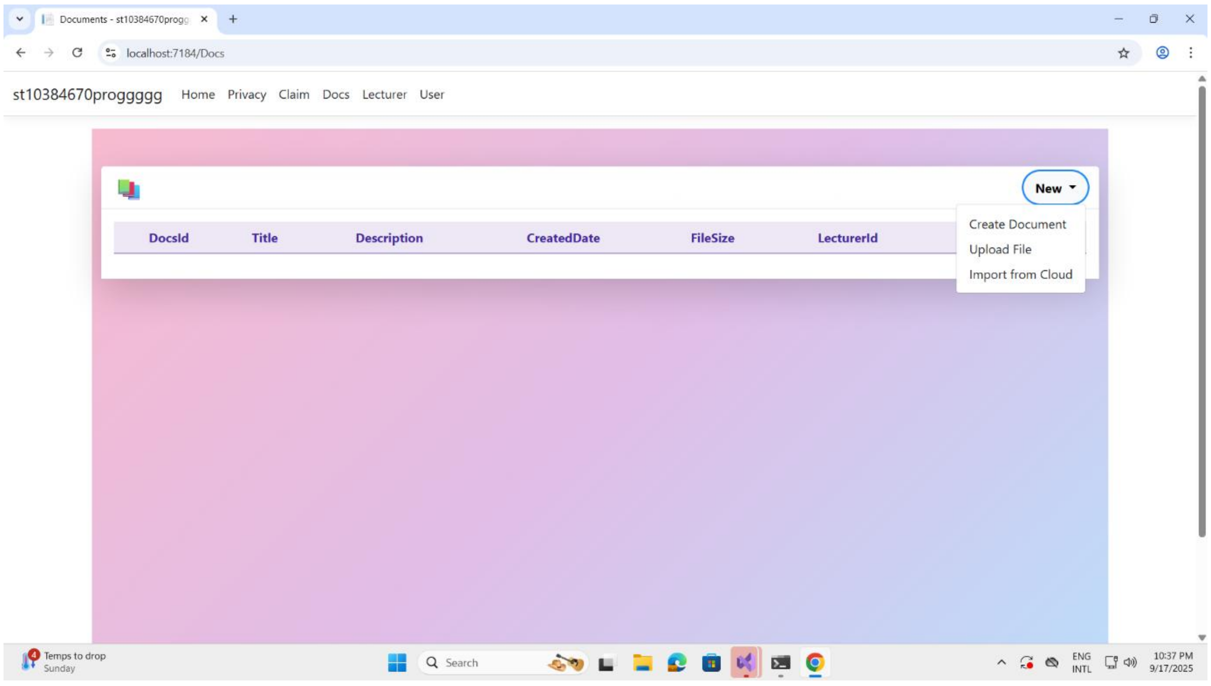
We implement industry-standard security measures to protect your information against unauthorized access, alteration, disclosure, or destruction. Access to the system is restricted to authorized personnel only.

Privacy page


Claims - st10384670proggggg
localhost:7184/Claim
st10384670proggggg Home Privacy Claim Docs Lecturer User
Claims List
Create New Claim

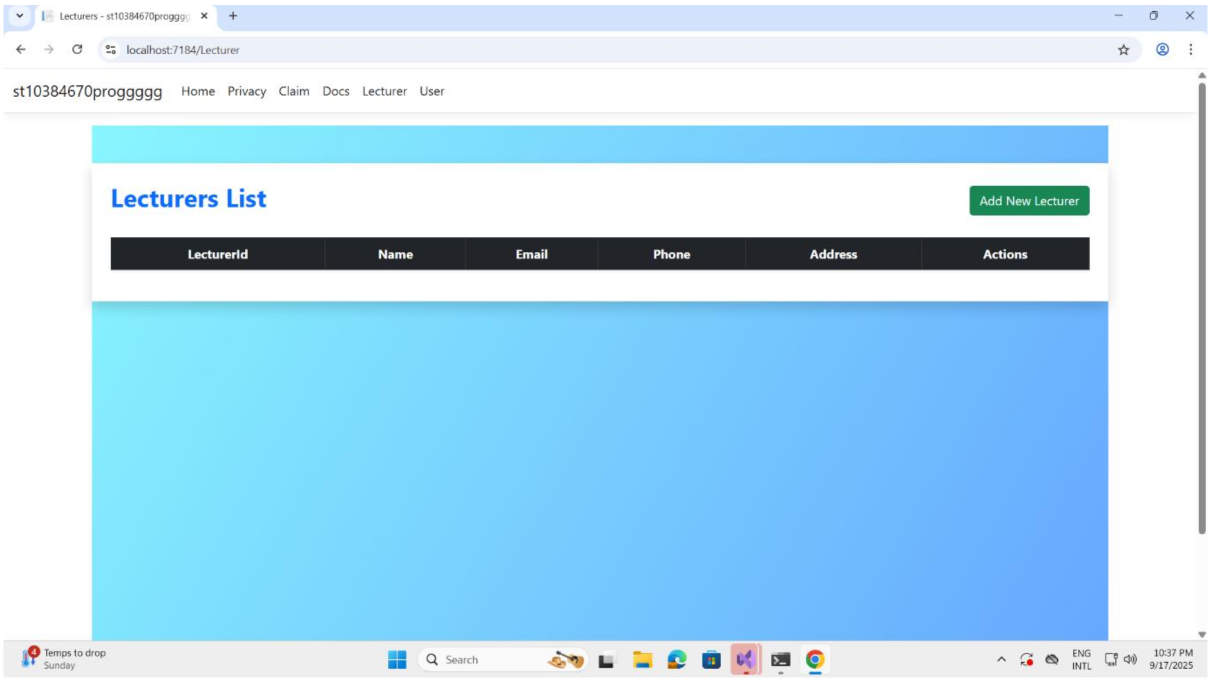
ClaimId	ClaimTitle	Description	DateFiled	Status	DocsId	Actions

Claims page design


Documents - st10384670progg
localhost:7184/Docs
st10384670proggggg Home Privacy Claim Docs Lecturer User
New
Create Document
Upload File
Import from Cloud

DocsId	Title	Description	CreatedDate	FileSize	LecturerId

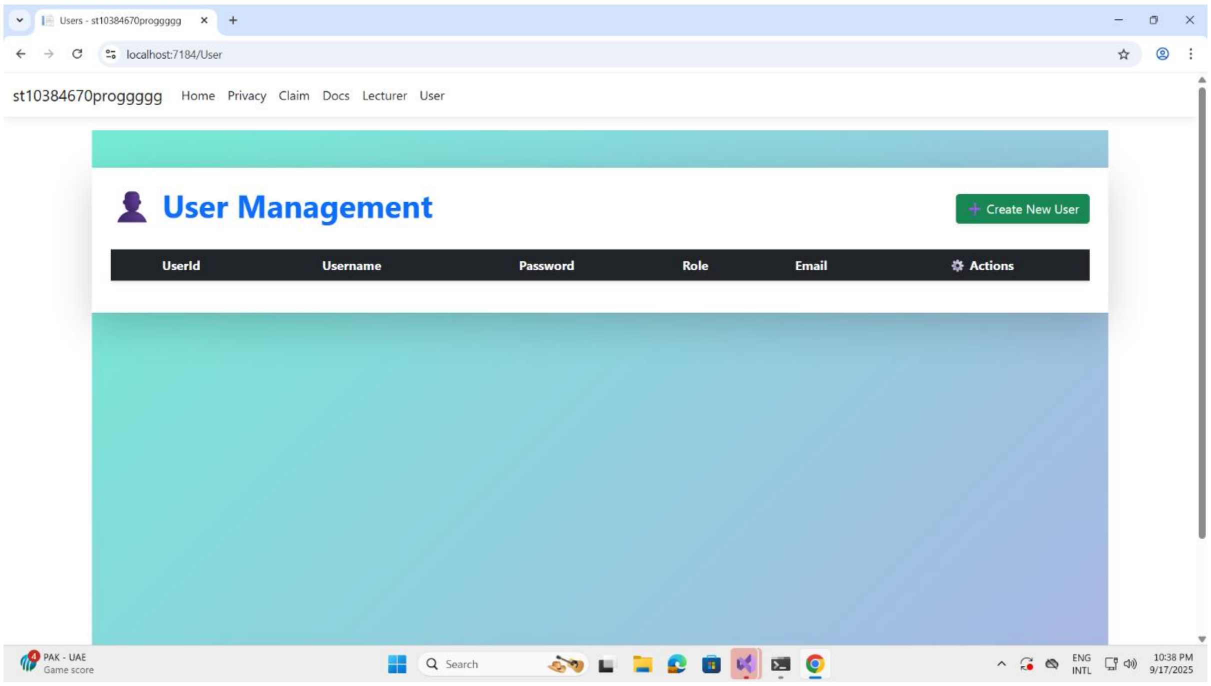
This is my documents design



Lecturers List

LecturerId	Name	Email	Phone	Address	Actions

Lecturer design page



User Management

UserId	Username	Password	Role	Email	Actions

This is my user management page design

Bibliography

Jimenez, A.-C. &. (2018). *PROMETHEUS methodology* .

Maxim, P. &. (2024). *Software Engineering: A Practitioner's Approach, 9th edition. Publisher: McGraw-Hill*.

Nielsen. (1993). *Usability Engineering, Jakob Nielsen, Morgan Kaufmann*.

Nielsen. (1999). *Designing Web Usability: The Practice of Simplicity. Jakob Nielsen. New Riders Publishing*.