

Part 1 Project Documentation:

Overview:

This Contract Monthly Claims System (CMCS) allows lecturers to submit claims (by entering details such as the hours they have worked, their hourly rate and any additional notes as well as allowing them to select a file or two to support their claim) for approval. The app tracks whether their claim has been approved or rejected. A coordinator, manager and HR member can approve or reject a lecturer's claim. A HR member can also manage a lecturer's data if they have received an update request from them. The app is built using the MVC (Model-View-Controller) architecture in C#.

1. Design Choices:

1.1. MVC Architecture:

1.1.1. MVC consists of the following:

- Model: Responsible for managing the application's data and its business logic, including claim information, statuses, documents, and lecturer data (Open AI, 2024).
- View: Represents the user interface (UI), which displays data such as the claims, their statuses and supporting documents (Open AI, 2024).
- Controller: Manages user input, utilizes the model, and updates the view, managing workflows such as submitting claims, updating their status, and managing lecturer data (Open AI, 2024).

1.1.2. Why MVC?

- I chose MVC because it is more flexible, efficient, dependable, better at web development, less prone to randomly breaking and easier to work with than WPF.

1.2. Language and Framework

- Language: C# due to its flexible and compatible nature with the .NET framework and MVC architecture.
- Framework: Due to its separation of concerns, we chose ASP.NET MVC, making the app easier to manage and scale.

2. Database Structure:

2.1 Tables

Claims

- Claim_Id (Primary Key, Auto-Increment): Unique identifier for each claim.
- Hours_Worked (double): Number of hours a lecturer worked.
- Hourly_Rate (double): Rate a lecturer makes per hour.

- Additional_Note (string): A description related to the claim.
- Document (Byte Array): The actual document(s) stored as a byte array.
- isApproved (Boolean): A boolean value representing the claim status (true for "Approved", false for "Rejected").
- Date_Submitted (DateTime): The date when the claim was submitted.

Lecturer

- Lecturer_Id (Primary Key, Auto-Increment): Unique identifier for each lecturer.
- First_Name (string): First name of lecturer.
- Surname (string): Surname of lecturer.
- Email (string): Email of lecturer.
- Password (string): Password of lecturer.

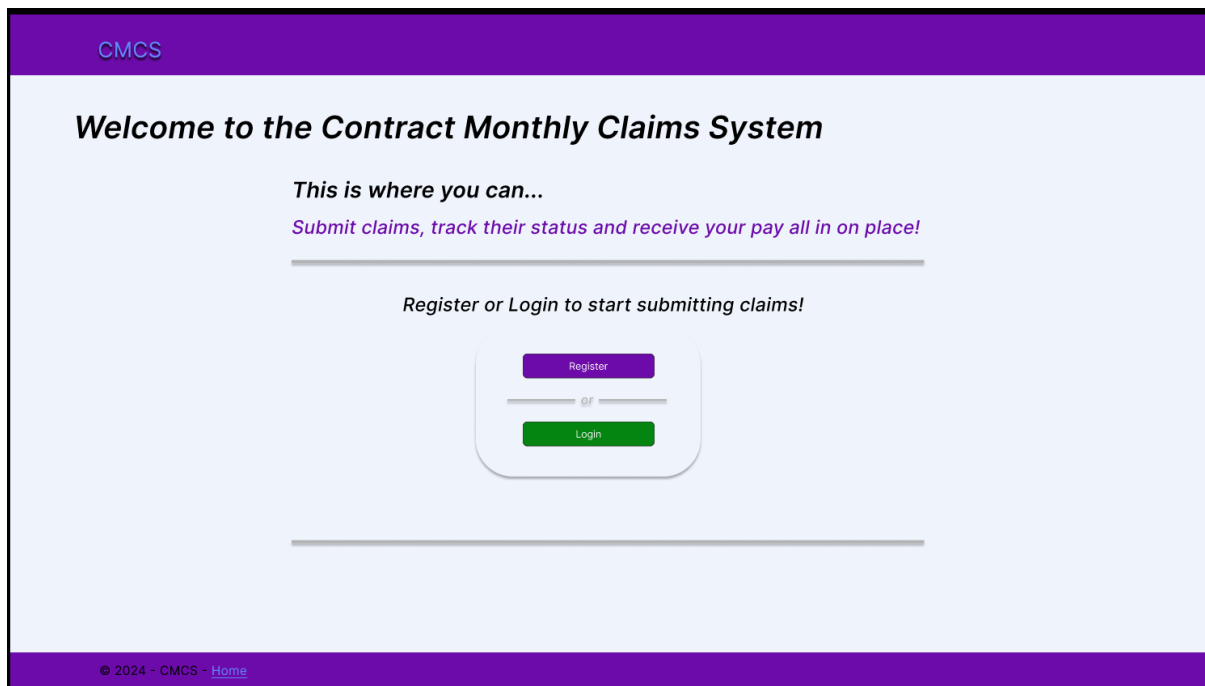
NOTE: The data structure for Coordinator, Manager and HR tables respectively is similar to the Lecturer table with the main difference being that each table has their own id and the attributes store data for a different type of user.

2.2 Relationships:

- Coordinator, manager, and HR users inherit from generic User table.
- One and only one lecturer can submit one or many claims.
- One and only one coordinator/manager/HR can approve or reject zero or many claims.

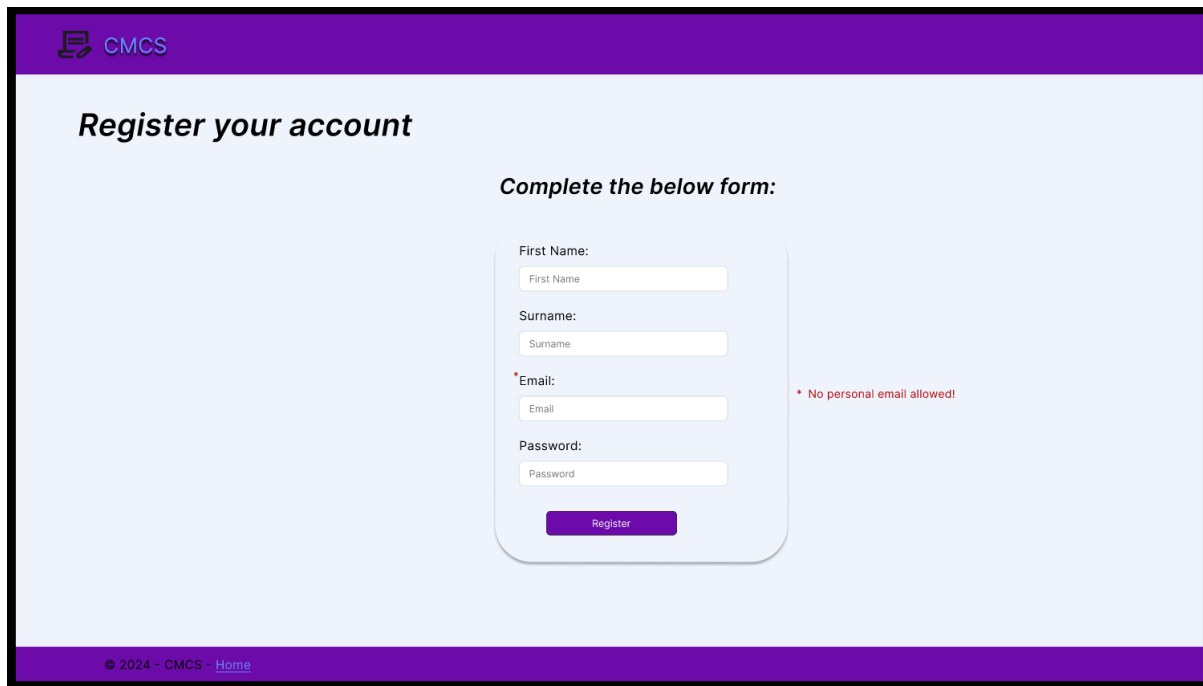
3. GUI Layout:

3.1. Landing View:



- As soon as one visits the CMCS, they are presented with page with two buttons:
- Register button allows them to create an account if they do not have one already.
- Login button allows a user to enter the system proper.

3.2. Registration View:



The image shows a web page for registering an account on the CMCS system. The page has a purple header with the CMCS logo and a light blue main content area. The title 'Register your account' is in bold. Below it, the instruction 'Complete the below form:' is centered. The registration form is a rounded rectangle with four input fields: 'First Name', 'Surname', 'Email', and 'Password'. The 'Email' field has a red asterisk and a red error message 'No personal email allowed!'. A purple 'Register' button is at the bottom of the form. The footer is purple and contains the text '© 2024 - CMCS - [Home](#)'.

CMCS

Register your account

Complete the below form:

First Name:

Surname:

* Email:
 * No personal email allowed!

Password:

© 2024 - CMCS - [Home](#)

- One would be able to register for an account here by entering details such as their first name, surname, email, and password.
- However, their email must be their academic email, or an email already associated with their role (Lecturer, Coordinator, Manager and HR)

3.3.1. Lecturer Home View:



Welcome Back, Lecturer!

The CMCS is where you can...

Manage your monthly contracts and submit claims to receive payment for your hard work!

What happens after claim submission...

Once your claim is submitted, it will undergo a review by a Programme coordinator and an Academic manager, who will either approve or reject your claim!

Additionally, you can get to track the status of your claims:

Stay up to date with approvals or rejections of your claims, and ensure everything is processed smoothly and you receive your payment

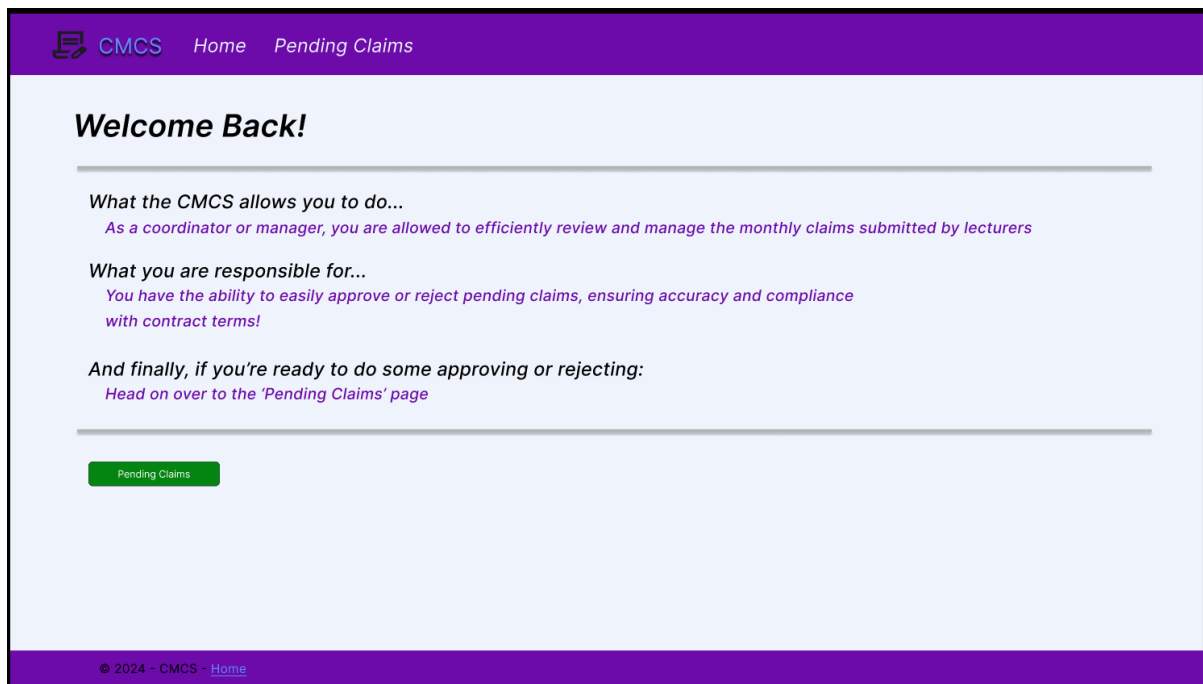
And finally, if you're ready to submit a claim:

Head on over to the 'Claims' page

Claims

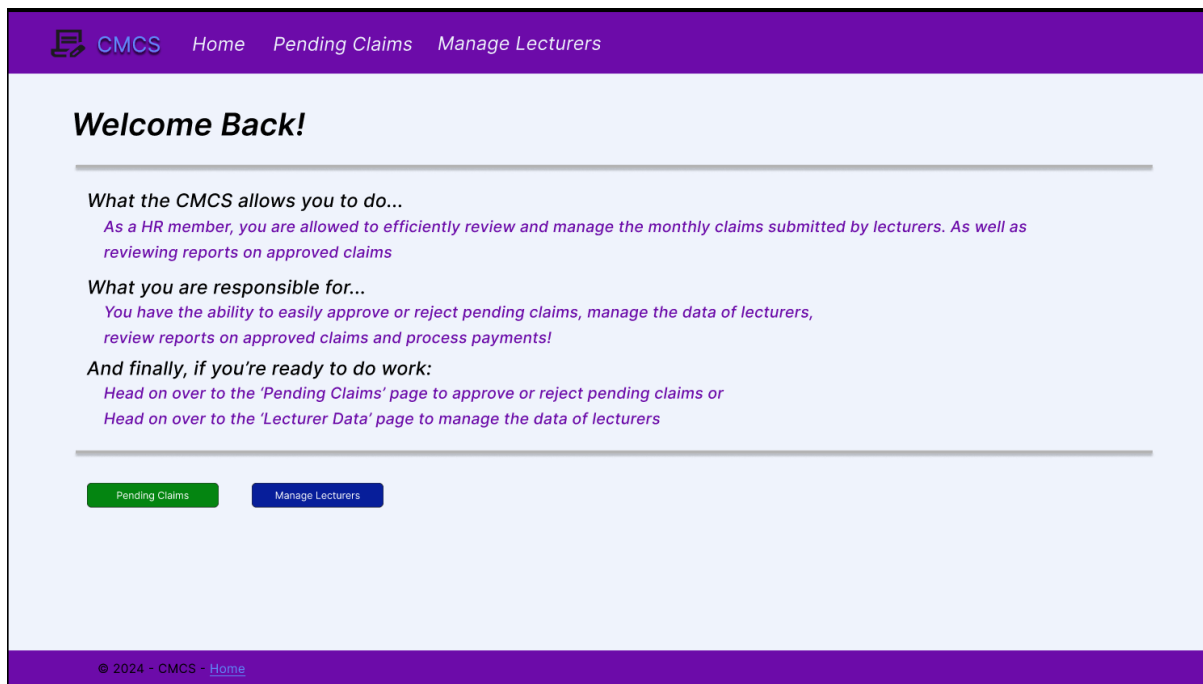
- This view informs the lecturer of how the claims approval process works and what role they play in that.
- This view includes a 'Claims' button, which once pressed, it redirects the user to the 'Claims' view where they will be able to submit a claim.

3.3.2. Coordinator and Manager Home View:



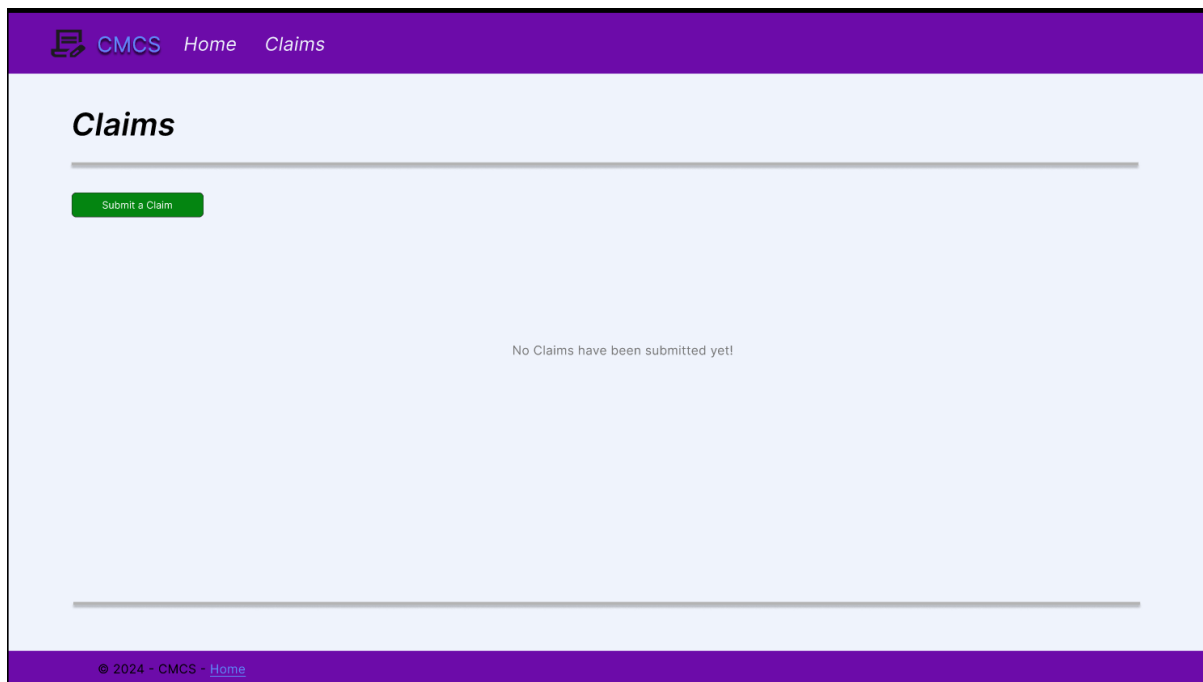
- This view informs the coordinator or manager and HR of their duties and responsibilities.
- This view includes a 'Claims' button, which once pressed, it redirects the user to the 'Pending Claims,' view which displays all submitted claims, along with 'Approve' and 'Reject' buttons (which do as they say) next to them.
- Before rejecting a claim, the coordinator or manager would be given a confirmation dialog asking whether they are sure or not.

3.3.3. HR Home View:



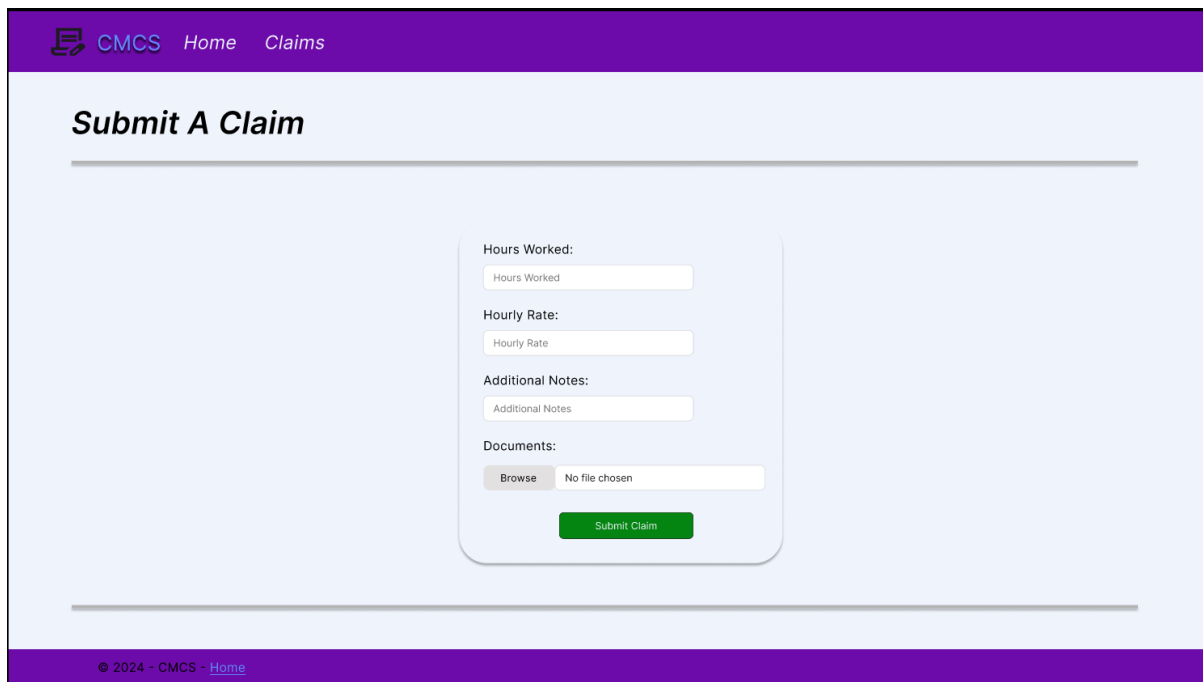
- In addition to being able to approve or reject claims as described in the previous view, a HR user can manage and update the data of a lecturer upon request through the press of the 'Manage Lecturer' button.

3.4 Claims View:



- All claims that a lecturer has submitted will be displayed here.
- If no claims have been submitted yet, the lecturer is alerted of such by an appropriate message.
- If the lecturer wishes to submit a claim, they can do so by pressing the 'Submit a Claim' button.

3.5 Submit A Claim View:



The screenshot shows a web application interface for submitting a claim. At the top, there is a purple navigation bar with a logo and the text 'CMCS Home Claims'. Below this, the main heading 'Submit A Claim' is displayed. The form itself is a light blue rounded rectangle containing several input fields: 'Hours Worked' with a label 'Hours Worked:', 'Hourly Rate' with a label 'Hourly Rate:', and 'Additional Notes' with a label 'Additional Notes:'. Below these is a 'Documents:' section with a 'Browse' button and a text field showing 'No file chosen'. A green 'Submit Claim' button is at the bottom of the form. The footer of the page is purple and contains the text '© 2024 - CMCS - Home'.

- After the Lecturer has navigated to this page, they will be able to enter all the necessary claim details, and submit a claim, through the use of the following elements:
- A text field is provided for each claim attribute input (Hours_Worked, Hourly_Rate and Additional_Note).
- Document upload: File upload control for supporting documents.
- Submit button: Allows user to submit claim once all data is input.

4. Assumptions:

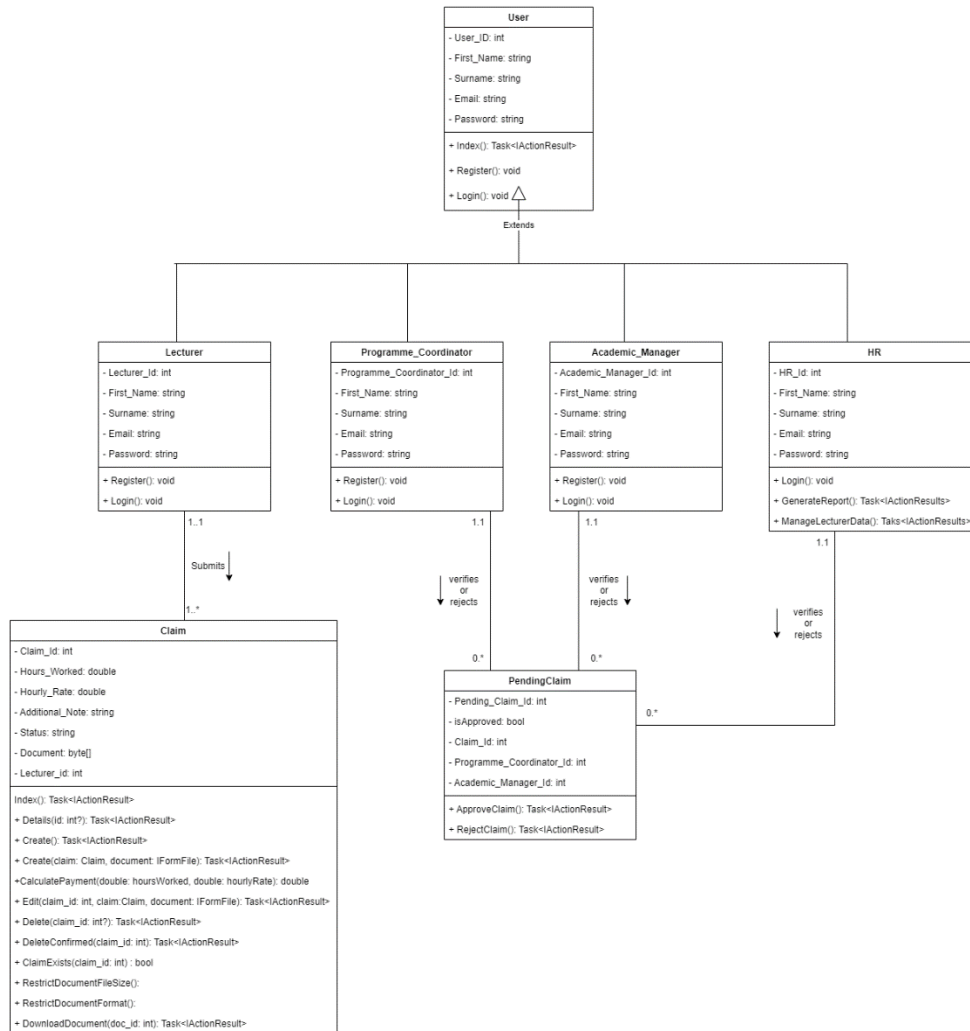
- **Submission of Claims:** Each lecturer can submit one or more claims with some restrictions in place. Lecturers can submit their claims at any time of the day, but they must submit their monthly claims before the 15th of each month otherwise they will be locked out of the claim submission feature until it is available again at the end of the month.
- **Uploading of Documents:** The app would ensure that documents are stored securely and are connected to the corresponding claim.
- **Data Integrity:** It is assumed that all input data (such as hours worked and hourly rate) will be provided in a proper format and will be validated before being stored in the database.
- **Pending Claims:** Coordinators and managers will have a separate view where all pending claims are displayed, along with the necessary details and options will be provided to allow them to verify or reject these claims.
- **Checking of claims against predefined criteria:** The app assumes that feature is a filtering feature that allows coordinators and managers to search for or display specific claims that meet a certain criterion.
- **HR:** It is assumed that the HR users have already been registered into the system and thus given access to the system and documents.

- **Generation and Viewing of Invoices:** The app assumes that the invoices summarizing the approved claims are generated automatically and the app will allow HR users to view these invoices.
- **Data Management of Lecturers:** It is assumed that Lecturers will notify HR of any changes of their personal information or contact details by sending HR an email (provided they use the same email they registered as Lecturer with). The HR view would have the necessary functionality to allow HR to successfully update the lecturer's personal and contact information. Once done with updating lecturer data, HR will send a confirmation email to the Lecturer.
- **Security:** The required security measures have been implemented to prevent unauthorized access to the database and uploaded documents. This would ensure only authorized users will have access.

5. Constraints:

- **Claim Submission Limit:** Once a claim is submitted, the lecturer cannot submit another one until the current one is either approved or rejected by a coordinator and manager.
- **File Size & Type Limit:** Only files up to a certain file size and of a common file type are allowed to be uploaded for each claim. This may be due to database storage constraints or application settings.
- **Database Performance:** Performance could be impacted when large numbers of documents are stored in the form of byte arrays in the database, especially if the database grows significantly in size.

UML Class Diagram:



Project Plan:

1. Project Phases

1. Phase 1: Planning and Requirements Gathering

2. Phase 2: System Design

3. Phase 3: Development

4. Phase 4: Testing

5. Phase 5: Deployment

6. Phase 6: Documentation and Handover

2. Tasks, Dependencies, and Timeline

- **Phase 1:** Planning and Requirements Gathering
 - **Task 1.1:** Define Requirements (2 days)
 - Identify user needs, key features, and technical requirements, by consulting the POE document and lecturer.
 - **Dependency:** None.
 - **Task 1.2:** Set Up Development Environment (1 day)
 - Install necessary tools such as:
 - Visual Studio - for MVC web app development with ASP.NET Core and C#.
 - SQL Server - for the hosting of the CMCS database.
 - **Dependency:** Completion of Task 1.1.

Total Time: 3 days

Phase 2: System Design

- **Task 2.1:** Design Database Schema (2 days)
 - Design tables, relationships, and data types for storing user data, claims and supporting documents. This can be done by drawing a UML Class Diagram and an ERD.
 - **Dependency:** Completion of Phase 1.
- **Task 2.2:** Design Application Architecture (3 days)
 - Define the MVC structure, including models (one for each type of user and one for Claims), views, and controllers (one for each model)
 - **Dependency:** Completion of Task 2.1.
- **Task 2.3:** Design User Interface Mockups (2 days)
 - Create mockups for the GUI layout (each wireframe representing a view, such 'Claims', 'Submit A Claim' and 'Manage Lecturer')
 - **Dependency:** Completion of Task 2.2.

Total Time: 7 days

Phase 3: Development

- **Task 3.1:** Implement Database (2 days)
 - Create the database for the CMCS app and implement the predefined schema.
 - **Dependency:** Completion of Phase 2.
- **Task 3.2:** Develop Models (3 days)
 - Implement the models to manage data for Claims, making provisions for storage of documents and ensuring documents are linked to the corresponding claims.
 - **Dependency:** Completion of Task 3.1.
- **Task 3.3:** Develop Controllers (5 days)
 - Implement the controllers (one for each: Claims, Lecturers, Coordinators, Managers and HR) to manage interactions between the user interface and the models.
 - **Dependency:** Completion of Task 3.2.
- **Task 3.4:** Develop Views (5 days)
 - Implement the user interface, by including forms for registering users, submitting, and displaying claims, as well as managing Lecturers.
- **Task 3.5:** Implement a File Size Limit (1 day)
 - Implement a control that limits the size of an uploaded file to a specific value in kilobytes.
 - **Dependency:** Completion of Task 3.4.
- **Task 3.6:** Implement a Common File Type Restriction (1 day)
 - Implement a functionality that prevents the lecturer from uploading documents in an unsupported format that cannot be stored in the database easier. Allow formats such as pdfs, docx or xls.
 - **Dependency:** Completion of Task 3.5.

Total Time: 17 days

Phase 4: Testing

- **Task 4.1:** Unit Testing (3 days)
 - Evaluate individual components (each model and each controller) to ensure they work as expected.
 - To name a few features to test: test whether submitting a claim, displaying claims, approving, or rejecting claims, managing lecturers, and generating reports work properly.
 - **Dependency:** Completion of Phase 3.

Total Time: 3 days

Phase 5: Deployment

- **Task 5.1:** Prepare Deployment Environment (1 days)
 - Set up the GitHub repository for deploying the application.
 - **Dependency:** Completion of Phase 4.

- **Task 5.2:** Deploy the Application (1 day)
 - Push the app to the GitHub repository and perform the final checks.
 - **Dependency:** Completion of Task 5.1.

Total Time: 2 days

Phase 6: Documentation and Handover

- **Task 6.1:** Write User Documentation (3 days)
 - Create documentation that explains how to use the app, for lecturers.
 - **Dependency:** Completion of Phase 5.
- **Task 6.2:** Write Technical Documentation (3 days)
 - Document the database, architecture, and deployment or pushing steps, for future reference or development or maintenance.
 - **Dependency:** Completion of Task 6.1.
- **Task 6.3:** Handover and Training (2 days)
 - Provide training or a handover session for the lecturers, coordinators, managers and HR staff and any other stakeholders.
 - **Dependency:** Completion of Task 6.2.

Total Time: 8 days)

Summary Timeline

- **Phase 1: Planning and Requirements Gathering:** 3 days.
- **Phase 2: System Design:** 7 days
- **Phase 3: Development:** 17 days
- **Phase 4: Testing:** 8 days
- **Phase 5: Deployment:** 2 days
- **Phase 6: Documentation and Handover:** 8 days
- **Total Project Duration:** 40 days (approximately 6 weeks)

References:

1. Open AI. (2024). ChatGPT (GPT 4.o) [Large Language Model](<https://chatgpt.com/>)