**PROG6212**

**POE-PART 1**

**ST10279488**

1. Documentation

Design Choices:  
I have chosen the Model-View-Controller architecture using .NET Core. By clearly separating concerns, the MVC design improves the system's scalability and maintainability. The Controller will handle requests, carry out business logic, and provide the user with appropriate responses. The Model will represent the database entities, such as `Lecturer`, `Claim`, and `Document`. The View will serve as the front-end interface for various users, such as lecturers, academic managers, and program coordinators.

(geeksforgeeks, 2024)

Database Structure:

To house all of the data pertaining to the Contract Monthly Claim System (CMCS), I decided to use a relational database. Relational databases are useful for managing organized data and securing connections between different entities. Entities like {Lecturer}, 'Claim', 'Document', and 'Status' are interrelated in this system. Data consistency and efficient query, update, and maintenance are guaranteed when SQL Server is used.

* Entities:
* Lecturer: Keeps track of lecturer details including name, email, and hourly rate.
* Claim: Includes information on the claim itself, including the number of hours worked, the date of submission, the status, and the professor who made the submission.
* Document: Oversees the routes to the files that instructors upload.
* Status: Indicates if the claim has been accepted by the Academic Manager and confirmed by the Program Coordinator.

User Interface (GUI) Layout:

* Lecturer's View:
* A dashboard where lecturers can enter the amount of hours worked, make claims, and upload supporting materials. There will be a button on the page that can be clicked to submit the claim.
* A section designed to monitor the status of claims, indicating whether they are "Pending," "Verified by Program Coordinator," or "Approved by Academic Manager."
* Views of the academic manager and program coordinator:
* A dashboard listing all claims pending approval or verification will be available. By checking each claim's information, including the hours worked and any supporting documentation, they can designate a claim as "Verified" or "Approved" with a single button click.
* Navigation:
* The sidebar maintains consistency and features navigation options like "Submit Claim," "View Submitted Claims," "Pending Claims," and "Approved Claims." This facilitates users' transitions between parts.

(geeksforgeeks, 2024)

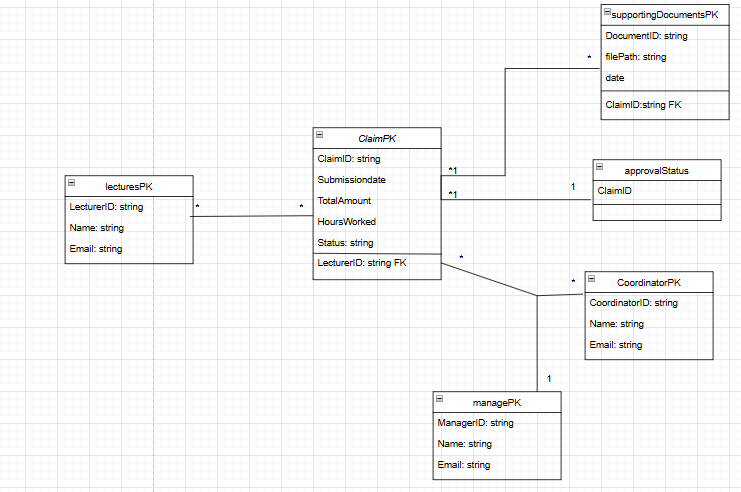
Assumptions:

* Each lecturer will have a unique ID that will be used to identify their claims;
* Lecturers will submit a single claim every month, covering all of the hours worked for that month.
* Until all necessary supporting papers, including timesheets, are uploaded, claims won't be processed.
* The roles of Academic Managers and Program Coordinators will be different:
* Program Coordinator: Confirms the accuracy of the statement.
* Academic Manager: Grants the ultimate clearance.

Constraints

* Time constraint: Claims from lecturers have to be submitted within a certain time frame.
* Data validation: Before approving a claim submission, the system must verify that all necessary information has been supplied.
* Role-based access: The only people who can see and approve claims are Academic Managers and Program Coordinators. The only claims available to lecturers will be their own.   
    
  These design choices guarantee that the system can handle claim submissions and approvals in an organised and effective way, and that it is scalable, dependable, and easy to use.

2. UML Class Diagram



3. Project Plan for Contract Monthly Claim System Prototype

Project Plan

Duration: 5 days

Phase 1: Research & Planning (Day 1)

* Tasks:
* Identify the needs of the project.
* Investigate MVC.NET Core GUI development.
* Determine which UML components are required.

Phase 2: UML Class Diagram (Day 2)

* Tasks:
* Identify the main entities.
* Establish connections between various entities.
* Create a UML class diagram.

Phase 3: Documentation of the Project Plan (Day 3)

* Tasks:
* Draft the project plan, including a schedule and work breakdown.
* Make chart to see the plan in action.

Phase 4: GUI Design (Day 4)

* Tasks:
* Create interface wireframes.
* Use.NET Core to construct the GUI prototype (WPF or MVC).

Phase 5: Evaluation and Complete Submission (Day 5)

* Tasks:
* Seek input on the UI design, project plan, and UML diagram.
* Make any necessary corrections and turn in any required documents.

Timeline

|  |  |
| --- | --- |
| Day 1 | Research & Planning |
| Day 2 | UML Class Diagram |
| Day 3 | Documentation of the Project Plan |
| Day 4 | GUI Design |
| Day 5 | Evaluation and Complete Submission |

(educative.io, 2024)

4. GUI Design for Contract Monthly Claim System

# References

(2024, 09). Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/mvc-framework-introduction/

(2024, 09). Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/what-is-graphical-user-interface/

(2024, 09). Retrieved from educative.io: https://www.educative.io/blog/how-to-plan-a-coding-project