### **1. Documentation of Design Choices**

#### **Design Choices for CMCS:**

* **Framework & Language:** The project will be built on.NET Core, notably Windows Presentation Forms (WPF) for the GUI, with C# as the major programming language. This choice is based on WPF's versatility and ease of GUI development, which enables for the creation of a responsive and modern interface.
* **Database Structure:** The database will use a relational format, including tables for Lecturers, Claims, Program Coordinators, Academic Managers, and Supporting Documents. We'll use SQL Server for database management because of its close connection with.NET.
* **GUI Layout:** The user interface will emphasize simplicity and clarity. Lecturers should be able to submit claims fast, and Coordinators and Managers should have simple ways to verify and approve these claims. File uploads for supporting documents will be made more intuitive, with clear indicators for claim monitoring.

#### **Assumptions:**

* Each lecturer has a unique identifier (LecturerID).
* Claims are tied to both the Programme Coordinator and Academic Manager for verification and approval.
* Each claim can have multiple supporting documents.
* Claims involve calculation based on the lecturer's **hours worked** and **hourly rate**.

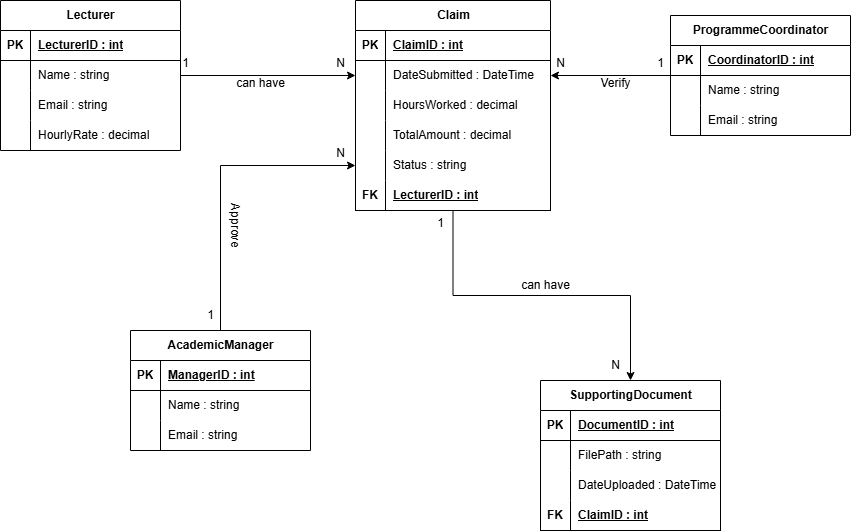
#### **Constraints:**

* Claims must be submitted within the calendar month.
* Only verified and approved claims can be processed for payment.
* There must be data validation to ensure no lecturer submits a claim exceeding hours worked in a month.

### **2. UML Class Diagram for Databases**

The UML class diagram will represent the database entities, attributes, and their relationships. Below is the uml diagram for the CMCS:

#### **Diagram:**



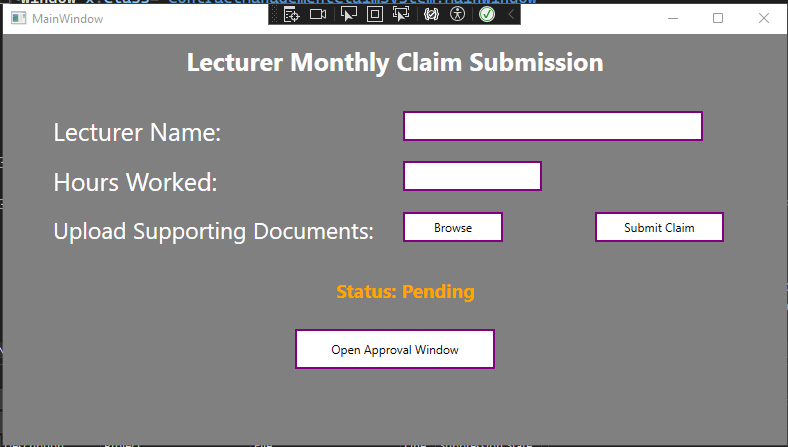
### **3. Project Plan**

The project will be broken down into the following tasks, with a realistic timeline of about **6 weeks (about 1 and a half months)**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Phase | Task | Duration | Dependencies | Description |
| 1. Planning & Requirements | Define Requirements and Project Scope | 3 days | None | Meet stakeholders to gather and finalize system requirements, constraints, and assumptions. Outline the project scope and deliverables. |
|  | Design UML Class Diagram | 2 days | Define Requirements | Develop a detailed UML class diagram outlining the data structure and entity relationships for the system. |
|  | Setup Development Environment | 1 day | None | Set up Visual Studio 2022 and ensure that all necessary libraries and dependencies are installed for WPF development. |
|  | Total for Phase 1 | 6 days |  |  |
| 2. Database Design | Design Database Schema (SQL Server) | 2 days | Define Requirements, UML Diagram | Convert the UML class diagram into a SQL Server database schema by specifying tables, columns, relationships, and data constraints. |
|  | Setup SQL Server Database | 1 day | Design Database Schema | Set up the SQL Server instance, create tables, and enforce schema-specific restrictions such as foreign key associations. |
|  | Total for Phase 2 | 3 days |  |  |
| 3. GUI Prototyping | Design Lecturer Interface (WPF) | 4 days | None | Design and construct the Lecturer UI in WPF so that lecturers can submit claims, upload documentation, and track claim status. |
|  | Design Coordinator/Manager Interface (WPF) | 4 days | Design Lecturer Interface | Create UI elements that allow Programme Coordinators and Academic Managers to view and approve/reject claims. |
|  | Link Interfaces Together | 2 days | Design Coordinator/Manager Interface | Connect the Lecturer and Coordinator/Manager user interfaces and handle window changes with WPF events and navigation. |
|  | Total for Phase 3 | 10 days |  |  |
| 4. Business Logic Implementation | Implement Claim Submission Logic | 4 days | Design Lecturer Interface | Implement back-end logic for claim submission, ensuring that claims are saved in the database and computations are performed using hours worked and hourly rate. |
|  | Implement Claim Verification Logic (Coordinator) | 3 days | Implement Claim Submission Logic | Implement the verification mechanism so that Programme Coordinators can assess claims and modify their status to "Verified" or "Rejected". |
|  | Implement Claim Approval Logic (Manager) | 3 days | Implement Claim Verification Logic | Implement the approval logic so that Academic Managers can approve or reject confirmed claims and update their statuses accordingly. |
|  | File Upload and Document Storage | 3 days | Implement Claim Submission Logic | Implement file upload logic for supporting documents, which will allow users to attach files to claims and save file locations in the database. |
|  | Status Tracking and Notifications | 2 days | Implement Claim Approval Logic | Implement claim status tracking to provide real-time updates on claim processing for lecturers, coordinators, and managers. |
|  | Total for Phase 4 | 15 days |  |  |
| 5. Testing & Refinements | Unit Testing for Claim Submission | 3 days | Implement Claim Submission Logic | Perform unit tests to check that the claim submission process works properly and that all inputs are validated. |
|  | Unit Testing for Verification/Approval | 3 days | Implement Verification/Approval Logic | Test verification and approval routines, including exceptions such as rejected claims or incomplete submissions. |
|  | UI Usability Testing | 2 days | Unit Testing | Test the user interface's usability and make changes based on user feedback or accessibility problems. |
|  | Final Refinements | 2 days | UI Usability Testing | Address any bugs or issues identified during testing and make necessary final refinements to the system. |
|  | Total for Phase 5 | 10 days |  |  |
| Total Project Duration |  | 44 days |  |  |

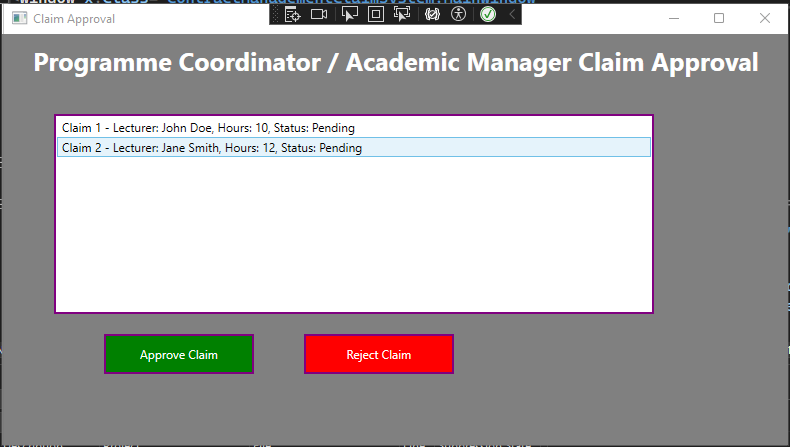
### **4. GUI (Windows Presentation Forms in .NET Core)**

For the GUI, we'll use **WPF (Windows Presentation Foundation)** to provide a simple, intuitive interface. Here are the designs with explanations:



#### **Lecturer Interface:**

* **Submit Claim Form**: A form for lecturers to register hours worked and attach supporting documentation.
* **Upload Button**: An intuitive button to upload documents.
* **Submit Button**: A button for submitting the claim, which triggers validation and saves it to the database.
* **Claim Status View**: A section that shows the status of submitted claims.
* **Open Approval Window:** A button that switches to the claims form.



#### **Programme Coordinator/Academic Manager Interface:**

* **Claim Verification & Approval Screen**: A list of claims pending verification or clearance. Each claim can be enlarged to show the hours worked, total amount, and any supporting documentation.
* **Approve/Reject Buttons**: Buttons for approving or rejecting claims, with explanations explaining why a claim may be refused.

#### **UI Design Considerations:**

* The system will provide real-time feedback on the claim status.
* Claims will show the current stage: **Submitted → Verified → Approved → Settled**.
* The interface will ensure accessibility, with clear labels and guidance.