**Introduction**  
This report outlines the design and prototype development for the Contract Monthly Claim System (CMCS), a .NET-based system aimed at streamlining monthly claim submissions and approvals for independent contractor lecturers. The prototype focuses on a non-functional front-end for a desktop application using Windows Presentation Foundation (WPF) in .NET, a UML class diagram for the database, and a project plan. The design prioritizes user-friendliness, efficiency, and scalability to address real-world administrative challenges in educational settings.

**Design Choices**  
The overall design adopts a modular architecture to separate concerns: user management, claim processing, and document handling. I chose WPF for the GUI because it allows for the rapid creation of rich, responsive desktop interfaces with precise control over layout and styling, which is ideal for a visual prototype. The colour scheme uses blues and grays for a professional and calming look, with intuitive navigation via a tabbed interface for different system functions (e.g., "Submit Claim," "Approve Claims," "Track Status"). The layout follows a grid-based structure to ensure consistency and alignment.

For the database, I opted for a relational model (e.g., SQL Server) to handle structured data efficiently. The key entities include User, Claim, and Document classes, with relationships designed to ensure data integrity and support complex queries for approvals and tracking. This choice aligns with the system's core requirements for accuracy and accountability.

**Database Structure**  
The database structure is defined by the following UML class diagram:

[Insert Rendered UML Image Here]

The diagram consists of three main classes:

* **User:** Represents all system users (Lecturers, Coordinators, Academic Managers). Its attributes include UserId (primary key), Name, Email, PasswordHash, and Role (e.g., "Lecturer", "Coordinator", "Manager").
* **Claim:** Captures the details of each monthly submission. Its attributes include ClaimId (primary key), UserId (foreign key linking to the submitting Lecturer), HoursWorked, HourlyRate (stored with the claim for historical accuracy), TotalAmount, Status (e.g., Pending, Approved, Rejected), SubmissionDate, and ApprovalDate.
* **Document:** Manages files uploaded to support claims. Its attributes include DocumentId (primary key), ClaimId (foreign key), FileName, FilePath (storing the location on the server), and UploadDate.

The relationships are one-to-many: one **User** (in the role of Lecturer) can submit many **Claims**, and one **Claim** can have many **Documents** associated with it. This structure ensures referential integrity and is optimized for queries such as retrieving all claims pending approval by a specific coordinator.

**GUI Layout**  
The WPF prototype features a main window with tabbed navigation, simulating access for different user roles. The "Lecturer Submit" tab includes TextBox controls for hours worked, hourly rate, and notes, a Button for file upload, and a prominent "Submit Claim" Button. The "Approver View" tab contains a ListView control displaying simulated claim data with "Approve" and "Reject" Buttons for each entry. The "Track Status" tab features a DataGrid showing claim history with status labels. The interface is visually cohesive and designed for clarity. No backend logic is implemented, as the focus is solely on the visual representation and user experience.

**Assumptions and Constraints**  
**Assumptions:** User authentication will be implemented later using the Email and PasswordHash fields. Claims are processed monthly, with TotalAmount calculated as HoursWorked \* HourlyRate. File uploads will be limited to common formats (e.g., .pdf, .docx, .xlsx) with a maximum size of 5MB.  
**Constraints:** This is a non-functional prototype; therefore, there is no database connectivity, real-time data processing, or business logic. Development is constrained to a .NET 8 environment. Security measures, such as data encryption, are out of scope for this initial phase.

**Project Plan**  
The project was executed according to the following plan:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Task ID | Task Description | Dependencies | Estimated Effort | Timeline | | 1 | Research POE Requirements | None | 2 hours | Day 1 | | 2 | Design UML Class Diagram | Task 1 | 3 hours | Day 1-2 | | 3 | Develop Project Plan & Timeline | Task 1 | 1 hour | Day 2 | | 4 | Build WPF GUI Prototype | Task 2 | 6 hours | Day 3-4 | | 5 | Compile Final Documentation | Tasks 2-4 | 3 hours | Day 5 | |  | **Total** |  | **15 hours** |  | |

This plan was designed to be realistic and achievable, with dependencies ensuring a logical workflow.

**Conclusion**  
In summary, this prototype establishes a solid foundation for the CMCS, emphasizing an intuitive user interface and a well-structured, scalable data model. The design choices made here will facilitate the implementation of full functionality in subsequent parts of the project.