**Introduction**

The report outlines the prototype development of the Contract Monthly Claim System (CMCS) as a .NET Core web application for independent contractor lecturer approval and monthly claim submission. The prototype includes non-functional aspects, for example, database design using UML class diagram, project plan, and GUI layout using ASP.NET Core MVC. The system aims to enhance efficiency since it allows lecturers to make claims, upload supporting documents and check status, and the aforementioned are accredited and certified by coordinators and managers.

**1. Documentation: Design Decisions, Database Schema, and GUI Design**

**Design Decisions**

I have employed ASP.NET Core MVC for GUI because it has a proven pattern for developing web applications with separation of concern (Model-View-Controller) where it is simple to maintain and scale. MVC supports clean routing, razor views for dynamic UI, and non-obtrusive integration with Entity Framework for database access. For the database, I chose a relational model with SQL Server to be able to handle inter-entity relationships like users, claims, and documents efficiently. User interface choices are made with simplicity in mind: Bootstrap and responsive design for mobile compatibility, business-like color scheme (blues and grays), and simple navigation to minimize user errors.

**Database Structure**

Database is centered on main entities: Lecturer (user info), Claim (submission details like hours, rate, and status), Document (documents), and Approver (managers/coordinators). There are one-to-many (approver claims) and many-to-one (one lecturer can have multiple claims) relationships. This is for data consistency and status query tracking. Assumptions: All users are authenticated using ASP.NET Identity; claims are computed as (hours \* rate). Constraints: Limited 5MB file uploading; no real-time notifications in-prototype; local development environment over cloud hosting assumed.

**GUI Design**

The GUI is a front-end static prototype with pages. The home dashboard has tabs labeled "Submit Claim," "Upload Documents," "Track Status," and "Approve Claims" (role-based). The Submit Claim page has a form with hours worked, hourly rate, notes, and submit buttons. Approval view shows pending claims in a table with approve/reject buttons. Layout incorporates navbar header, sidebar navigation, and footer status. Colors: Blue (#007BFF) as a main color for buttons, white background for readability purposes. The design makes it easy to flow through, i.e., large center submit button.

All these choices have been made to simulate actual admin systems, for a small user base (under 100 users) and simplicity in not over-engineering for the prototype stage.

**2. UML Class Diagram for Databases**

A screenshot of a computer screen

AI-generated content may be incorrect.

**3. Project Plan (Prototype Development )**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task No. | Task Description | Dependencies | Duration (Days) | Start Date | End Date | Responsible |
| 1 | Review POE requirements and define scope | None | 2 | Day 1 | Day 2 | Student |
| 2 | Identify assumptions and constraints | Task 1 | 1 | Day 3 | Day 3 | Student |
| 3 | Design UML class diagram (database structure) | Task 1 | 3 | Day 4 | Day 6 | Student |
| 4 | Draft initial GUI layout (wireframes/prototype screens) | Task 1 | 3 | Day 4 | Day 6 | Student |
| 5 | Document design choices (database, GUI, assumptions) | Tasks 2–4 | 2 | Day 7 | Day 8 | Student |
| 6 | Refine UML and GUI design based on feedback/self-review | Tasks 3–4 | 2 | Day 9 | Day 10 | Student |
| 7 | Create project report (Word doc) including plan, UML, GUI | Tasks 5–6 | 3 | Day 11 | Day 13 | Student |
| 8 | Push commits to GitHub (5 commits with messages) | Ongoing | Throughout | Day 1 | Day 13 | Student |
| 9 | Final review and submission of Part 1 deliverables | Tasks 7–8 | 1 | Day 14 | Day 14 | Student |