PART 1 Documentation

**1. Overview**

The Contract Monthly Claim System (CMCS) is a System that is there to make life easier for independent contractor lecturers by simplifying how they submit, approve, and manage monthly claims. For this first phase, I’m focusing on laying the groundwork—planning the project and building a prototype using the MVC design pattern in ASP.NET Core. This document covers my design decisions, database setup, assumptions, project timeline, and the initial GUI mockups.

**2. Why I Chose This Approach**

I went with the **Model-View-Controller (MVC)** pattern because it keeps things organized. It splits the database logic (Model), the user interface (View), and the behind-the-scenes processing (Controller) into separate pieces. This makes the system easier to maintain, scale, and troubleshoot down the line.

**Database Setup** The database revolves around four core components:

* **Lecturer**: Stores details like name, email, and banking info.
* **Claim**: Tracks claim details, including hours worked, hourly rate, total amount, and approval status.
* **Document**: Manages uploaded supporting files, with their file paths and upload dates.
* **UserRole**: Defines user types (Lecturer, Coordinator, Manager) to control access levels.

**How They Connect**:

* A single Lecturer can submit multiple Claims.
* Each Claim can include several Documents.
* Every Lecturer is tied to a specific UserRole for access control.

**GUI Design** I designed the interface to be intuitive and role-specific. Lecturers get a straightforward form to submit claims and upload documents. Coordinators see a clean list of pending claims with options to approve or reject them. Managers have a dashboard to keep tabs on all claim statuses. My goal was to make navigation feel natural and tailored to each user’s needs.

**3. Assumptions and Limitations**

**Assumptions**:

* Lecturers are already registered and can log in to submit claims.
* HR sets the hourly rates, so lecturers don’t have to input them manually.

**Limitations**:

* Only PDF, DOCX, and XLSX files are allowed for document uploads.
* For now, the prototype is just the front-end—there’s no database connection yet.

**4. UML Class Diagram**

(Insert UML diagram image in my final report.) The UML diagram maps out the database structure, showing how the Lecturer, Claim, Document, and UserRole entities are structured and linked.

**5. Project Timeline**

I’ve broken the work into weekly tasks to stay on track:

* **Week 1**: Set up the GitHub repo and basic MVC framework.
* **Week 2**: Create the UML diagram and finalize the database design.
* **Week 3**: Build the Lecturer’s claim submission interface prototype.
* **Week 4**: Finish the Coordinator and Manager interface mockups.
* **Week 5**: Wrap up the documentation, double-check the rubric, and prepare for submission.

This schedule keeps things manageable and ensures I meet deadlines.

**6. GUI Prototype**

The prototype includes three main views:

* **Lecturer View**: A simple form where lecturers enter hours worked, view their preset hourly rate, and upload files.
* **Coordinator View**: A list of pending claims with buttons to approve or reject them.
* **Manager View**: A dashboard summarizing claim statuses at a glance.

Right now, these are just visual mockups built in the MVC project’s Views folder, with no back-end functionality.

**7. Version Control**

I’m using **GitHub** to manage versions, with at least five commits planned for key milestones: project setup, adding the UML diagram, creating GUI prototypes, and finalizing documentation. I’ll use clear, descriptive commit messages to keep everything transparent.