Contract Monthly Clam System (CMCS)

Module: PROG6212

Part 1: Project Planning and Prototype Development

Author: Thembela Dakalo Memela

Date: 27/08/2025

1. Problem Statement & Scope

Independent Contract (IC) lecturers submit monthly clams for teaching-related work. Claims must be checked by a Program Co-Ordinator (PC) and approved by an Academic Manager (AM). The current process is manual and error prone. CMCS streamlines submission, verification, approval, and storage of supporting documents.

In Scope (prototype):

* Basic domain model & database design (UML/ER diagram).
* Non-functional GUI prototype in WPF (.NET Core) or ASP.NET Core MVC.
* Navigation stubs, sample screens, placeholder controls (no business logic).
* Project plan with milestones, WBS, and risks.

Out of Scope (Part 1):

* Authentication/authorization logic.
* Data persistence, validation, workflows, file I/O
* Calculations, emails, audit trails.

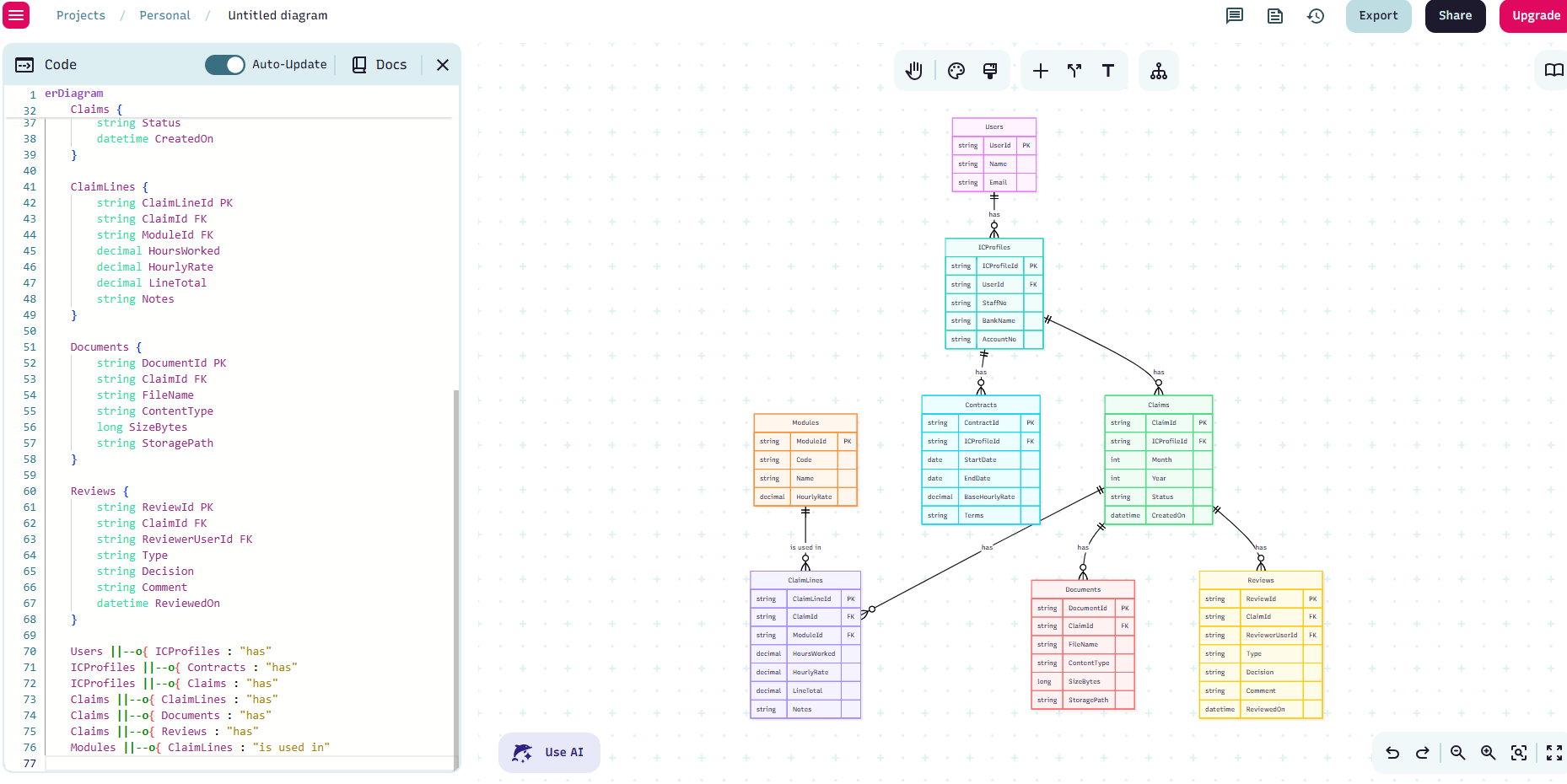
2. Stakeholders and Roles

* Independent Contractors (IC) - Creates and submits monthly claims with supporting documents.
* Program Co-Ordinator (PC) - Verifies accuracy, returns for corrections or recommends approval.
* Academic Manager (AM) - Final approval/rejection.
* System Admin (Admin) - Maintains users, rates, modules, and system settings.

3. High-Level Requirements (User Stories)

* As an IC, I want to capture hours and attach documents so that I can submit a monthly claim.
* As a PC, I want to review claims and leave comments so that I can approve or reject.
* As an AM, I want a clear summary of claim totals so that I can approve or reject.
* As an Admin, I want to manage users, modules, hourly rates, and IC contracts.
* As any user, I want a dashboard with my pending actions to work efficiently.

UML Diagram for Database



Project Plan:

Objective: To deliver a working prototype of the Contract Monthly Clain System within the allocated time frame, ensuring adherence to the module requirements.

Tasks and Dependencies

Task No. Task Description Dependency Duration

1 Requirements Gathering & Analysis None Week 1

2 Database Design (UML & ERD Diagrams) Task 1 Week 2

3 GUI Wireframes & Layout Design Task 1 Week 2

4 Prototype Development – GUI (WPF/MVC) Task 2, Task 3 Week 3–4

5 Documentation of Design Choices & Assumptions Task 1–3 Week 3

6 Integration of Diagrams & Documentation Task 5 Week 4

7 Review & Refinement of Prototype Pack (Part 1 submission) Task 6 Week 5

4. GUI/UI Design

Design Choice

I selected WPF (.NET Core) for the prototype because:

* It supports rich desktop interfaces with XAML.
* It's easier to prototype layouts visually requiring full back-end logic.
* It allows us to keep the design and align with MVC principles (UI separate from logic).

User-Centered Design Principles

* Consistency: Same navigation buttons across windows.
* Clarity: Clear labels (Submit, Approve, Reject, Report).
* Accessibility: Large fonts, simple layout minimal clutter.
* Role-based UI: Each user sees only what's relevant (Lecturer vs Coordinator vs Manager)

GUI Layout Overview

The CMCS interface will consist of the following main windows:

1. Login Window

* Fields: Username, Password.
* Buttons: Login, Exit.
* Role-based access: Lecturer, Program Coordinator, Academic Manager.

2. Dashboard Window

* Lecturer View:

1. Submit Claim (navigate to claim form).
2. View My Claim (see status of claims)

* Coordinator/Manager View:

1. Review Claims (approve/reject claims).
2. Reports (monthly/lecturer claim summaries).

3. Submit Claim Form (Lecturer)

* Fields:

1. Lecturer Name (auto-filled).
2. Hours Worked.
3. Hourly Rate (auto or manual)
4. Supporting documents (Upload)

* Buttons: Submit, Cancel.

4. Claims Review Form (Coordinator/Manager)

* Table/List of submitted claims with: Lecturer Name, Hours, Rate, Total, Status.
* Actions: Approve, Reject.
* Filter/Sort by Lecturer or month

5. Reports Window

* Generate Monthly Claim Reports.
* Export to PDF/Excel option (future enhancement).

6. Assumptions and Constraints

* Only staff with valid credentials can log in.
* The system will run on Windows environment (WPF) or hosted server (MVC).
* Authentication is handled via ASP.NET identity or simple DB authentication.
* Claims are always tied to valid contact.
* Passwords stored as hashes, not plain text.