POE PART 1

PROG6212

ST10441732

TSHIFHIWA THAMGANE

Contract Monthly Claim System (CMCS) – Part 1: Project Planning & Prototype

Documentation

1. Introduction

The Contract Monthly Claim System (CMCS) is a .NET Core-based application designed to simplify the process of submitting, verifying, and approving Independent Contractor (IC) lecturer claims. The prototype described here focuses on project planning, database design, GUI layout, and a non-functional front-end model. The design emphasizes usability, maintainability, and scalability, while addressing real-world administrative requirements such as claim accuracy, document handling, and approval workflows.

Design Choices and Assumptions

The project uses a three-tier architecture:

* Presentation layer (GUI): Built with WPF (MVVM) or MVC, providing user-friendly interaction.
* Business logic layer: Responsible for validating claims, calculating totals, and enforcing approval rules.
* Data layer: Relational SQL Server database for reliable storage and integrity.

**Key design assumptions:**

* Lecturers submit one claim per month linked to a valid contract.
* Hourly rates may differ across contracts and are locked for audit purposes.
* Claims require at least one supporting document (e.g., proof of work, timesheets).
* Approval is hierarchical: first Programme Coordinator, then Academic Manager.
* Claims statuses follow a lifecycle: Draft → Submitted → Under Review → Approved/Rejected → Settled.
* Security measures include role-based access, ensuring lecturers cannot approve claims.

**Constraints:**

* Prototype stage excludes authentication logic (login is mock-up only).
* No live database integration yet — only GUI placeholders.

2.UML Class Diagram



Relationships:

* A Lecturer has many Claims.
* A Claim has many ClaimItems and SupportingDocuments.
* Each Claim is associated with multiple ApprovalRecords.
* A Contract ties directly to a single Lecturer.

This structure ensures every claim and approval can be traced back to a lecturer and contract, supporting accountability.

3.Project Plan

Timeline

|  |  |
| --- | --- |
| Weeks | Tasks |
| Week 1 | Requirements analysis, UML diagram |
| Week 2 | Database schema design, repo setup |
| Week 3 | GUI wireframes (lecturer, coordinator, manager views) |
| Week 4 | WPF/MVC front-end layouts |
| Week 5 | Document upload prototype + claim status tracker |
| Week 6 | Usability review, refine GUI, documentation |

**Dependencies**

The following dependencies must be managed to ensure smooth project progression:

* Database Design → Application Prototype
* The relational database schema (tables, relationships, keys) must be finalized before binding to the GUI prototype.
* UML Diagram → Development Workflows
* Accurate UML diagrams are required to align team members on system scope and data flows.
* GUI Wireframes → WPF/MVC Prototype
* Wireframes must be validated before implementing screens in WPF/MVC.
* Version Control → Team Collaboration
* GitHub repository setup must be completed before collaborative development starts.
* Approval Workflow → Role-Based Interfaces
* Lecturer, Coordinator, and Manager dashboards depend on the completion of role-based access logic.
* Testing → Final Prototype Delivery
* Unit and usability testing can only begin once front-end mockups and navigation flow are stable.

**Deliverables**

The following deliverables will be produced across the project timeline

* Documentation Deliverables
* UML Class Diagram of CMCS database
* Technical report outlining design choices, constraints, and assumptions
* Project plan (tasks, dependencies, timeline, deliverables)
* Prototype Deliverables (Front-End Only, No Logic Yet)
* Lecturer GUI: Claim submission form, upload function, claim history viewer
* Coordinator GUI: Verification dashboard, claim status tracker
* Manager GUI: Approval dashboard, comments/feedback functionality
* Common GUI: Login page, navigation menus, claim status visibility
* Version Control Deliverables
* GitHub repository with at least 5 commits
* Descriptive commit messages showing project evolution
* Review Deliverables
* Usability feedback from peers/lecturers
* Submission-ready PoE report with diagrams, screenshots, and documentation