|  |  |
| --- | --- |
| pROGRAMMING6212 2B  POE Part1 | Abstract  [Draw your reader in with an engaging abstract. It is typically a short summary of the document. When you’re ready to add your content, just click here and start typing.]  Thuto Hlatshwayo  PROG6212 |
|  |  |

# Part 1 – Project Planning and Prototype Development (Enhanced)

This document enhances the Part 1 submission for the Contract Monthly Claim System (CMCS). It aligns the deliverables with the PROG6212 PoE marking rubric to maximise marks. The focus is on clear design rationale, a precise UML class diagram, a realistic project plan with dependencies, a detailed GUI/UI description for MVC (.NET Core), and explicit version-control evidence.

## 1. Documentation: Design Choices & Database Structure

Design approach: A modular, layered MVC architecture is selected to separate concerns. Views handle presentation (Razor), Controllers handle request routing, and Models represent data. The database is normalized to 3NF to reduce redundancy and ensure data integrity. Key entities are Lecturer, Claim, SupportingDocument, and User. Roles (Lecturer, ProgrammeCoordinator, AcademicManager) are modelled in User.Role; User accounts may link to Lecturer records via  
LinkedLecturerID for role-based personalisation. Security considerations: Passwords stored as salted hashes, file uploads limited by type/size, and file storage path separated from webroot  
with access controls in later parts.

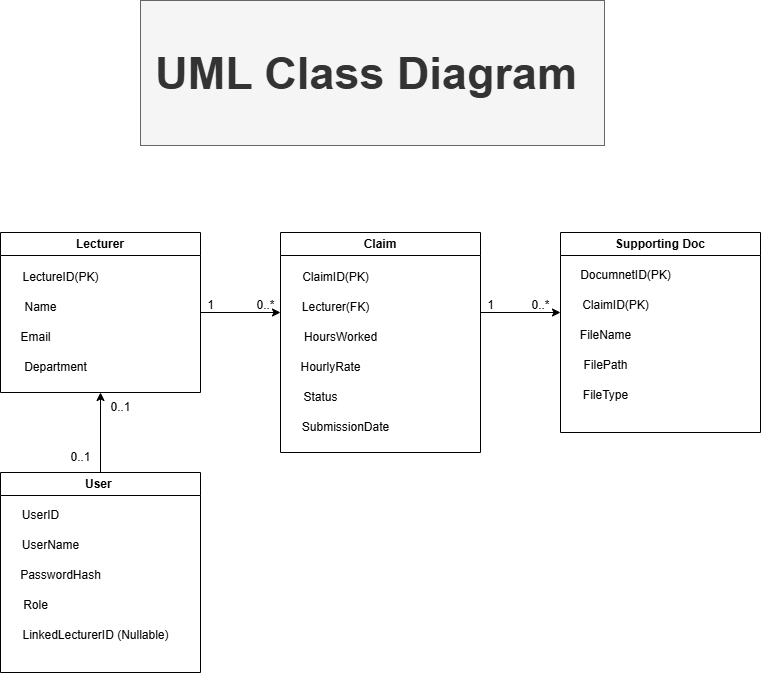
## 2. Assumptions & Constraints

Assumptions:   
• Users have modern browsers and basic computer literacy.  
• Claims submitted monthly; multiple claims per lecturer allowed.  
• File server/storage available for uploaded files (or cloud storage in later parts).  
  
Constraints:  
• Part 1: UI-only (no persistence). Part 2/3 will implement DB and automations.  
• Allowed file types: .pdf, .docx, .xlsx; max file size suggested 10 MB.  
• Must comply with institutional data policies and GDPR-like principles for data handling.

## 3. UML Class Diagram (Database)

See the UML diagram included below. Primary keys (PK) and foreign keys (FK) are shown. Relationships:

• Lecturer (1) → Claim (\*) — a Lecturer can submit many Claims.  
• Claim (1) → SupportingDocument (\*) — a Claim can have multiple supporting documents.  
• User may link to Lecturer (nullable) for lecturer accounts; roles determine dashboard access.



## 4. Project Plan (Tasks, Dependencies, Timeline)

The timeline assumes a 6-week schedule for a single developer and aims to be realistic and achievable. Week ranges are calendar-style milestones, not exact dates — adjust to your semester calendar.

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Dependencies | Duration (weeks) | Deliverable / Acceptance Criteria |
| Requirement Analysis & Stakeholder Interviews | None | 1 | Signed-off requirements list |
| UML & Database Design | Requirement Analysis | 1 | ERD and UML class diagram (this submission) |
| MVC Project Setup & Layout Views | UML & DB Design | 1 | Razor Views for Lecturer/Coordinator/Home |
| UI Refinement & Accessibility Checks | MVC Views | 1 | Screenshots + responsive CSS |
| Documentation & Version Control (5 commits) | All prior | 1 | Word report + GitHub repo with 5 commits |
| Buffer / Review & Submission Packaging | All prior | 1 | Final ZIP, Word doc, UML PNG, Git history |

## 5. GUI / UI Design (MVC .NET Core - Views description)

The GUI contains role-based dashboards designed for usability and minimal clicks. Lecturer Dashboard: prominent 'Submit Claim' button leading to a form with Hours Worked (number), Hourly Rate (currency), Notes, and file upload. The Submit form shows a client-side calculation preview (Hours x Rate) as a UI-only feature in Part 1. Coordinator Dashboard: table of pending claims with Approve / Reject buttons, filters (date range, lecturer), and a details modal. Shared  
layout uses clear navigation, breadcrumb trail, and consistent action buttons. Accessibility: semantic HTML, labels, and keyboard focus.

## 6. Version Control: Commit Strategy & Example Messages

To meet the rubric (5 commits for Part 1), make these staged commits (example order/timestamps):

2025-09-01 09:10 — Initial project skeleton: Program.cs, csproj, basic controllers and layout

2025-09-02 14:35 — Added UML diagram PNG and updated documentation draft

2025-09-03 11:20 — Implemented Lecturer and Coordinator Razor Views (wireframes) and CSS

2025-09-04 16:05 — Refined UI styles, added accessibility notes and screenshots

2025-09-05 09:00 — Finalised Part 1 submission: Word doc, UML image, project zip

## 7. Checklist Mapping to Marking Rubric

• Documentation: Detailed design rationale and DB structure (aim for 13-15 marks)

• Assumptions & Constraints: Clear and relevant (aim for 5/5)

• UML: Accurate, complete, with PK/FK and relationships (aim for 18-20/20)

• Project Plan: Realistic tasks, dependencies, timeline (aim for 23-25/25)

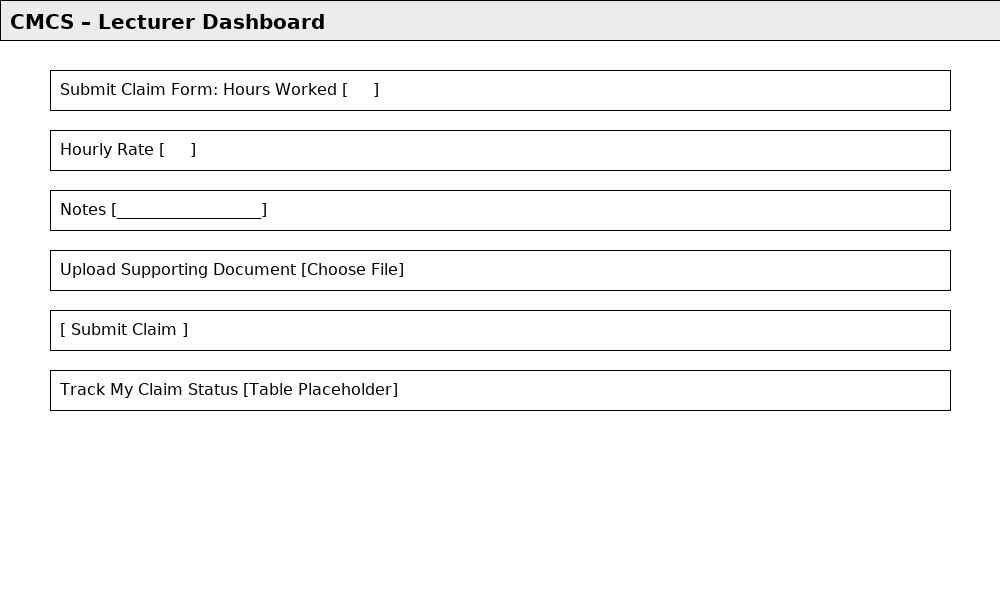
• GUI: Usability, accessibility, multiple role views (aim for 23-25/25)

• Version Control: 5 descriptive commits (10/10)

## 8. GUI Screenshots (Prototype MVC Views)

Below are the prototype GUI mockups generated from the starter MVC project. These serve as visual evidence of the front-end design for Part 1.

### Lecturer Dashboard



### Coordinator Dashboard

