**MUDAU MULWELI PEGGY**

**PROG6212**

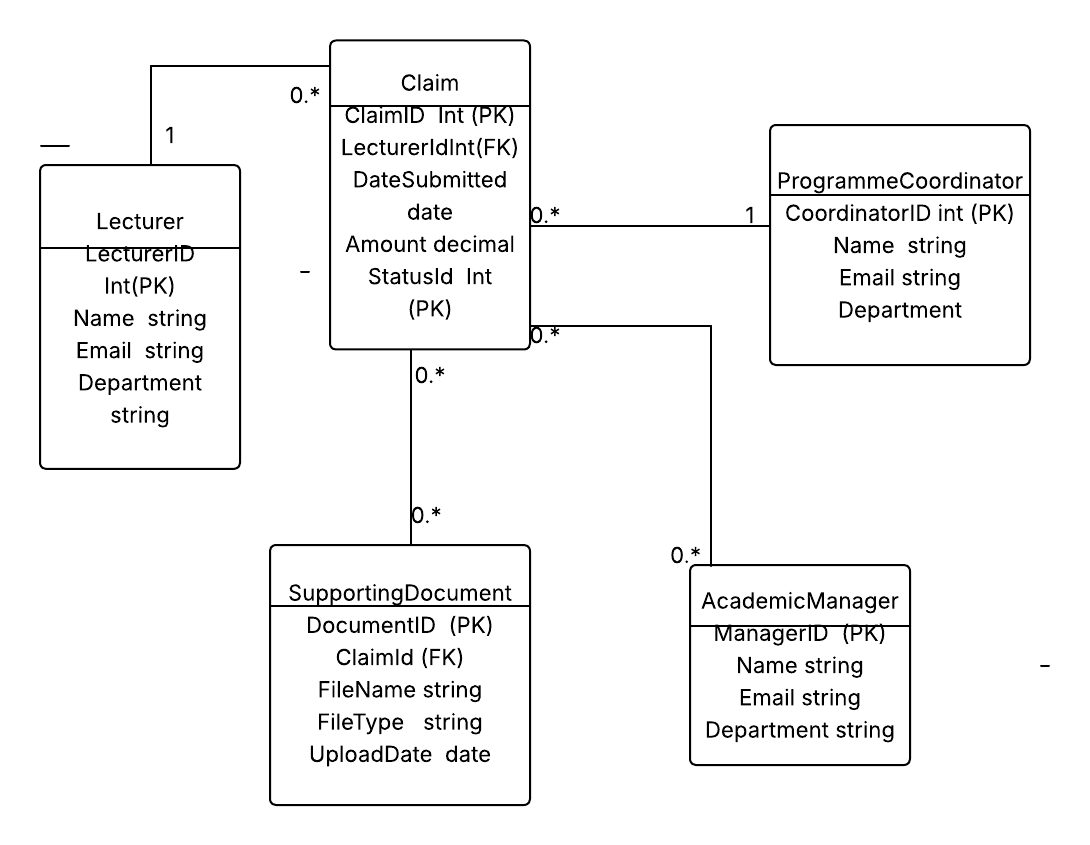
**Contract Monthly Claim System (CMCS) Prototype Report**

**1. Introduction**

The Contract Monthly Claim System (CMCS) is a prototype .NET WPF application developed to streamline the monthly claim process for independent contractor lecturers. The aim is to reduce manual errors, speed up approvals, and provide a transparent platform for claimants and approvers. The system supports three roles: Lecturers, Programme Coordinators, and Academic Managers. Each role has specific functions, ranging from claim submission to verification and approval.

The prototype focuses on the front-end user interface, using a dark theme with neon blue and white highlights to create a professional and user-friendly experience. This ensures that users can navigate the system with ease, while also making the workflow visually engaging and consistent.

**2. UML Class Diagram**

****

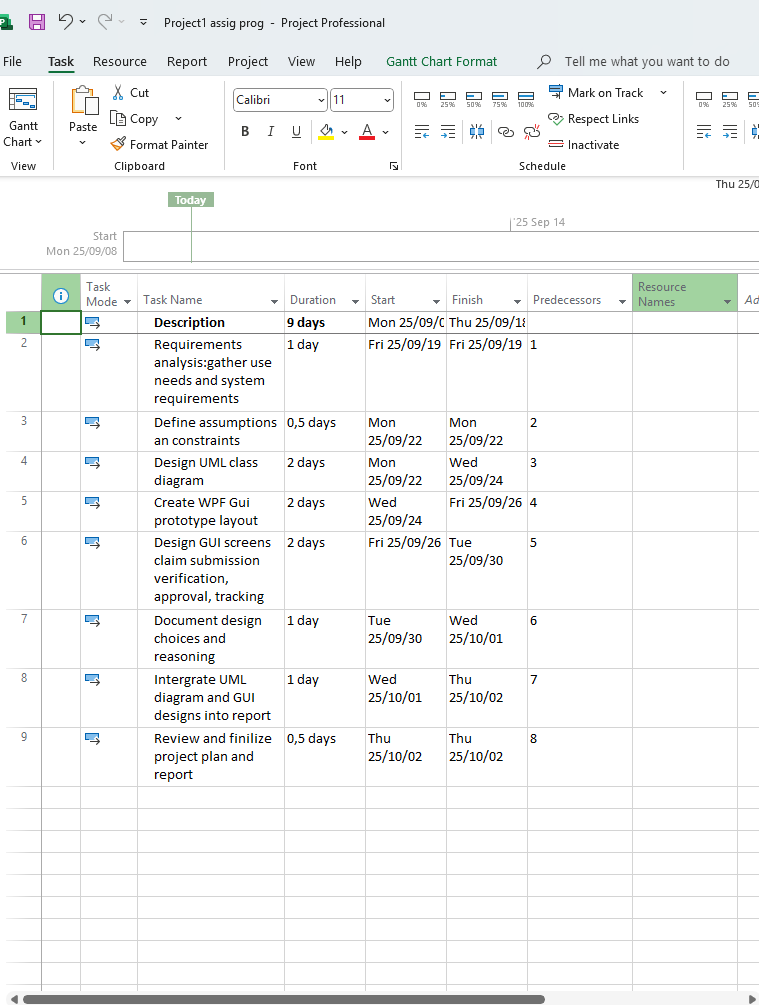
The UML class diagram illustrates the structure of the CMCS database and highlights relationships between different entities. At the center is the **Claim** class, which records submission details such as claim ID, date, amount, and status. Claims are linked to lecturers through Lecturer ID, ensuring accountability and traceability. Each lecturer can submit multiple claims, but every claim belongs to only one lecturer.

The **Supporting Document** class provides flexibility by allowing multiple files to be attached to a single claim. Its attributes include *Filename, Filetype,* and *Upload Date,* ensuring proper documentation of evidence for each claim.

Two reviewer roles are included: **Programme Coordinator** and **Academic Manager.** Each role contains identifiers and contact information and can oversee multiple claims. This supports the multi-level approval process, where claims are first checked by coordinators and then finalized by managers. The diagram also makes use of primary and foreign keys to maintain data integrity and multiplicity notation (1, 0..\*) to reflect real-world relationships.

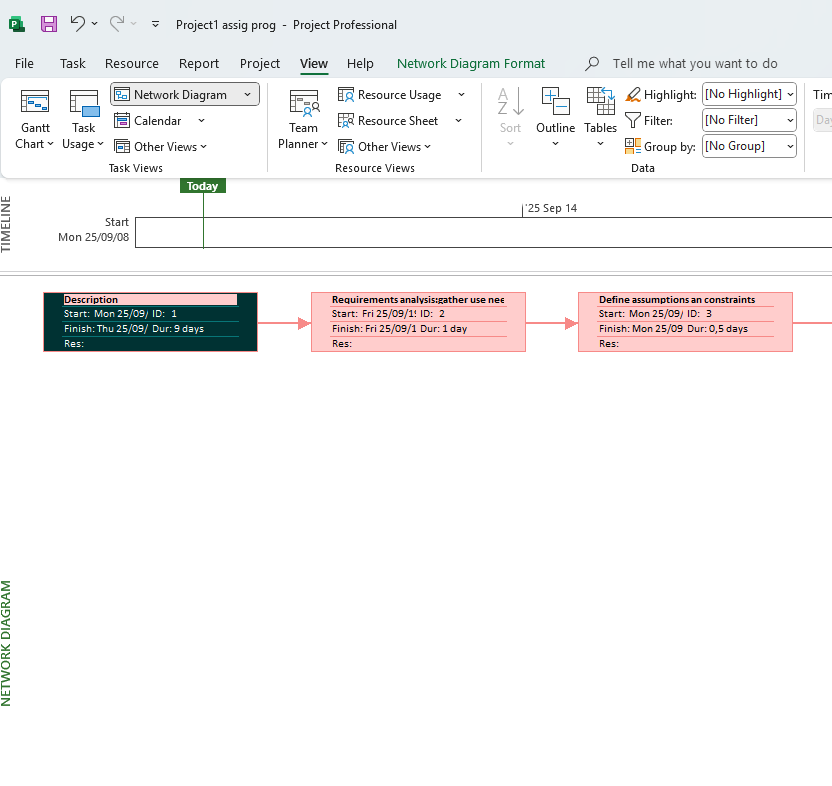
This structure ensures that claims, documents, and reviews are clearly connected, allowing for a transparent and auditable workflow.

**3. Project Plan**

****

**A screenshot of a computer

AI-generated content may be incorrect.**

****

**A screenshot of a computer

AI-generated content may be incorrect.**

**Objective**

To design a WPF prototype that enables lecturers to submit claims, attach documents, and allows coordinators and managers to approve them.

**Scope**

**In-scope:** UML diagram, GUI prototype, project plan, and documentation.  
**Out-of-scope:** Functional backend, live database, and automation.

**Tasks and Timeline**

* T1: Requirements analysis – 1 day
* T2: Define assumptions – 0.5 day
* T3: UML design – 2 days
* T4: GUI layout – 2 days
* T5: GUI screens for each role – 2 days
* T6: Document design choices – 1 day
* T7: Compile report – 1 day
* T8: Final review – 0.5 day

**Total:** 9 days

**Milestones**

Requirements (Day 1), UML diagram (Day 3), GUI prototype (Day 5), Final report (Day 9).

**Risks**

Possible confusion in GUI layout (mitigated by consistent styling) and delays in UML design (mitigated through early drafts).

**Gantt Chart View**

* Visual timeline of the tasks with bars representing task duration.
* Arrows indicate dependencies between tasks.
* Shows task progress and schedule from late September to early October 2025.
* You can see task dependencies and how tasks flow from one to the next.

**Image 3: Network Diagram View**

* Tasks are shown as boxes connected by arrows representing dependencies.
* This diagram focuses on the logical relationships and task order.
* The first task ("Description") is at the start, and each subsequent task follows based on predecessors.
* The timeline at the top shows the start and finish dates of the whole project.

**Conclusion**

The CMCS prototype provides a clear structure for claiming submission and approval through its UML diagram and GUI design. Although limited to the front-end, it demonstrates a realistic workflow that reduces errors, ensures transparency, and sets the foundation for future functional development.