Ethan Jansen – ST10440914

Prog POE Part 1

Table of Contents

[Github Link 2](#_Toc209015852)

[Documentation 3](#_Toc209015853)

[UML Class Diagram 5](#_Toc209015854)

[Project Plan 6](#_Toc209015855)

# Github Link

https://github.com/Rockslide9959/PROG6212POE

# Documentation

The Claim Management System is designed to streamline the submission and approval of claims for lecturers within an academic institution. The objective of this prototype is to provide them with a visual representation of the user interface and system flow. We wont be implementing backend logic at this stage. The prototype is developed using ASP.NET Core MVC to allow for structured separation of concerns and future scalability

Design Choices

Technology Stack: ASP.Net Core MVC was selected for this project. It has a structured approach, allows you to easily create views and controller and has good integration compatibility with future backend services

Architecture: The model-view-controller pattern ensures a clean separation of concerns. Even though models and controllers currently only hold placeholder logic, the architecture allows for easy expansion in part 2

Database Representation: The UML class diagram models essential entities required to support claim submission and approval. My core classes were:

* Lecturer(LecturerID, Name. Email, HourlyRate)
* Claim(ClaimID, LecturerID, HoursWorked, TotalAmount, Status, DateSubmitted)
* Document(DocumentID, ClaimID, FileName, FilePath, UploadDate)
* ClaimApproval(ApprovalID, CoordinatorID, ClaimID, ManagerID, Role, Decision, DateApproved)
* ProgrammeCoordinator(CoordinatorID, Name, Email)
* AcademicManager(ManagerID, Name, Email)

User Interface Design: The GUI emphasizes simplicity and clarity. Each user role has an intuitive interface. Lecturers can submit claims via a form and upload supporting documents. Coordinators and Managers have dashboards listing pending claims with approve/reject buttons. All roles can track claim statuses via a detected claims page.

Assumptions:

* The institution follows a fixed hourly rate policy for claims. The exact rate is not defined in this phase
* Each lecturer submits one claim per month. This contains all relevant hours worked
* Document upload is limited to pdf or image formats but my prototype does not perform file validation
* There is a stable internet connection for users since the prototype assumes web based deployment

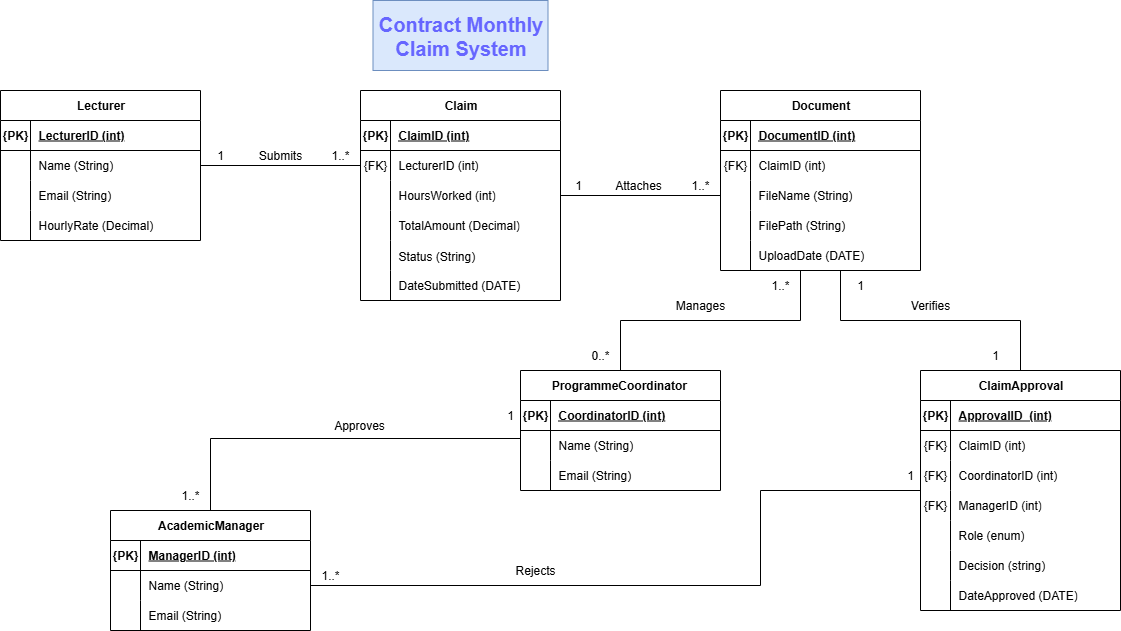
Constraints

* No backend logic: part 1 focuses only on UI and design. Database connectivity and processing will be added in part 2
* Time bound prototype: the interface must be functional enough for stakeholder feedback but without advanced styling or responsive design at this stage.
* Version control: a minimum of five commits with meaningful messages is required to demonstrate development progression
* Security features such as authentication and authorization is not included in part 1. It will be necessary in later phases however.

Gui Prototype Overview

* Login Page: For role based navigation
* Lecturer Dashboard: Add claim form with fields and file upload section
* Manager Dashboard: Pending claims table with approve and reject options
* Claims Tracking Page: Displays the current status of the claims

# UML Class Diagram



# Project Plan

|  |  |  |  |
| --- | --- | --- | --- |
| **Day** | **Task** | **Description** | **Deliverable** |
| **Day 1** | **Project Initiation & Requirements Analysis** | Review POE brief, identify requirements (lecturer submissions, approvals, document uploads, claim tracking). Define scope and assumptions. | Initial requirements notes |
| **Day 2** | **Database Design Planning** | Identify entities (Lecturer, Claim, Document, ClaimApproval, Coordinator, Manager). Define attributes, PKs/FKs, data types. | Draft entity list & attributes |
| **Day 3** | **UML Class Diagram Development** | Create UML class diagram (ASCII and/or tool-based). Show classes, attributes, PKs/FKs, multiplicities. | UML class diagram |
| **Day 4** | **GUI Layout Design** | Sketch mockups for Lecturer, Coordinator/Manager dashboards, and Claim Status page. Ensure usability and intuitive layout. | GUI wireframes |
| **Day 5** | **Project Plan Documentation** | Write documentation explaining design choices, constraints, and rationale. Add assumptions (e.g., only PDF uploads, fixed hourly rate). | Documentation draft |
| **Day 6** | **Version Control Setup & Initial Commits** | Create GitHub repository, push UML diagram, wireframes, documentation. Make 2–3 meaningful commits with descriptive messages. | GitHub repo with commits |
| **Day 7** | **Final Report Compilation & Submission** | Combine documentation, UML diagram, GUI mockups, and project plan into one structured Word document. Make final GitHub commit. | Final Part 1 submission |