IIEMSA

POE – Part 1

PROG6212

ST10395938
10-9-2025

Table of Contents

Design Choice

The design of the web application is intended to incorporate all the necessary elements required for a seamless claims management workflow. The primary goal is to allow lecturers to submit claims, which are then reviewed and approved sequentially by the Programme Coordinator and the Academic Manager. This structured workflow ensures proper verification and accountability at each stage of the approval process.

The application is developed using the Model-View-Controller (MVC) architectural pattern, which was chosen for its ability to separate the application into three interconnected components: the Model, which handles data; the View, which manages the user interface; and the Controller, which processes user input and coordinates interactions between the Model and the View (Code Academy, 2025).

Database Structure

The database consists of three main classes: Claims, Programme Coordinator, and Academic Manager. The Claims class stores all lecturer-submitted information, including month, year, hours worked, hourly rate, additional notes, and supporting documents. Each claim is linked to the Programme Coordinator through a foreign key and is assigned a unique primary key for tracking within the approval workflow. Once the Programme Coordinator approves a claim, it is forwarded to the Academic Manager, whose class contains its own primary key and foreign keys referencing both the claim and the associated coordinator. This relational structure maintains data integrity, provides a clear audit trail, and effectively supports the sequential approval process while keeping the database design simple and organised.

GUI Design

The web application features an intuitive and easy-to-navigate interface. The homepage provides options to submit, track, or approve claims, accessible via a persistent navigation bar. Lecturers submit claims through a form capturing month, year, hours worked, hourly rate, additional notes, and supporting documents, then return to the homepage. The Track view displays claim statuses in a dynamic table showing responses from both the Programme Coordinator and Academic Manager. Coordinators review pending claims in a table view with Approve and Reject options; rejections require a reason before returning to the dashboard. The Academic Manager follows a similar workflow but can only act on claims already approved by the coordinator, with rejection also requiring a note. The interface emphasises usability, a clear sequential workflow, and efficient, transparent claim processing (W3Schools, 2025).

Layout Design

The web application layout emphasises simplicity, clarity, and ease of navigation. The homepage provides quick access to submitting, tracking, and approving claims, with a consistent top navigation bar. Submission forms are vertically structured with clearly labelled fields and prominent action buttons. Tracking and approval views use responsive tables to display claim details and statuses, with action buttons placed for easy access. Overall, the design prioritises readability, usability, and responsiveness, enabling efficient task completion across devices while maintaining a professional interface (W3Schools, 2025).

Assumptions or Constraints

Assumptions

The system assumes the existence of three distinct user roles: Lecturer, Programme Coordinator, and Academic Manager, each with predefined responsibilities. Lecturers are assumed to submit claims, Programme Coordinators approve or reject claims, and Academic Managers approve or reject claims only after the coordinator's approval. It is also assumed that claims will follow a strict sequential workflow, meaning a claim cannot reach the Academic Manager until it has been approved by the Programme Coordinator.

The database design assumes that each claim has a unique identifier (primary key), and the relationships between claims, Programme Coordinators, and Academic Managers are properly maintained using foreign keys. Users are expected to provide all required claim information correctly, including month, year, hours worked, hourly rate, additional notes, and supporting documentation. Uploaded documents are assumed to be in valid formats.

The system also assumes that users will follow the intended navigation paths, moving from the homepage to submitting, tracking, or approving claims, and that approvals and rejections will be reflected immediately in the tracking table.

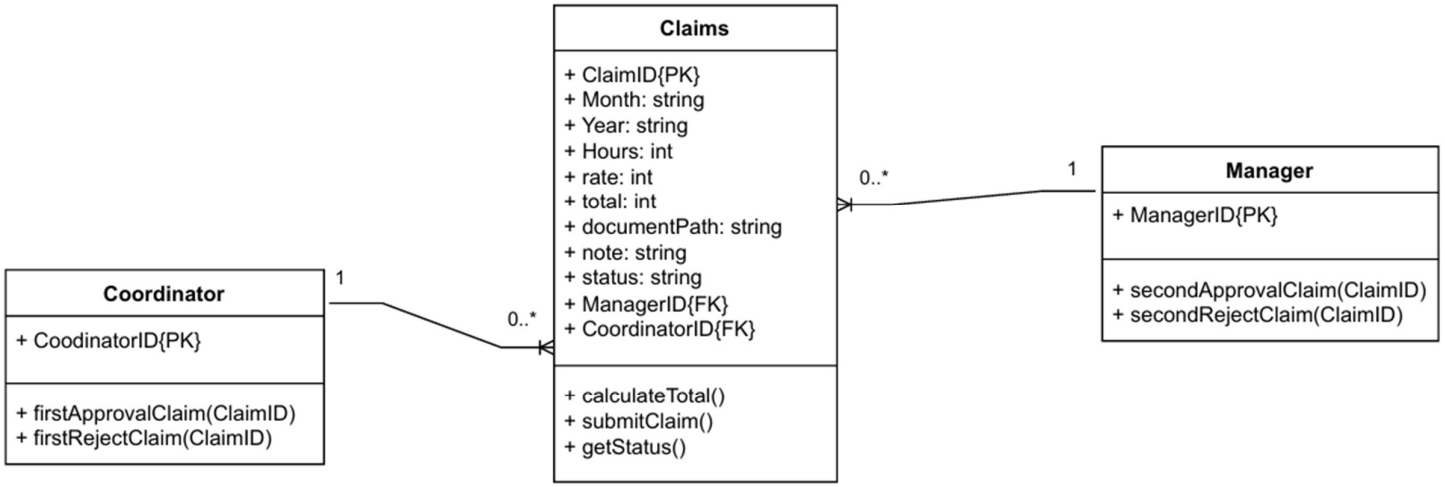
Constraints

The system is constrained by the sequential approval process, whereby a claim cannot be processed by the Academic Manager unless it has first been approved by the Programme Coordinator.

Functional constraints ensure that only users with the Programme Coordinator role can access the coordinator approval page, and only Academic Managers can access the academic approval page. Rejected claims must include a rejection note before the workflow can continue. Navigation is constrained such that users must follow the defined workflow; for instance, after submitting a claim, the user is redirected to the homepage, and skipping steps in the approval process is not allowed.

The GUI is constrained to capture all required claim information before submission, display claim statuses in a dynamic table for tracking, and provide mandatory approve and reject options for approvers.

UML Diagram

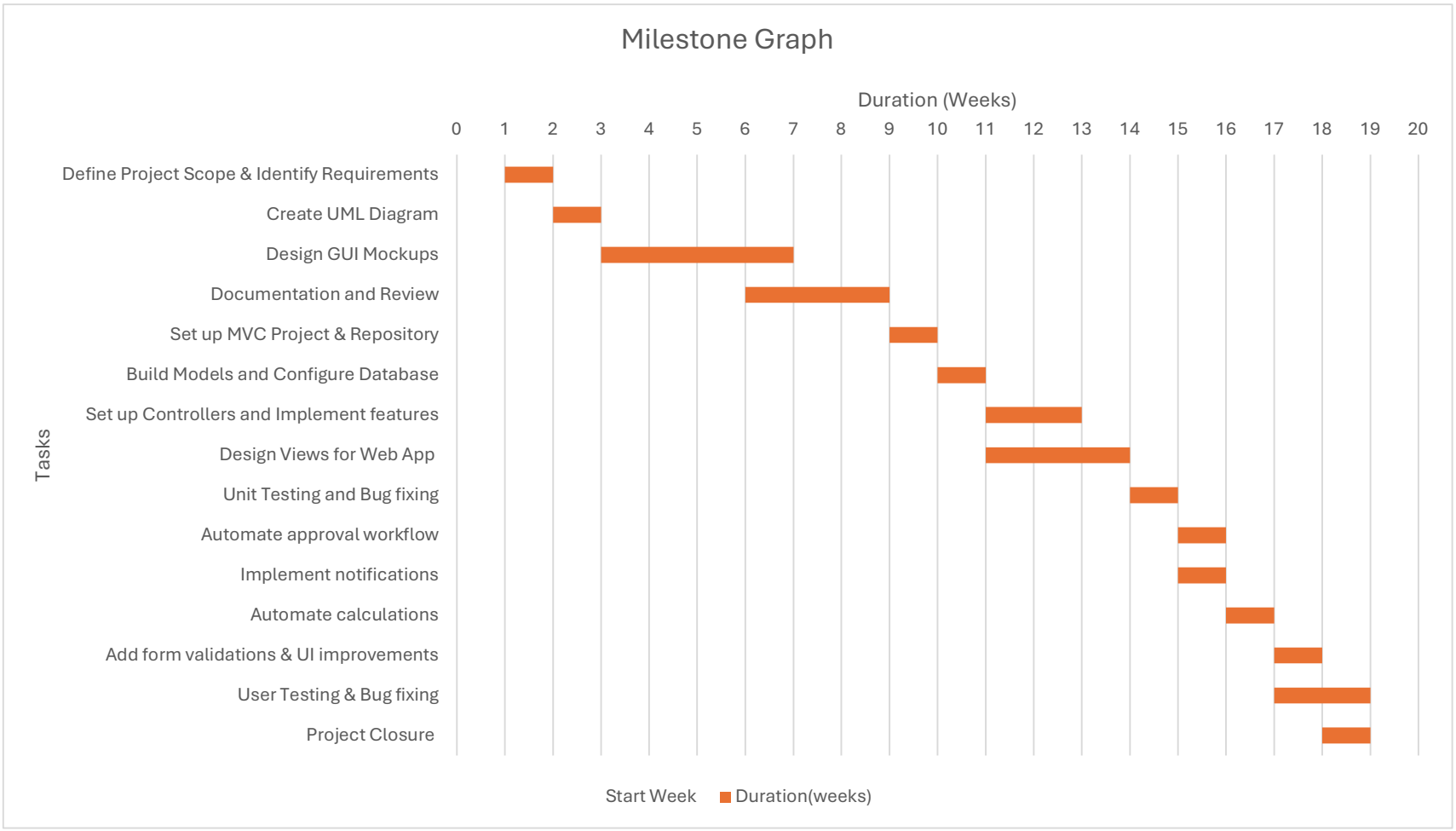

Claims
+ ClaimID{PK}
+ Month: string
+ Year: string
+ Hours: int
+ rate: int
+ total: int
+ documentPath: string
+ note: string
+ status: string
+ ManagerID{FK}
+ CoordinatorID{FK}
+ calculateTotal()
+ submitClaim()
+ getStatus()
Coordinator
+ CoodinatorID{PK}
+ firstApprovalClaim(ClaimID)
+ firstRejectClaim(ClaimID)
Manager
+ ManagerID{PK}
+ secondApprovalClaim(ClaimID)
+ secondRejectClaim(ClaimID)
1
0..*
0..*
1

Project Plan

Milestone Date Table

Task ID	Task	Start Date	Start Week	Duration (weeks)	Dependencies
1.1	Define Project Scope & Identify Requirements	21-Jul	1	1	N/A
1.2	Create UML Diagram	28-Jul	2	1	1.1
1.3	Design GUI Mock-ups	04-Aug	3	4	1.2
1.4	Documentation and Review	25-Aug	6	3	1.4
2.1	Set up MVC Project & Repository	22-Sept	9	1	1.4
2.2	Build Models and Configure Database	24-Sept	10	1	2.1
2.3	Set up Controllers and Implement features	29-Sept	11	2	2.2
2.4	Design Views for Web App	15-Oct	11	3	2.3
2.5	Unit Testing and Bug fixing	20-Oct	14	1	2.4
3.1	Automate approval workflow	27-Oct	15	1	2.5
3.2	Implement notifications	30-Oct	15	1	2.5
3.3	Automate calculations	04-Nov	16	1	2.5
3.4	Add form validations & UI improvements	10-Nov	17	1	3.3
3.5	User Testing & Bug fixing	14-Nov	17	2	3.4
3.6	Project Closure	17-Nov	18	1	3.4

Gantt Chart


Milestone Graph
Duration (Weeks)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Tasks
Define Project Scope & Identify Requirements
Create UML Diagram
Design GUI Mockups
Documentation and Review
Set up MVC Project & Repository
Build Models and Configure Database
Set up Controllers and Implement features
Design Views for Web App
Unit Testing and Bug fixing
Automate approval workflow
Implement notifications
Automate calculations
Add form validations & UI improvements
User Testing & Bug fixing
Project Closure
Start Week
Duration(weeks)

Repository Link

https://github.com/Ochwo-Anthony/ST10395938_PROG6212_POEPart1

References

Code Academy, 2025. *MVC Architecture Explained: Model, View, Controller.* [Online]
Available at: https://www.codecademy.com/article/mvc-architecture-model-view-controller
[Accessed 16 September 2025].

W3Schools, 2025. *W3Schools.* [Online]
Available at: https://www.w3schools.com/Css/
[Accessed 16 September 2025].

Declarations

AI was used to grammatically paraphrase the content of the Design Choice.

AI was used to help create the theme for the layout of the webapp.

AI was used in formatting the view pages to fit the created theme.