PROGRAMMING 2B

PROG6212

PORTFOLIO OF EVIDENCE

NOTHANDO MPOFU
ST10440515

Table of Contents

1. INTRODUCTION 2
2. Project Planning 3
3. System Design Choices 5
4. Database Design 6
5. Graphical User Interface (Prototype) 8
6. Assumptions and Constraints 14
7. Conclusion 15
References 16

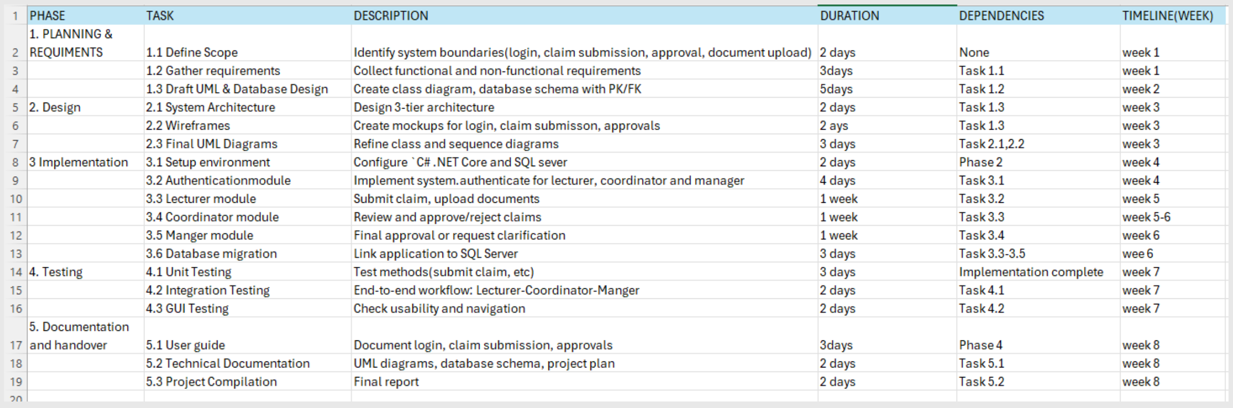
1. INTRODUCTION

The **Contract Monthly Claim System (CMCS)** is designed to streamline the process of managing lecturer claims within an academic institution. Traditionally, the process of submitting, verifying, and approving monthly claims has relied on manual paperwork and email communication, which is often time-consuming, prone to errors, and difficult to track. CMCS addresses these challenges by providing a centralized digital platform where lecturers can submit their claims, coordinators can review and verify submissions, and academic managers can finalize approvals.

The purpose of CMCS is to ensure efficiency, transparency, and accountability in the claims process. By digitizing claim submissions and incorporating layered verification steps, the system reduces administrative workload, minimizes the risk of fraud or oversight, and ensures that lecturers are compensated fairly and promptly for their work.

From an industry perspective, CMCS reflects the growing demand for digital transformation in higher education administration**.** Universities and colleges increasingly rely on web-based systems to handle finance, human resources, and academic processes, making solutions like CMCS both relevant and practical. The system not only improves operational workflows but also aligns with modern institutional goals of accuracy, compliance, and sustainable resource management.

2. Project Planning



PHASE	TASK	DESCRIPTION	DURATION	DEPENDENCIES	TIMELINE(WEEK)
1. PLANNING & REQUIMENTS	1.1 Define Scope	Identify system boundaries(login, claim submission, approval, document upload)	2 days	None	week 1
	1.2 Gather requirements	Collect functional and non-functional requirements	3days	Task 1.1	week 1
	1.3 Draft UML & Database Design	Create class diagram, database schema with PK/FK	5days	Task 1.2	week 2
2. Design	2.1 System Architecture	Design 3-tier architecture	2 days	Task 1.3	week 3
	2.2 Wireframes	Create mockups for login, claim submisson, approvals	2 ays	Task 1.3	week 3
	2.3 Final UML Diagrams	Refine class and sequence diagrams	3 days	Task 2.1,2.2	week 3
3 Implementation	3.1 Setup environment	Configure `C# .NET Core and SQL sever	2 days	Phase 2	week 4
	3.2 Authenticationmodule	Implement system.authenticate for lecturer, coordinator and manager	4 days	Task 3.1	week 4
	3.3 Lecturer module	Submit claim, upload documents	1 week	Task 3.2	week 5
	3.4 Coordinator module	Review and approve/reject claims	1 week	Task 3.3	week 5-6
	3.5 Manger module	Final approval or request clarification	1 week	Task 3.4	week 6
	3.6 Database migration	Link application to SQL Server	3 days	Task 3.3-3.5	wee 6
4. Testing	4.1 Unit Testing	Test methods(submit claim, etc)	3 days	Implementation complete	week 7
	4.2 Integration Testing	End-to-end workflow: Lecturer-Coordinator-Manger	2 days	Task 4.1	week 7
	4.3 GUI Testing	Check usability and navigation	2 days	Task 4.2	week 7
5. Documentation and handover	5.1 User guide	Document login, claim submission, approvals	3days	Phase 4	week 8
	5.2 Technical Documentation	UML diagrams, database schema, project plan	2 days	Task 5.1	week 8
	5.3 Project Compilation	Final report	2 days	Task 5.2	week 8

2.1 Methodology

The project followed an **Agile development approach**, as it supports flexibility and gradual progress. Instead of completing the system in one long cycle, development was broken down into smaller phases where feedback could be applied early. This allowed adjustments to be made to the lecturer, coordinator, and manager features as they were designed, ensuring the solution stayed aligned with real needs.

2.2 Resources

To deliver the CMCS, a combination of technologies, tools, and frameworks were used:

- **Core Technologies:** ASP.NET MVC for application structure, SQL Lite for database management, and HTML/CSS with Razor for interface development.
- **Frameworks & Libraries:** Entity Framework was used to handle database interactions, while custom CSS was used for professional UI.
- **Supporting Tools:** GitHub provided version control and collaboration throughout the project.

2.3 Risks, Assumptions, and Constraints

- **Risks:** There was a chance of setbacks such as losing cloud resources, encountering delays if modules weren’t integrated smoothly, or low user adoption if the design wasn’t simple enough.
- **Assumptions:** It was assumed that all users (lecturers, coordinators, and managers) would have stable internet access, basic computer literacy, and institutional support for a digital claims workflow.
- **Constraints:** The project was limited by time and scope, focusing mainly on the user interface rather than full backend integration. Dependence on Azure also introduced a constraint, as availability of cloud resources was not guaranteed.

3. System Design Choices

The design of the Contract Monthly Claim System (CMCS) was shaped by the need to create a structured, transparent, and user-friendly platform that mirrors the real-world processes of claim submission, verification, and approval within academic institutions. A key decision was to separate the system into role-based dashboards for lecturers, coordinators, and academic managers. This separation ensures that each user interacts only with functionality relevant to their responsibilities, thereby reducing complexity and minimizing the risk of unauthorized actions.

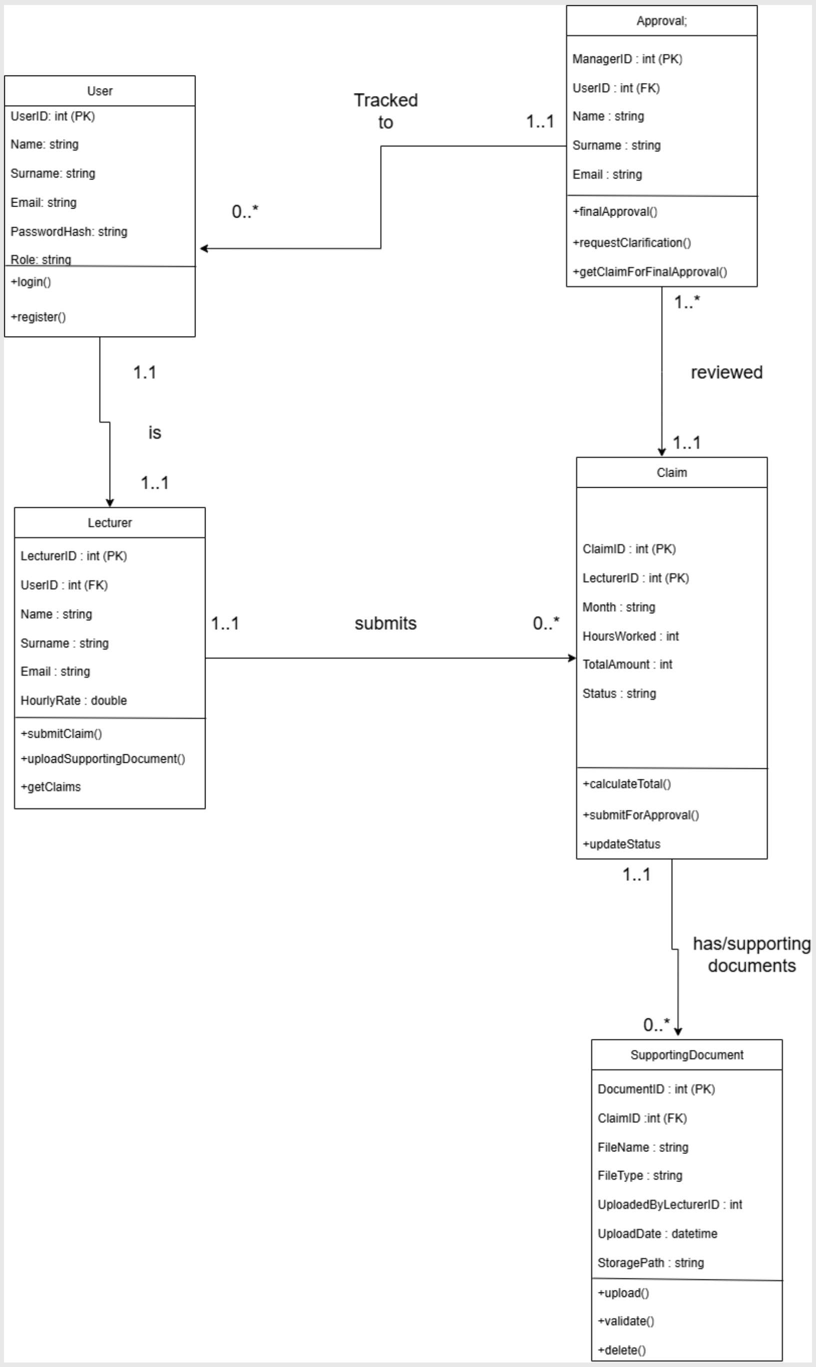
The user interface follows a modern, consistent theme across all views, with lecturers submitting claims through simple, form-based inputs while coordinators and managers review claims presented in neatly styled white cards. These cards display lecturer details, claim information, and the status of supporting documents, complemented by visually distinct colored badges (such as Pending, Accepted, Rejected, or Verification Required), which improve clarity and allow quick decision-making at a glance. Action buttons like Review, Accept, Reject, and Further Verification were included to streamline the approval workflow and align with actual administrative procedures.

On the backend, the database structure was carefully designed to support this process by organizing data around three core entities: Lecturers, Claims, and Reviews. Each Lecturer entity holds personal details and hourly rate information, Claims are linked to lecturers to record the hours worked, total amount, and uploaded supporting files, while Reviews capture the decision trail of coordinators and managers to maintain accountability and transparency.

This relational structure not only enforces data integrity but also allows scalability as more lecturers and claims are added to the system. By combining a clean, role-driven interface with a logical, relational data model, the system supports the full lifecycle of claim handling submission by lecturers, verification by coordinators, and approval or rejection by managers while maintaining usability, accuracy, and traceability across the institution.

4. Database Design

The database design of the Contract Monthly Claim System (CMCS) follows a **relational model** to ensure data integrity, scalability, and ease of querying. The system revolves around four main entities: **Lecturers, Claims, Documents, and Approvals**, which together support the full workflow of claim submission, verification, and approval.



Entities & Relationships:

- A **Lecturer** can submit many **Claims** (1-to-Many).
- Each **Claim** may have one or more **Documents** uploaded as supporting evidence (1-to-Many).
- Each **Claim** undergoes multiple **Approvals** (from both Coordinator and Manager), forming a 1-to-Many relationship between Claims and Approvals.
- The **Approval** entity ensures role-specific accountability by recording the decision (Accepted, Rejected, Pending, or Further Verification).

Data Attributes & Constraints:

- **Primary keys (PKs)** ensure unique identification of lecturers, claims, documents, and approvals.
- **Foreign keys (FKs)** enforce referential integrity, linking claims to lecturers, documents to claims, and approvals to claims.
- **HourlyRate** must be a positive decimal value.
- **HoursWorked** must be a non-negative integer.
- **Status** of claims is restricted to controlled values (e.g., Pending, Accepted, Rejected, VerificationRequired) using either an enum or lookup table.
- **FileType** is constrained to accepted formats (.pdf, .doc, .docx) for compliance with institutional requirements.

Rationale for Data Model:
The relational model was chosen because it provides strong **data consistency** and supports **transactional workflows** such as claim submission and approval, which require reliable records. By normalizing the data into distinct entities, the system avoids redundancy (e.g., storing lecturer details repeatedly in each claim) while maintaining flexibility for future expansion, such as adding new roles (e.g., Finance Officer) or additional document types. The separation of approvals into a dedicated entity ensures that the decision history is fully traceable, which is critical for transparency and audit purposes in a real-world academic institution. This design aligns with industry best practices for enterprise systems where accountability, scalability, and secure handling of documents are required.

5. Graphical User Interface (Prototype)

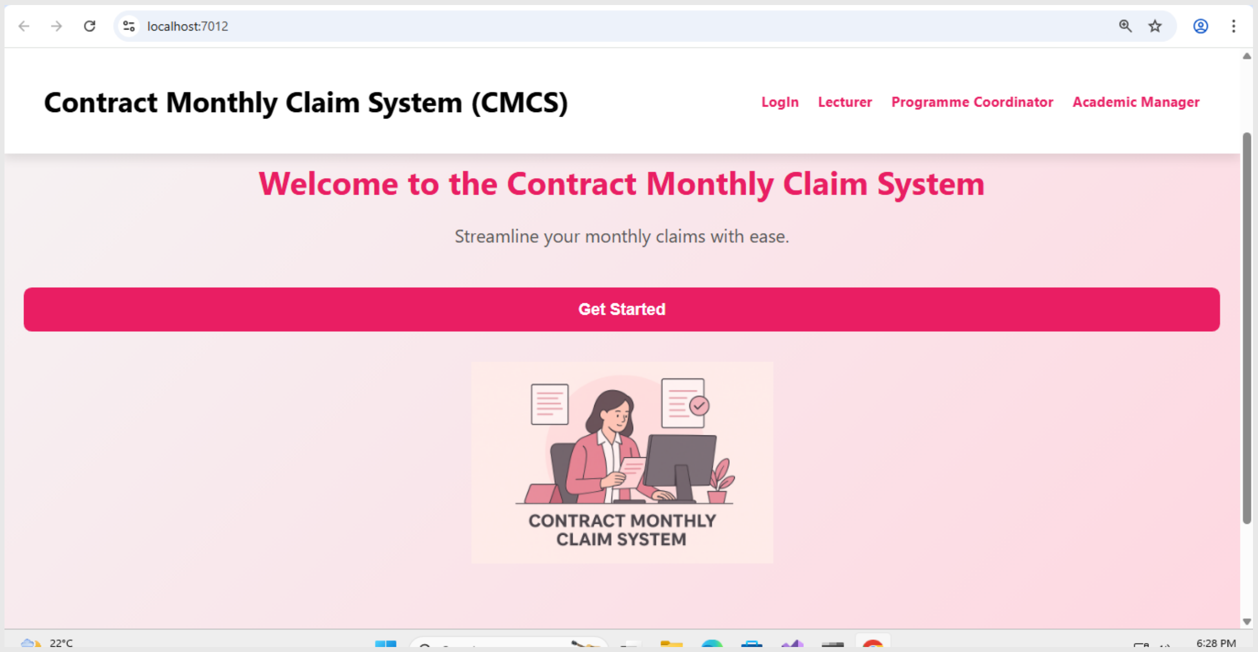
The CMCS was developed using **ASP.NET MVC**, which separates the user interface from backend logic while allowing dynamic content rendering. The prototype emphasizes clarity, ease of use, and consistent navigation across all user roles.

Key Screens

1. **Home / Landing Page** – Introduces the system and provides a "Get Started" button to navigate to the login page.
2. **Login / Sign-Up Page** – Allows first-time users to register and returning users to log in. The page uses toggle buttons for switching between login and registration forms.
3. **Lecturer Dashboard** – Displays personal details, claim status, and a "Submit Claim" button. Badges indicate whether a claim has been uploaded, and its review status.
4. **Claim Submission Page** – Contains a form for hours worked, hourly rate, total amount (auto-calculated), and file upload for supporting documents.
5. **Coordinator & Academic Manager Dashboards** – Show multiple lecturers in card-style blocks. Each card displays lecturer information, claim status, supporting documents, and action buttons (Review, Accept, Reject, Further Verification).
6. **Review Page** – Displays claim details, hours, hourly rate, and supporting documents to allow coordinators or managers to make decisions.

The CMCS was designed with usability, consistency, and accessibility at the forefront to ensure an intuitive and efficient experience for all users. **Usability** is achieved through clear and logically organized navigation menus, prominent buttons, and straightforward workflows that guide lecturers through claim submission and coordinators or managers through review and approval processes. Forms include descriptive placeholders, labels, and input validations to minimize errors and make the system easy to complete even for users with limited technical experience. **Consistency** is maintained across all dashboards and pages through a uniform card-based layout, standardized font styles, and a cohesive color scheme featuring pink highlights, white information cards, and colored status badges. This visual uniformity helps users quickly understand the interface, locate relevant information, and interact with actions such as submitting or reviewing claims without confusion. **Accessibility** was carefully considered by selecting readable font sizes, appropriately sized buttons, and high-contrast color combinations to support users with visual impairments or limited dexterity. Important information, like claim statuses, is emphasized using clearly distinguishable colored badges, and all interactive elements are designed to be easily identifiable and operable, ensuring that the system is inclusive and user-friendly for all staff members.

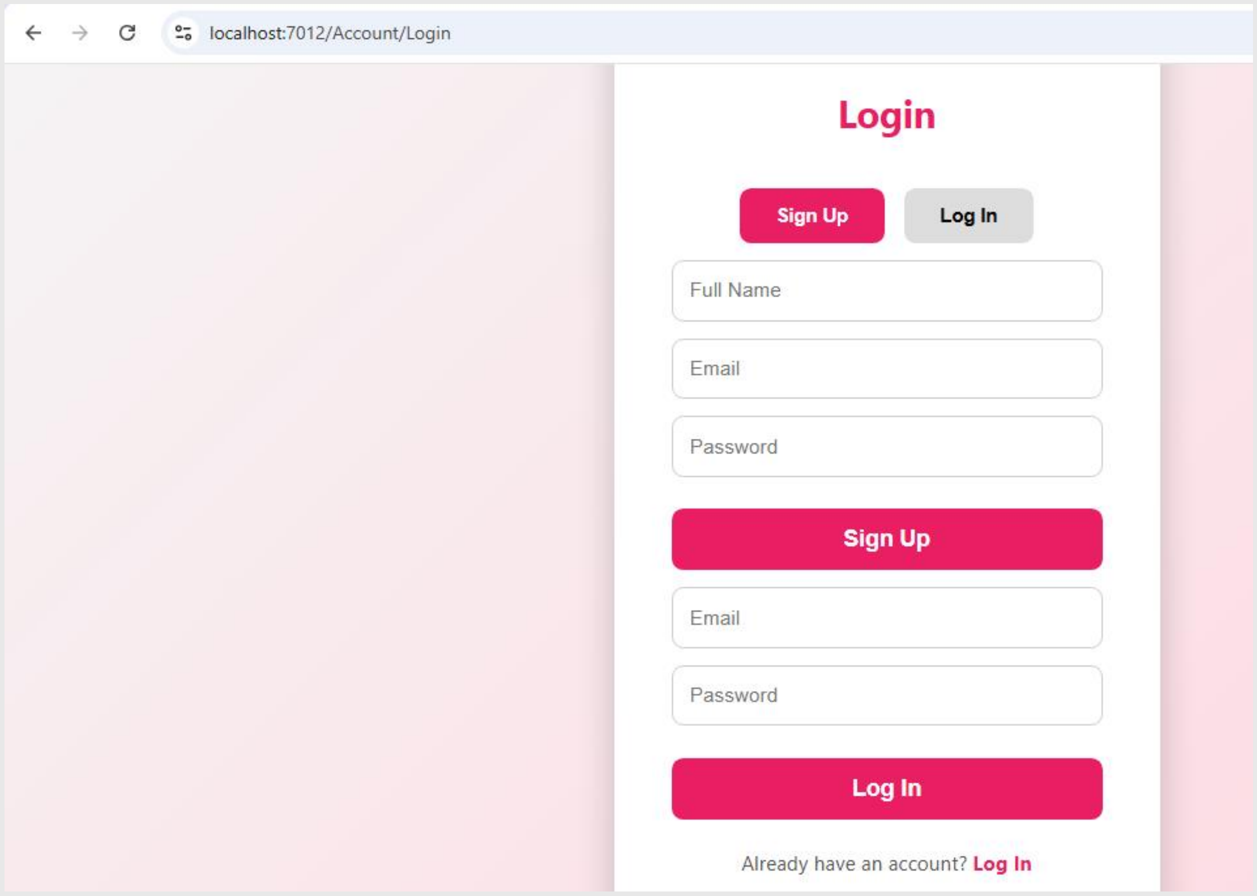
WIREFRAMES


localhost:7012

Contract Monthly Claim System (CMCS)

Welcome to the Contract Monthly Claim System

Streamline your monthly claims with ease.


localhost:7012/Account/Login

Login

Sign Up | Log In

Full Name

Email

Password

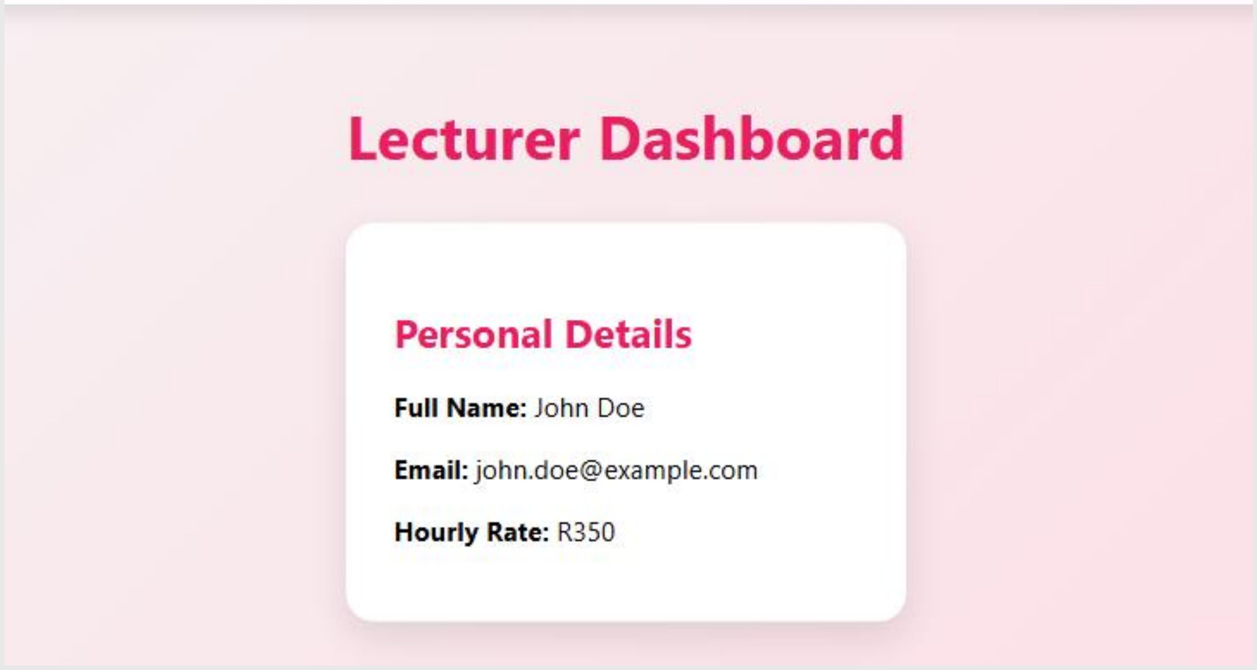
Sign Up

Email

Password

Log In

Already have an account? **Log In**

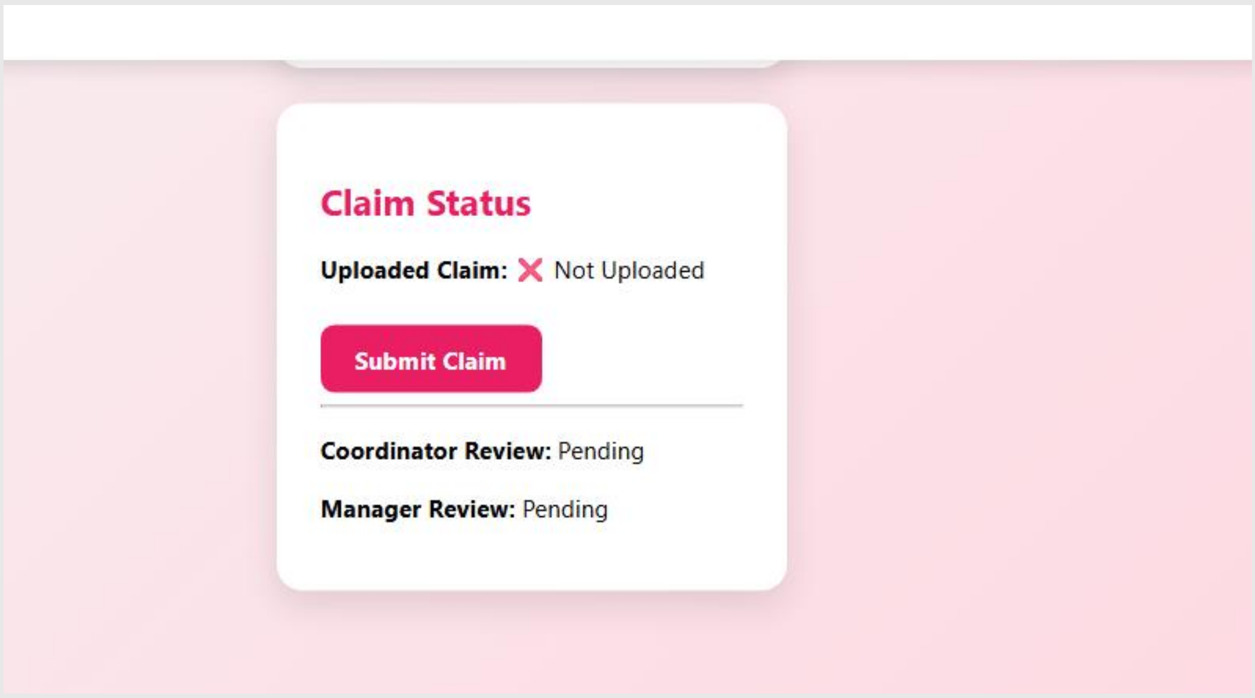

Lecturer Dashboard

Personal Details

Full Name: John Doe

Email: john.doe@example.com

Hourly Rate: R350

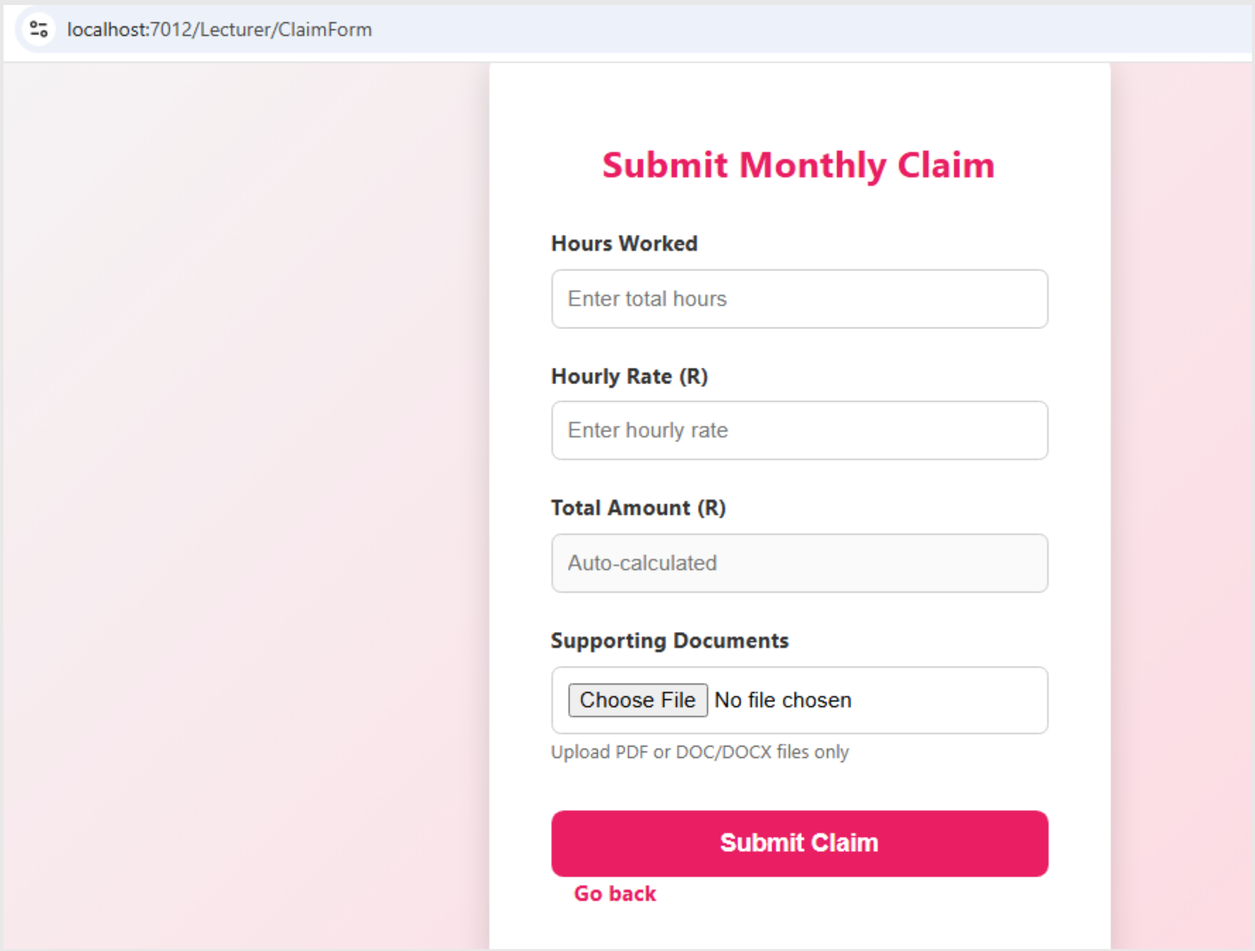


Claim Status

Uploaded Claim: ❌ Not Uploaded

Coordinator Review: Pending

Manager Review: Pending



localhost:7012/Lecturer/ClaimForm

Submit Monthly Claim

Hours Worked

Enter total hours

Hourly Rate (R)

Enter hourly rate

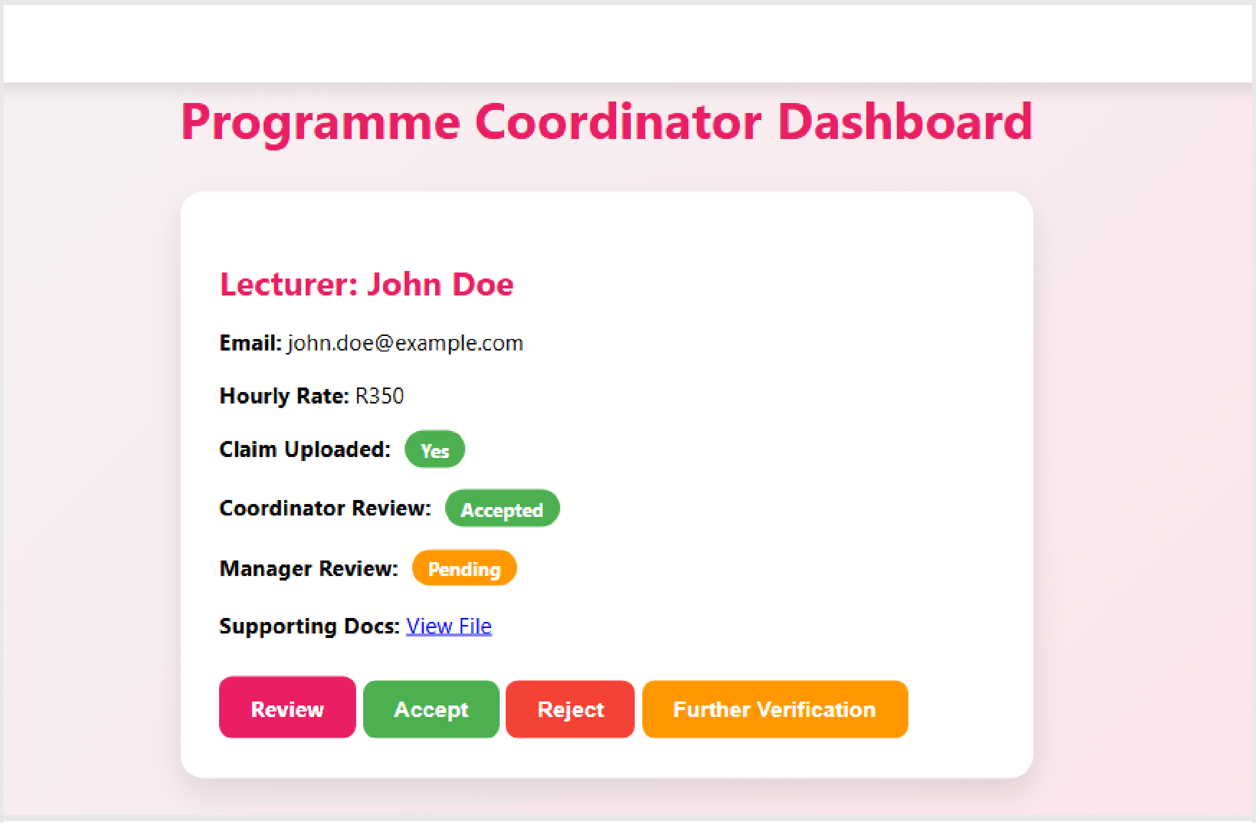
Total Amount (R)

Auto-calculated

Supporting Documents

Choose File No file chosen

Upload PDF or DOC/DOCX files only



Programme Coordinator Dashboard

Lecturer: John Doe

Email: john.doe@example.com

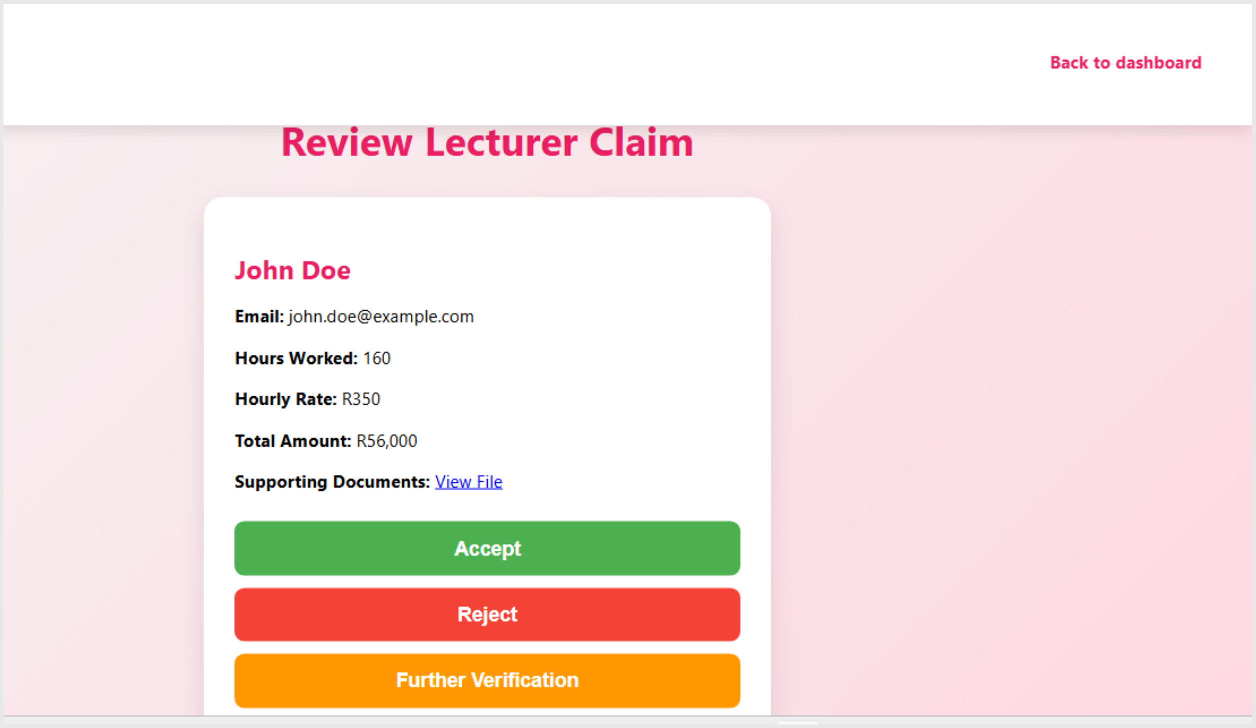
Hourly Rate: R350

Claim Uploaded: Yes

Coordinator Review: Accepted

Manager Review: Pending

Supporting Docs: View File



Review Lecturer Claim

John Doe

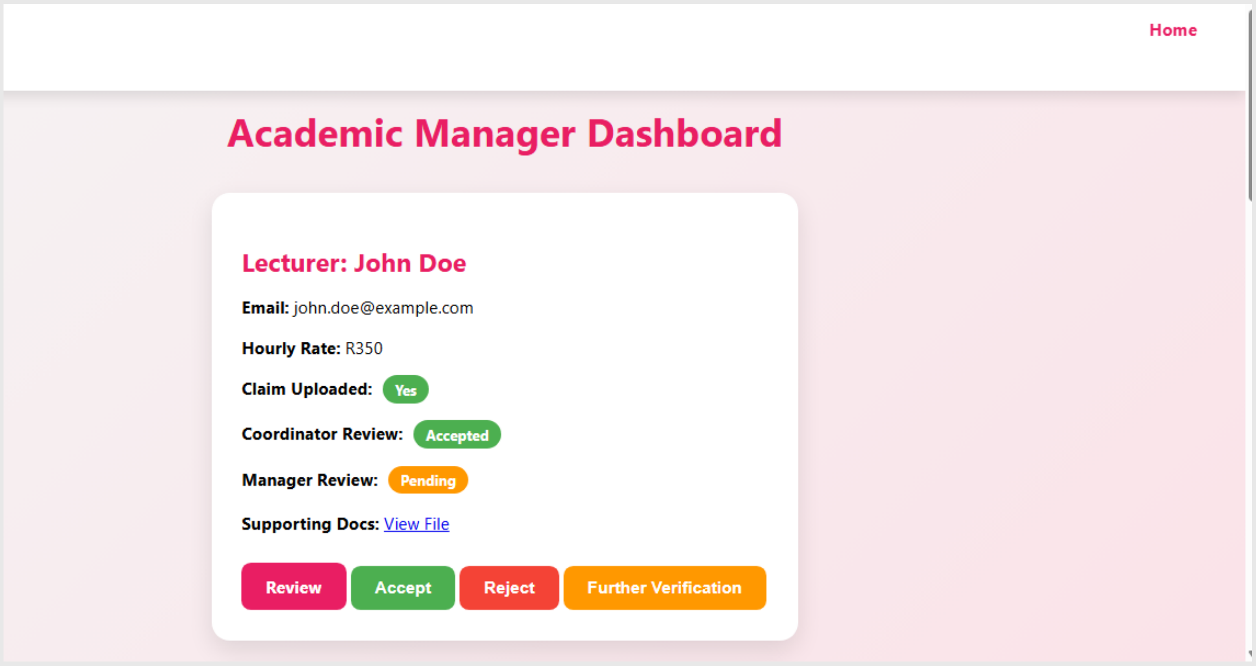
Email: john.doe@example.com

Hours Worked: 160

Hourly Rate: R350

Total Amount: R56,000

Supporting Documents: View File



Academic Manager Dashboard

Lecturer: John Doe

Email: john.doe@example.com

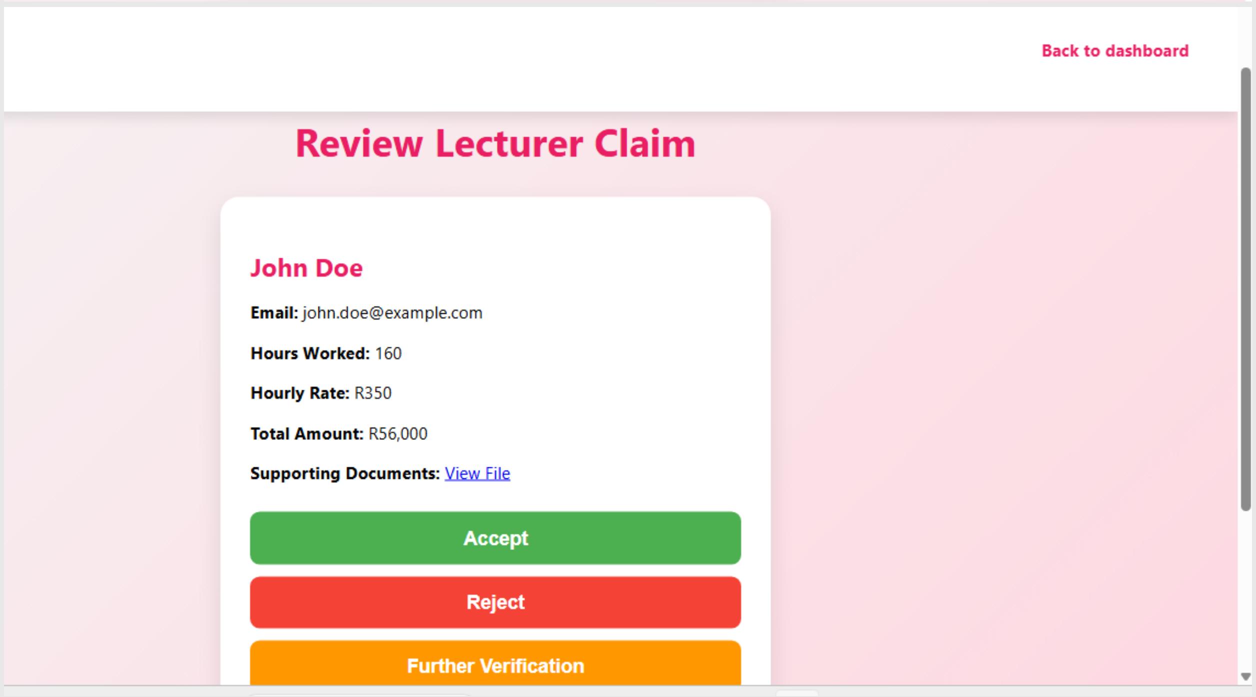
Hourly Rate: R350

Claim Uploaded: Yes

Coordinator Review: Accepted

Manager Review: Pending

Supporting Docs: View File



Review Lecturer Claim

John Doe

Email: john.doe@example.com

Hours Worked: 160

Hourly Rate: R350

Total Amount: R56,000

Supporting Documents: View File

6. Assumptions and Constraints

The development of the Contract Monthly Claim System (CMCS) is based on several key assumptions. It is assumed that each lecturer, coordinator, and academic manager has a unique login account and that lecturers will submit accurate information regarding hours worked and supporting documents. Coordinators and managers are assumed to follow the defined review and approval workflow, with coordinators reviewing claims first before they reach managers. It is also assumed that all submitted documents are in acceptable formats (PDF, DOC, DOCX) and that staff have basic computer literacy to interact with the system.

Several constraints apply to this prototype. The current system is a non-functional prototype, meaning it does not yet include live database integration, real authentication, or file storage functionality; actions such as submitting claims or reviewing documents are simulated for demonstration purposes. Additionally, the prototype's design is limited to front-end functionality and UI interactions; back-end processes like automatic total calculations, notifications, or workflow automation are not implemented. These constraints also include browser compatibility considerations, as the system is optimized for modern desktop browsers and may not display perfectly on older devices or mobile platforms. Despite these limitations, the prototype successfully demonstrates the intended workflow, visual design, and usability of the CMCS.

7. Conclusion

In Part 1 of the Contract Monthly Claim System (CMCS), the main deliverables included the development of a fully designed user interface for all key roles, including lecturers, programme coordinators, and academic managers. This encompassed the home page, login and sign-up forms, lecturer dashboards, claim submission forms, coordinator and manager dashboards, and review pages. Each screen was designed with usability, consistency, and accessibility in mind, using card-based layouts, colored status badges, and intuitive navigation to reflect real-world workflows for claim submission, verification, and approval. Although the prototype is non-functional, it establishes a clear foundation for the later development of functional features such as database integration, user authentication, file storage, and automated workflows. By demonstrating the system structure, visual design, and role-based interactions, Part 1 ensures that subsequent functional development can proceed efficiently while maintaining the intended user experience and workflow logic.

References

Clever Learning. 2025. Available at: https://youtu.be/tFxAhNALYJI?si=YWl2dbgPJ6nRRcR- [Accessed 01 Sep. 2025].

Coding Under Pressure. 2021. Available at: https://youtu.be/nEc10-pTlp0?si=YLmTKPW4Nc1LJ_gl [Accessed 01 Sep. 2025].