

# Programming Part 1

*17 September 2025  
Prog6212*

*Lebohang Selematsela  
ST10446504*

# Documentation

## Introduction

Universities and colleges rely on part-time or contract lecturers most of the time who submit monthly claims for their teaching hours and other activities they do. Traditionally this process was always paper based which required the lecturers to fill out forms, attach the required documents and submit the forms and documents to the department. After this whole process the manager had to review the claims manually, which was time-consuming, which led to errors, and it had a lack of transparency. The Contract Monthly Claim System (CMCS) has been proposed as a digital solution to streamline the claim process. The current stage of this development is to focus on the design and the presentation of a graphical user interface (GUI) which currently is only a prototype. This system will allow lecturers to create and submit claims online, allow managers to review and approve the claims efficiently and both parties are going to benefit from a transparent and paperless workflow.

## Objectives

Main objectives:

- . Provide lecturers with an easy-to-use interface for submitting monthly claims.
- . Ensures supporting documents are uploaded easily and associated with the claims.
- . Gives managers the ability to review and approve/reject claims digitally.
- . It maintains transparency by allowing lecturers to track their claim status.
- . It reduces paper usage and moves towards a fully electronic approval process.

## Assumptions and Constraints

- . This prototype is just a GUI meaning it has no back end or database functionality.
- . Only registered contact lecturers can access this claim submission feature.
- . Managers and administrative staff have distant roles with the appropriate permissions.
- . This project should be completed within 5 weeks, focusing on design deliverables.

## UML Class Diagram

UML class diagram main entities and methods:

- . **Lecturer**: Represents the contract lecturer who submits the claims.

Attributes: LecturerID, Name, Department and Email

Methods: submitClaim() and updateClaim()

- . **Claim**: Represents each submitted claim.

Attributes: ClaimID, LectureID, DateSubmitted, ClaimAmount, Status and Description

Methods: createClaim(), updateClaim() and cancelClaim()

- . **SupportingDocuments**: Represents the files uploaded to justify a claim.

Attributes: DocumentID, ClaimID, FileName, FilePath and Description

Methods: uploadDocument() and removeDocument()

- . **Manager**: Represents the user responsible for reviewing the claims.

Attributes: ManagerID, Name, Email and Role

Methods: reviewClaim() and forwardClaim()

- . **Approval**: Records the manager's decision and links a claim with a manager.

Attributes: ApprovalID(), ClaimID, ManagerID, ApprovalDate, Decision and Comments

Methods: approveClaim() and rejectClaim()

# Project Plan

This project should be completed within a period of 5 weeks.

## **Week 1:** Requirements and Initial Design

Identify the system's scope, constraints and requirements. Draft an early documentation.

## **Week 2:** UML and Project Plan

Develop a UML diagram and create a table/Gantt chart for the project scheduling.

## **Week 3:** GUI Wireframes and Prototype

Design a key screen which includes the lecture dashboard, claim submission form and a Manager Dashboard.

## **Week 4:** Review and Refine GUI + Documentation

Conduct peer review, refine layouts and update the documentation.

## **Week 5:** Report and Git Flow

Finalize the report and make sure that the diagrams are embedded and finally commit all the files with proper version control.

# Wireframes

The 3 main wireframes:

- . **Lecturer Dashboard:** This provides navigation and quick access to create claims, upload documents and view past submissions.
- . **Claim Submission Form:** This contains fields for the date, amount, description and file uploads.
- . **Manager Dashboard:** This dashboard displays pending claims requiring review, approved claims and rejected claims.

# Version Control Plan

The 5 commits:

1. Added a ReadMe file
2. Added my controllers
3. Added my models
4. Added my views
5. Decorated my project made it look professional

# Conclusion

This CMCS prototype provides a GUI model of how the contract lecturer claim system is going to function. The documentation, UML diagram, project plan, wireframes and version control strategy together shows a clear pathway for a full system development in later phases.

# Project Plan

## Methodology

This project follows an Iterative development methodology. Each week represents a cycle of design, review and refinement. Using this approach allows Incremental progress, early feedback and continuous improvement without needing a fully finished product before evaluation.

## Tasks, Dependencies and Timeline

**Week 1:** Requirements and Initial Design

Define system scope and constraints.

**Week 2:** UML and Project Plan

Create a UML diagram and table/Gantt chart.

**Week 3:** GUI Wireframes and prototype

Design and implement initial front-end screens.

**Week 4:** Review and Refine GUI + Documentation

Review deliverables and apply feedback.

**Week 5:** Report + Git Flow

Finalize documentation and ensure proper version control.

## Resources

**Tools:** Draw.io (UML class diagram) and LucidChart (Table)

**Technologies:** Visual Studio (MVC)

**Version Control:** GitHub

## Risks, Assumptions and Constraints

**Risks:** Time limitations, inconsistent feedback and potential design misalignment.

**Assumptions:** Lecturers and managers use web-based access.

**Constraints:** Project limited to 5 weeks and prototype is restricted to GUI only (no back-end functionality)



Task	Start Date	End Date	Duration	Dependencies
Requirements and Initial Design	Week 1	Week 1	7 Days	None
UML and Project Plan Finalization	Week 2	Week 2	7 Days	Requirements
GUI Wireframes and Prototype	Week 3	Week 3	7 Days	UML and Plan
Review and Refine GUI + Documentation	Week 4	Week 4	7 Days	Wireframes and Prototype
Wrap up: Report and Git Flow	Week 5	Week 5	7 Days	Review and Documentation

# System Design Choices

## Design Justification

The system was designed with simplicity and usability as its primary goals. Since the lecturers and managers are the main users the interface prioritizes clarity, input fields and step-by-step navigation. This interface separates the lecturer and manager dashboards which reduces complexity and ensures that each user only interacts with the features that are relevant to their role. The system adopts an MVC (Model-View-Controller) architecture in the prototype. This approach adopts an MVC architecture in the prototype. This approach separates concerns which make it easier to maintain the interface, improve scalability and eventually connect to a database back end.

## Database Structure

Even though this stage only focuses on GUI design, the system's UML class diagram reflects an underlying database schema.

### Tables:

- . **Lecturer:** LecturerID, Name, Department, Email
  - . **Claim:** ClaimID, LecturerID, DateSubmitted, ClaimAmount, Status, Description
  - . **SupportingDocument:** DocumentID, ClaimID, FilePath, Description
  - . **Manager:** ManagerID, Name, Role, Email
  - . **Approval:** ApprovalID, ClaimID, ManagerID, ApprovalDate, Decision, Comments
- The primary keys are used to identify records while the foreign keys are used to link entities. Using this structure ensures traceability and enforces relationships between claims, documents and approvals.

## Support for Claim Workflow

- . **Claim Submission:** A lecturer logs in, enters the claim details into the claim form provided and uploads supporting documents which the claim table stores the record and the supporting documents table links the files to the claim.
  - . **Verification:** Once the claim is submitted the claim status is set to "Pending" then the managers can view the pending claims through their dashboard which retrieves data from the claim and the lecturer tables.
  - . **Approval:** A manager records the decision in the approval table which is linked to both the claim and the manager then the claim status is updated to "Approved" or "Rejected" and the lecturer can track the status from their status.
- This design ensures that every claim can be traced from the submission all the way through to the decision (Approval or rejection) with supporting evidence stored and accessible for verification. The database structure also allows future scalability such as adding audit logs or integrating with the payroll systems.

# Database Design

## UML Class Diagram

This UML class diagram represents the data requirements of the Contact Monthly Claim System (CMCS). It models the entities and their relationships. The diagram shows how the data flows between users and the claim verification process.

- . Lecturer (1..) → (1..\*) Claim
- . Claim (1..) → (0..\*) SupportingDocument
- . Manager (1..) → (0..) Approval
- . Claim (1..1) → (0..\*) Approval

## Entities and Relationships

### Lecturer

- . Attributes: LecturerID (PK), Name, Department, Email
- . Each Lecturer can submit multiple claims.

### Claim

- . Attributes: ClaimID (PK), LecturerID (FK), DateSubmitted, ClaimAmount, Status, Description
- . Each Claim belongs to one Lecturer and may have multiple SupportingDocuments.

### SupportingDocument

- . Attributes: DocumentID (PK), ClaimID (FK), FilePath, Description
- . Each document is tied to one Claim.

### Manager

- . Attributes: ManagerID (PK), Name, Role, Email
- . Each Manager can review multiple claims.

### Approval

- . Attributes: ApprovalID (PK), ClaimID (FK), ManagerID (FK), ApprovalDate, Decision, Comments
- . Each Approval record connects a Claim with a Manager's decision.

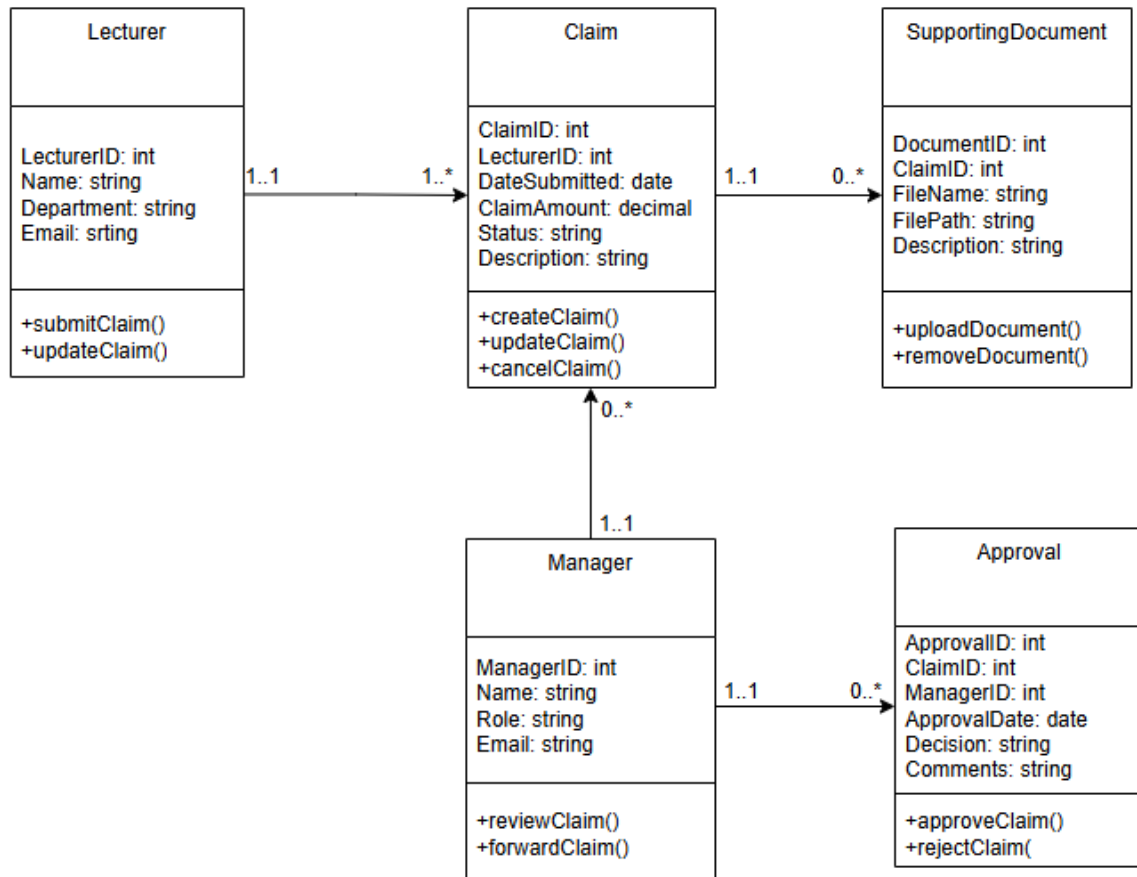
## Data Attributes and Constraints

- . **Primary Keys (PK):** Guarantees unique identification of records.
- . **Foreign Keys (FK):** Enforces referential integrity between tables.
- . **Status Field:** Constrained to values such as pending, approved, rejected to standardize claim tracking.
- . **Decision Field in Approval:** Restricted to approved or rejected values.
- . **Claim Amount:** Must be a positive number.
- . **Data Fields:** Must store valid calendar dates.

## Rationale for Data Model

- . **Data integrity** through primary and foreign keys.
- . **Flexibility** to handle one-to-many relationships.
- . **Scalability** for future integration with payroll systems or audit logs.
- . **Clarity** by organizing information into distinct and normalizing entities that reduce redundancy.

This database design makes sure that the system can support the full claim cycle.



# Graphical User Interface

## GUI Layout and Technology

This prototype for the Contract Monthly Claim System (CMCS) is implemented using an MVC. Using this approach allows clean separation design, logic and data representation. If the system were extended in the future the MVC structure would also support integration.

## Key Screens

1. **Lecturer Dashboard:** This provides an overview of submitted claims, statuses and an option to create a new claim.
2. **Claim Submission Form:** This allows lecturers to enter claim details and upload supporting documents.
3. **Document Upload Page:** This enables attaching multiple files to a claim.
4. **Manager Dashboard:** Displays pending claims requiring review with filters to show approval and rejected claims
5. **Claim Tracking Page:** This allows lecturers to view the progress and status of the submitted claims.

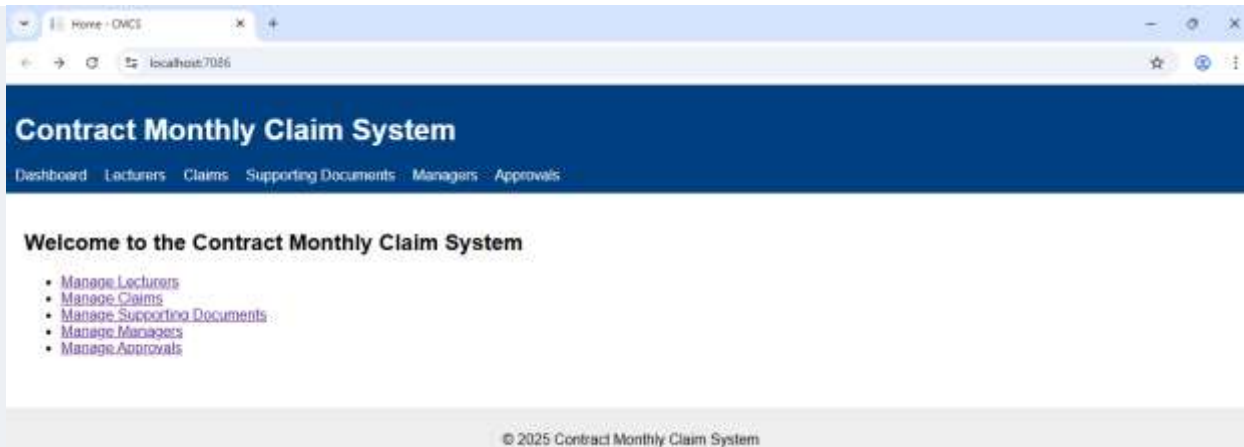
## Usability, Consistency and Accessibility Principles

- . **Usability:** Forms use clear labels, mandatory field indicators and validation feedback which is used to minimize input errors and navigation is simplified with a sidebar menu.
- . **Consistency:** Uniform layouts, consistent button styles and recurring icons make sure that the user can quickly recognize functions across different screens.
- . **Accessibility:** High-contrast colors and readable fonts were chosen to ensure legibility. Input fields and buttons follow accessibility standards for keyboard navigation and screen readers.

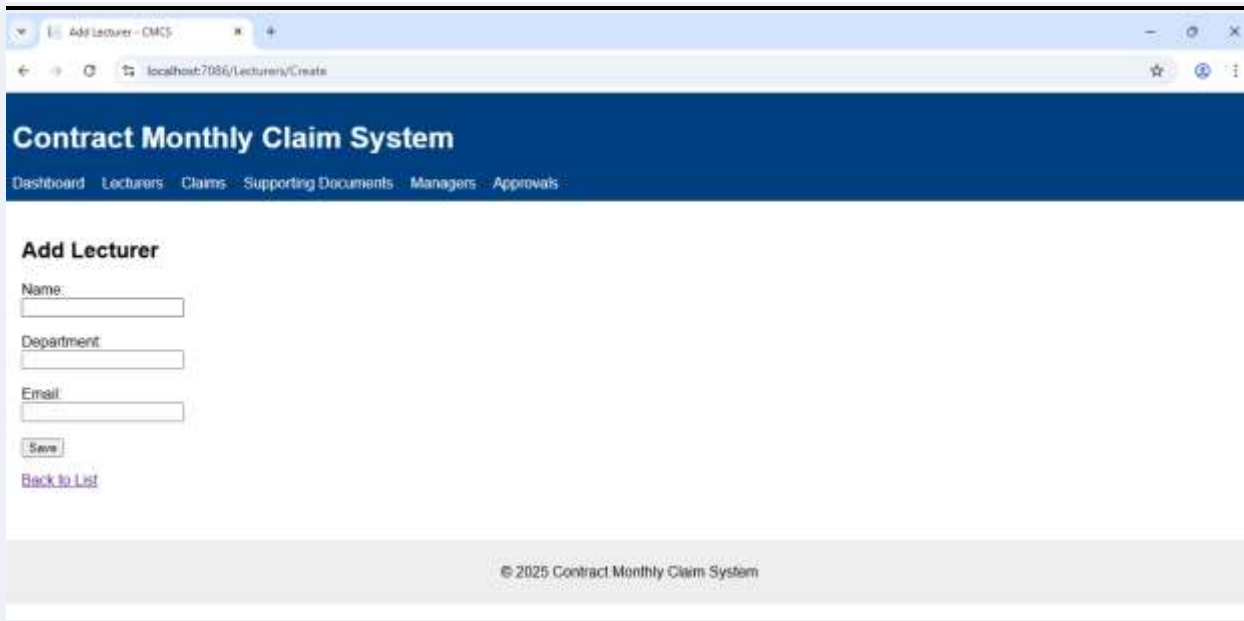
## Wireframes and Screenshots

### 1. Dashboard

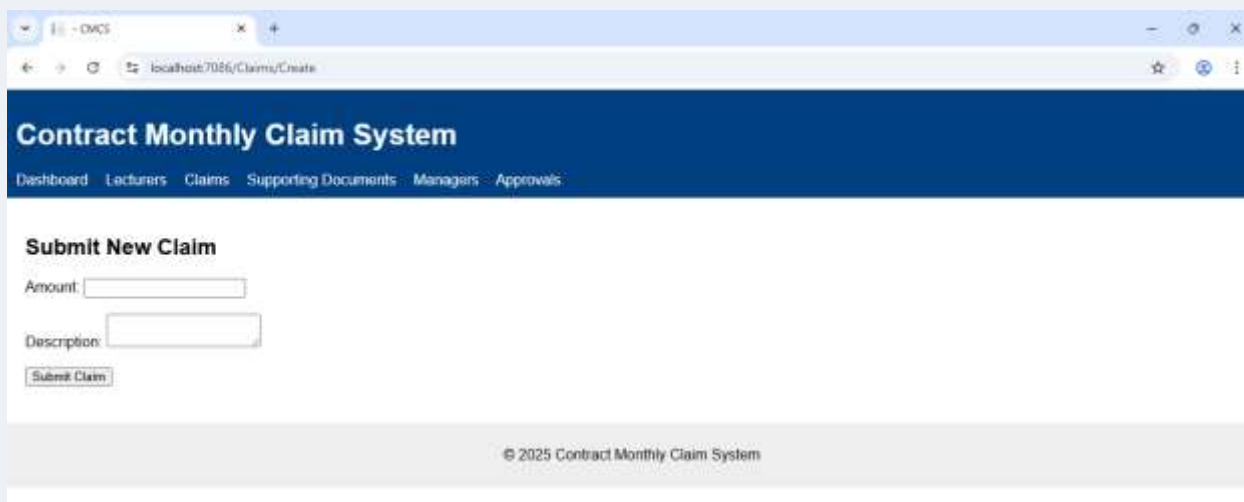
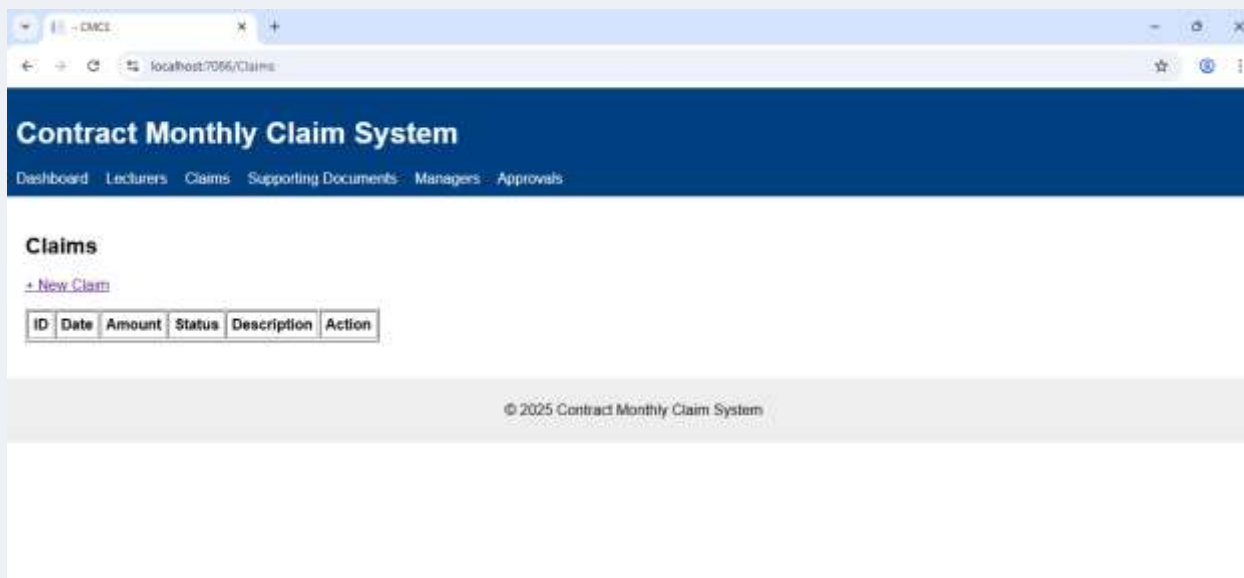




## 2. Lecturers



## 3. Claims



## 4. Managers



Contract Monthly Claim System

[Dashboard](#) [Lecturers](#) [Claims](#) [Supporting Documents](#) [Managers](#) [Approvals](#)

Add Manager

Name:

Role:

Email:

Save Manager

© 2025 Contract Monthly Claim System

## 5. Approvals

# Contract Monthly Claim System

[Dashboard](#) [Lecturers](#) [Claims](#) [Supporting Documents](#) [Managers](#) [Approvals](#)

## Approvals

[+ New Approval](#)

ID	ClaimID	ManagerID	Date	Decision	Comments
----	---------	-----------	------	----------	----------

© 2025 Contract Monthly Claim System

Contract Monthly Claim System

[Dashboard](#) [Lecturers](#) [Claims](#) [Supporting Documents](#) [Managers](#) [Approvals](#)

Record Approval

Claim ID:

Manager ID:

Decision:

Approved

Comments:

Save Approval

© 2025 Contract Monthly Claim System


# GitHub

## 1. ReadMe File

```
[main aadb339] Merge branch 'main' of https://github.com/Lebohang7843/ProgAssignment
lab_services_student@lab7L95SR MINGW64 ~/Desktop/progCMCS/progCMCS (main)
$ git commit
On branch main
nothing to commit, working tree clean

lab_services_student@lab7L95SR MINGW64 ~/Desktop/progCMCS/progCMCS (main)
$ git push -u origin main
Enumerating objects: 135, done.
Counting objects: 100% (135/135), done.
Delta compression using up to 8 threads
Compressing objects: 100% (125/125), done.
Writing objects: 100% (134/134), 1.09 MiB | 1.74 MiB/s, done.
Total 134 (delta 34), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (34/34), done.
To https://github.com/Lebohang7843/ProgAssignment.git
  91e9d4d..aadb339  main -> main
branch 'main' set up to track 'origin/main'.

lab_services_student@lab7L95SR MINGW64 ~/Desktop/progCMCS/progCMCS (main)
$
```







 main ▾





Go to file



 Code ▾

	Lebohang7843 Merge branch 'main' of <a href="https://github.com/L...">https://github.com/L...</a> aadb339 · 2 minutes ago 
 .vs	Initial commit 22 minutes ago
 progCMCS	Initial commit 22 minutes ago
 README.md	Initial commit 25 minutes ago
 progCMCS.sln	Initial commit 22 minutes ago

 README 

## 2. Controllers

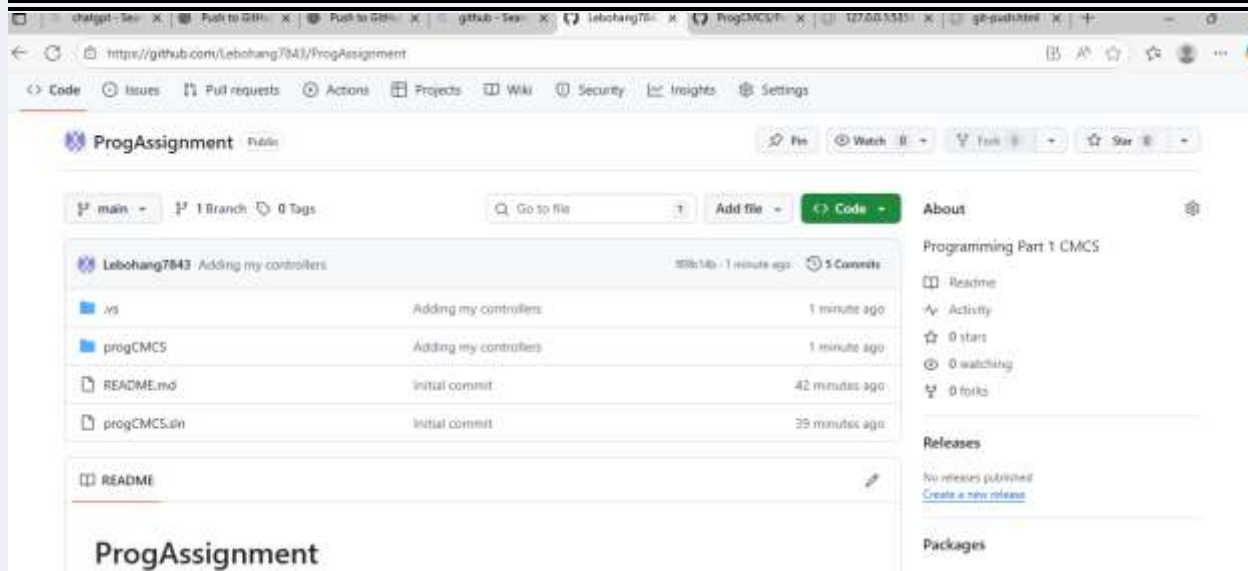
```

lab_services_student@lab7L95SR MINGW64 ~/Desktop/progCMCS/progCMCS (main)
$ git add .

lab_services_student@lab7L95SR MINGW64 ~/Desktop/progCMCS/progCMCS (main)
$ git commit -m "Adding my controllers"
[main f89b14b] Adding my controllers
33 files changed, 487 insertions(+), 26 deletions(-)
create mode 100644 .vs/VSWorkspaceState.json
create mode 100644 .vs/progCMCS/DesignTimeBuild/.dtbcache.v2
create mode 100644 .vs/progCMCS/FileContentIndex/1c05fb9f-4957-4abc-a3a6-8ee21abd42e1.vsidx
create mode 100644 .vs/progCMCS/FileContentIndex/30f6245b-f583-413e-ae66-362097e84b14.vsidx
create mode 100644 .vs/progCMCS/FileContentIndex/b6175994-6387-4d77-88ed-232ec37a3a22.vsidx
create mode 100644 .vs/progCMCS/FileContentIndex/b8c770ec-a9cf-447c-8bd1-12763646e2ac.vsidx
create mode 100644 .vs/progCMCS/FileContentIndex/bf411601-70d5-4bf0-891d-9132e75f3bfc.vsidx
create mode 100644 .vs/progCMCS/v17/.wsuo
create mode 100644 .vs/slnx.sqlite
create mode 100644 progCMCS/Controllers/ApprovalsController.cs
create mode 100644 progCMCS/Controllers/ClaimsController.cs
create mode 100644 progCMCS/Controllers/LecturersController.cs
create mode 100644 progCMCS/Controllers/ManagersController.cs
create mode 100644 progCMCS/Controllers/SupportingDocumentsController.cs
create mode 100644 progCMCS/obj/Release/net9.0/.NETCoreApp,Version=v9.0.AssemblyAttributes.cs
create mode 100644 progCMCS/obj/Release/net9.0/progCMCS.AssemblyInfo.cs
create mode 100644 progCMCS/obj/Release/net9.0/progCMCS.AssemblyInfoInputs.cache
create mode 100644 progCMCS/obj/Release/net9.0/progCMCS.GeneratedMSBuildEditorConfig.editorco
nfig
create mode 100644 progCMCS/obj/Release/net9.0/progCMCS.GlobalUsings.g.cs
create mode 100644 progCMCS/obj/Release/net9.0/progCMCS.RazorAssemblyInfo.cache
create mode 100644 progCMCS/obj/Release/net9.0/progCMCS.RazorAssemblyInfo.cs
create mode 100644 progCMCS/obj/Release/net9.0/progCMCS.assets.cache

lab_services_student@lab7L95SR MINGW64 ~/Desktop/progCMCS/progCMCS (main)
$ git push origin main
Enumerating objects: 64, done.
Counting objects: 100% (63/63), done.
Delta compression using up to 8 threads
Compressing objects: 100% (40/40), done.
Writing objects: 100% (43/43), 601.96 KiB | 3.63 MiB/s, done.
Total 43 (delta 17), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (17/17), completed with 8 local objects.
To https://github.com/Lebohang7843/ProgAssignment.git
   aadb339..f89b14b  main -> main

```



### 3. Models

```

lab_services_student@lab7L95SR MINGW64 ~/Desktop/progCMCS/progCMCS (main)
$ git add .

lab_services_student@lab7L95SR MINGW64 ~/Desktop/progCMCS/progCMCS (main)
$ git commit -m "Adding my models"
[main cd795cc] Adding my models
21 files changed, 289 insertions(+), 32 deletions(-)
delete mode 100644 .vs/progCMCS/FileContentIndex/1c05fb9f-4957-4abc-a3a6-8ee21abd42e1.vsix
delete mode 100644 .vs/progCMCS/FileContentIndex/30f6245b-f583-413e-ae66-362097e84b14.vsix
create mode 100644 .vs/progCMCS/FileContentIndex/4e079caa-c2e1-43c8-9dd6-e246eed27dd3.vsix
create mode 100644 .vs/progCMCS/FileContentIndex/6a37f70e-5015-444d-aa6c-151caaad1ec1.vsix
delete mode 100644 .vs/progCMCS/FileContentIndex/bf411601-70d5-4bf0-891d-9132e75f3bfc.vsix
create mode 100644 .vs/progCMCS/FileContentIndex/eaacadc2-a2a1-4e7d-aba0-2ecbc79de468.vsix
create mode 100644 progCMCS/Models/Approval.cs
create mode 100644 progCMCS/Models/Lecturer.cs
create mode 100644 progCMCS/Models/Manager.cs
create mode 100644 progCMCS/Models/SupportingDocument.cs
create mode 100644 progCMCS/Models/claim.cs

lab_services_student@lab7L95SR MINGW64 ~/Desktop/progCMCS/progCMCS (main)
$ git push origin main
Enumerating objects: 51, done.
Counting objects: 100% (51/51), done.
Delta compression using up to 8 threads
Compressing objects: 100% (28/28), done.
Writing objects: 100% (30/30), 14.48 KiB | 1.32 MiB/s, done.
Total 30 (delta 14), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (14/14), completed with 11 local objects.
To https://github.com/Lebohang7843/ProgAssignment.git
 f89b14b..cd795cc main -> main

```



## Programming Part 1 CMCS

☆ 0 stars    🍴 0 forks    👁 0 watching    🔗 1 Branch    🏷 0 Tags    📈 Activity

🌐 Public repository

🔗 main ▾



Go to file



<> Code ▾



**Lebohang7843** Adding my models

cd795cc · 1 minute ago



.vs

Adding my models

1 minute ago



progCMCS

Adding my models

1 minute ago



README.md

Initial commit

49 minutes ago



progCMCS.sln

Initial commit

47 minutes ago

## 4. Views

```

lab_services_student@lab7L95SR MINGW64 ~/Desktop/progCMCS/progCMCS (main)
$ git push origin main
Enumerating objects: 183, done.
Counting objects: 100% (182/182), done.
Delta compression using up to 8 threads
Compressing objects: 100% (143/143), done.
Writing objects: 100% (152/152), 2.52 MiB | 4.18 MiB/s, done.
Total 152 (delta 25), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (25/25), completed with 13 local objects.
To https://github.com/Lebohang7843/ProgAssignment.git
   cd795cc..ae63ed8  main -> main

```

ProgAssignment · Public

main · 1 Branch · 0 Tags

Go to file

Add file

Code

About

Programming Part 1 CMCS

Readme

Activity

0 stars

0 watching

0 forks

Releases

No releases published

Create a new release

Commit	Message	Time
ae63ed8	Adding my views	1 minute ago
cd795cc	Adding my views	1 minute ago
cd795cc	Adding my views	1 minute ago
cd795cc	Initial commit	1 hour ago
cd795cc	Initial commit	1 hour ago

## 5. Display

```

lab_services_student@lab7L95SR MINGW64 ~/Desktop/progCMCS/progCMCS (main)
$ git add .

lab_services_student@lab7L95SR MINGW64 ~/Desktop/progCMCS/progCMCS (main)
$ git commit -m "Changed the layout of my project"
[main 14365e0] Changed the layout of my project
 24 files changed, 127 insertions(+), 161 deletions(-)
 rename .vs/progCMCS/FileContentIndex/{526f7518-2f8a-45b5-8c39-d06b8480db24.vsidx => 2cbdb586-8123-482f-bf9a-1cfb93494095.vsidx} (99%)

lab_services_student@lab7L95SR MINGW64 ~/Desktop/progCMCS/progCMCS (main)
$ git push origin main
Enumerating objects: 59, done.
Counting objects: 100% (59/59), done.
Delta compression using up to 8 threads
Compressing objects: 100% (27/27), done.
Writing objects: 100% (32/32), 539.34 KiB | 3.41 MiB/s, done.
Total 32 (delta 22), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (22/22), completed with 19 local objects.
To https://github.com/Lebohang7843/ProgAssignment.git
   ae63ed8..14365e0  main -> main

```

ProgAssignment

main · 1 Branch · 0 Tags

Go to file

Add file

Code

About

Programming Part 1 CMCS

Readme

Activity

0 stars

0 watching

0 forks

Releases

Commit	Message	Time
14365e0	Changed the layout of my project	1 minute ago
ae63ed8	Changed the layout of my project	1 minute ago
cd795cc	Initial commit	1 hour ago
cd795cc	Initial commit	1 hour ago



# Assumptions and Constraints

## Assumptions

- . **User Roles:** There are only 2 primary roles in the system which are Lecturers (Submit claims) and Managers (Review and approve claims)
- . **Claim Process:** Claims can be submitted monthly which could include teaching-related description and supporting documents.
- . **Supporting Documents:** All claims submitted must include at least 1 valid document and documents should be in a digital format.
- . **System Access:** Users access the system through the website.
- . **Workflow:** Every claim must go through submission, verification and approval steps before being finalized.

## Constraints

- . **Prototype Scope:** This non-functional interface includes only GUI mock-ups, diagrams and documentation with no back end or database integration.
- . **Timeframe:** The project is running on a 5-week schedule which does restrict the depth of the development.
- . **Technology Choice:** This prototype is limited to a front end MVC structure with static data.

## Design Limitations

- . There is no live integration with payroll or institutional databases has been implemented.
- . Authentication and authorization are not fully modeled but instead user roles are stimulated in the wireframes.
- . Error handling, file validation and security checks assumed but not included in this stage of the prototype.
- . Scalability and performance will be discussed in the later stages.



# Conclusion

Part 1 of the Contract Monthly Claim System (CMCS) shows the foundation of a structured, paperless claim process for contact lecturers. This report has shown a clear problem statement, objectives and system documentation which is followed by a project plan which outlines the tasks, dependencies, methodology, resources and risks. A UML class diagram and database design which illustrates the underlying data requirements while the GUI prototype showcased how lecturers and managers will interact with the system through dashboards, submission forms and approval screens. Wireframes, screenshots and design justifications that the interface is developed with usability, consistency and accessibility in mind. Assumptions and constraints were documented to set realistic expectations, which ensures that the scope remained focused on a GUI-only prototype that meets the 5-week timeline. These deliverables form a robust foundation for future development. In the later phases this system can be extended and have a fully functional back end, database integration and live approval workflows. Once this stage is completed the team will be able to establish groundwork for a scalable and reliable system that can eventually support institutional claim processing at a production level.

## Reference List

SPARX SYSTEMS (n.d.). *Database Modeling with UML | Sparx Systems*.

[online] sparxsystems.com. Available at:

<https://sparxsystems.com/resources/tutorials/uml/datamodel.html>.

Malsam, W. (2025). *8 Project Plan Examples (Templates Included)*. [online]

ProjectManager. Available at: <https://www.projectmanager.com/blog/project-plan-examples>.



