

Portfolio of Evidence

PROG6212

Natheem Scott | ST10109685 | 08/08/24

Contents

[POE PART 1 2](#_Toc176851077)

[Documentation 2](#_Toc176851078)

[GUI Design Decision: 2](#_Toc176851079)

[Database Structure: 2](#_Toc176851080)

[Assumptions and Constraints: 3](#_Toc176851081)

[UML Design 4](#_Toc176851082)

[Project Plan 5](#_Toc176851083)

[GitHub Link 7](#_Toc176851084)

# POE PART 1

## Documentation

When creating a Graphical User Interface (GUI), one needs to plan and analyze the information they are given to create the best application needed for the project. I will discuss the Layout, design choices, and structure of the database that will be used in the project.

### GUI Design Decision:

* The GUI design should have an intuitive and user-friendly interface for the Contract Monthly Claim System (CMCS). Design decisions should be made with the goal of enhancing overall user experience and accelerating claim submission and clearance.
* It should be simple and logical for academic managers, program coordinators, lecturers, and other user roles to traverse the GUI. As a result, they will be able to just complete their specific jobs within the system.
* The design needs to have components that transparency, dependability, and accountability in the administrative processes. This supporting information, and providing feedback on the choices made in relation claims.
* The GUI should be designed to decrease error chances and increase user happiness in addition to increase productivity. This can mean implementing user-friendly error handling protocols, live validations, and user-friendly interfaces.

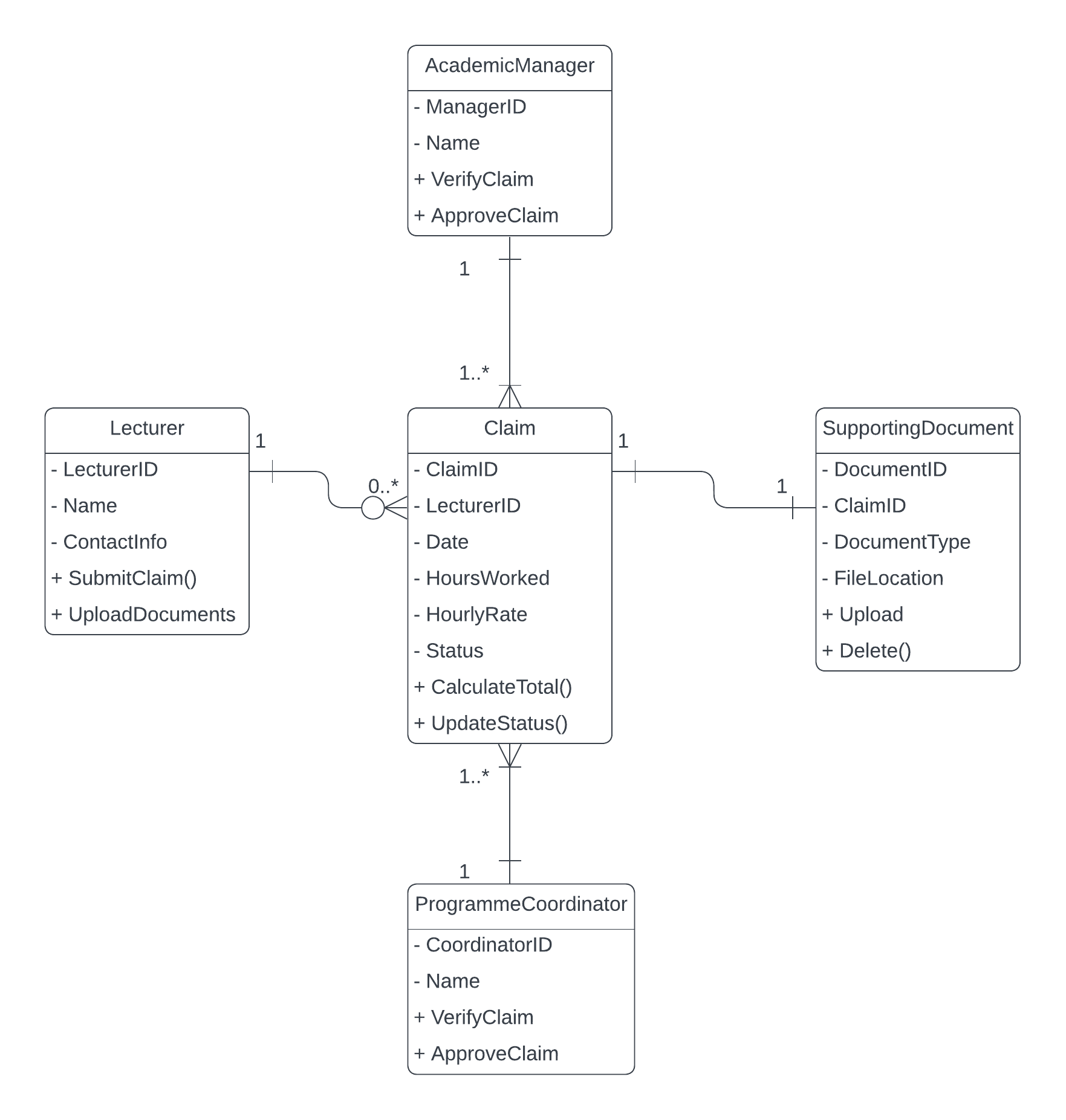
### Database Structure:

* The data requirements for the applications should be accurately reflected in the database of the CMCS.
* The database should contain entities like lecturers, claims, supporting documents, and approvals that have the right attributes to record the relevant data.
* The relationships between these entities-such as the one-to-many relationships between lecturers and their filled claims and the many-to-many relationship between supporting documents and claims-must be specified.
* The database design should offer data quality, consistency, and the ability to perform the complex calculations and validations required for accessibility, the preservation and retrieval of corroborating documents must be considered.

### Assumptions and Constraints:

* Academic managers, program coordinators, and independent contractor lecturers in a specific educational context are the targeted users of this application.
* The system should handle a variety of claim types and calculations that rely on hourly rates and corresponding hourly rates.
* The application must be integrated with a GitHub repository for version control and collaboration, as specified in the submission guidelines.
* Transparency should be guaranteed by the system’s reliable and consistent information delivery throughout the claim submission and approval process.
* For the software to accommodate any upcoming updates or revisions to the institution’s criteria, it must be adaptable and scalable.

## UML Design



This UML class diagram shows:

* The lecturer class with a one-to-many relationship to Claim.
* The Claim class with a one-to-many relationship to SupportingDocument.
* Both ProgrammeCoordinator and AcademicManager classes with one-to-many relationships to Claim.

Each class has key attributes and unique methods. The number of relationships shows how many there are, and the relationships are represented by the lines joining the classes.

This diagram, which provides a visual representation of the relationships and data structure in the Contract Monthly Claim System, may have an impact on the database architecture and system implementation.

## Project Plan

**Project Name: Crontract Montly Claim System (CMCS) Development**

**Product Owner: Client**

**Scrum Master: Natheem Scott**

**Start Date: 20th August 2024**

**Project Objective**

Develop a Contract Montly Claim System (CMCS) with a Graphical User Interface (GUI) built using .NET Core web application. The system will handle the front-end design, task dependencies, and deliverables for a smooth user experience.

**Sprints Overview**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sprint # | Start Date | End Date | Sprint Goals | Deliverables |
| Sprint 1 | 20/08/2024 | 21/08/2024 | Analyse project context and gather info | Completed context analysis (KAN-1) |
| Sprint 2 | 22/08/2024 | 24/08/2024 | Plan and document UI needs | Documented next steps and GUI information (KAN-2, KAN-3) |
| Sprint 3 | 25/08/2024 | 05/09/2024 | Design UML and GUI for CMCS | UML design, GUI front-end for CMCS (KAN-4, KAN-6) |
| Sprint 4 | TBD | TBD | Implement GUI functionalities | Functionality added to .Net Core web application (KAN-7) |

**Backlog**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature/User Story | Priority | Estimated Time | Assigned to | Status |
| Analyse the project context | High | 1 day | NS | Done |
| Plan the next steps based on gathered information | Medium | 2 days | NS | Done |
| Document GUI requirements | High | 2 days | NS | Done |
| Create a UML Design for CMCS | High | 2 hours | NS | Done |
| Design front-end for the GUI CMCS | High | 12 days | NS | In Progress |
| Add functionalities to the GUI in .NET Core web app | Medium | TBD | NS | To Do |

**Team Roles and Responsibilities**

* **Product Owner:** Client – Manage product Backlog, prioritize features.
* **Scrum Master:** NS – Facilitate meetings, remove blockers
* **Development Team:** NS – Design and implement GUI, Develop .NET Core functionalities

**Risk Analysis**

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Probability | Impact | Mitigation Plan |
| Delay in finalizing UI requirements | Medium | High | Set clear review deadlines with stakeholders |
| Integration issues with back-end APIs | Medium | Medium | Allocate buffer time for integration testing |

This strategy adheres to the Agile process and is flexible enough to be altered as new data is discovered throughout each sprint. Delivering little, useful system increments at the end of each sprint is the main goal.

# GitHub Link

<https://github.com/Natheem-ScottST10109685/Prog6212POE.git>