

Programming 2B Prog6212 POE Part 1

ST10446648 Orearabetse Riba





Contents

[1. Introduction & Context 2](#_Toc209045267)

[2. Project Planning 3](#_Toc209045268)

[3. System Design Decisions 4](#_Toc209045269)

[4 Database Design 5](#_Toc209045270)

[5. Graphical User Interface (Prototype) 6](#_Toc209045271)

[6. Assumptions and Constraints 7](#_Toc209045272)

[7. Conclusion 8](#_Toc209045273)

[References 9](#_Toc209045274)

# 1. Introduction & Context

This project is the first step in building the Contract Monthly Claim System (CMCS)—a .NET web app designed to simplify how contract lecturers at South African universities submit their monthly claims and get paid. Anyone who's dealt with manual timesheets and paper trails knows it's a slow and error-prone process (Van der Westhuizen, 2021). The CMCS aims to change that by creating a single, digital hub for lecturers to submit work and for managers to review and approve it, making everyone's lives easier and ensuring people get paid on time (UNISA, 2022).

# 2. Project Planning

We built this project step-by-step, using an Agile approach. This let our team stay adaptable and focus on getting the core structure and user flow just right before worrying about the complex backend code (Pressman and Maxim, 2020). We moved through three clear phases: Planning, Design & Building, and Documentation, which you can see detailed in the plan below.

Our main guiding principle was that this is a prototype. It was a unique challenge to build an app that looks and feels completely real—where every button clicks and every page loads—but doesn't actually save any data yet. It's like building a full-scale model of a car; everything is in place, and now we're ready to install the engine in the next phase (Sommerville, 2016).

|  |  |  |  |
| --- | --- | --- | --- |
| **Task ID** | **Description** | **Responsible** | **Deliverable(s)** |
| 1 | Part 1: Prototype Development | Student | A complete and documented prototype |
| 1.1 | Project Planning | Student | Project plan document |
| 1.2 | System Design | Student | UML class diagram and design justification |
| 1.3 | GUI/UI Prototype | Student | Non-functional GUI prototype built in .NET Core MVC |
| 1.4 | Final Report | Student | A report detailing design choices, GUI layout, and assumptions |
| 2 | Part 2: Functionality Implementation | Student | A working prototype with implemented features |
| 2.1 | Claim Submission | Student | A simple and intuitive claim submission form |
| 2.2 | Claim Approval System | Student | Separate view with 'Approve' and 'Reject' buttons for claims |
| 2.3 | Document Uploads | Student | A feature to upload and securely store supporting documents |
| 2.4 | Status Tracking | Student | A tracking system for claim status (e.g., 'Pending', 'Approved') |
| 2.5 | System Reliability | Student | Unit tests and implemented error handling mechanisms |
| 2.6 | Version Control | Student | A minimum of 5 commits to the GitHub repository with descriptive messages |

# 3. System Design Decisions

We chose C# and .NET Core MVC because they are powerful, modern tools that are perfect for building structured and scalable web applications (Microsoft, 2023). The MVC pattern helps keep the code organized by separating the data (Models), the screens the user sees (Views), and the logic that makes it all work (Controllers).

The heart of the system is its database design, which is built around five main tables:

* **User**: Everyone who logs in (Lecturers, Coordinators, Managers).
* **Role**: What permissions a user has (e.g., a Lecturer can submit claims, a Manager can approve them).
* **Claim**: The core details of a claim (hours, rate, total amount).
* **SupportingDocument**: Any files (like PDF timesheets) attached to a claim.
* **Approval**: The record of who approved or rejected a claim and when.

This structure ensures all data is connected and organized, preventing duplication and making the system efficient (Connolly and Begg, 2015).

# 4 Database DesignPlantUML diagram

# 5. Graphical User Interface (Prototype)

The prototype includes a full set of screens for managing users, roles, claims, and documents. The design is clean and professional, using Bootstrap to ensure it's easy to use and looks good on any device (W3Schools, 2025).

How it works (for now):

* You can click on every tab and button.
* The "Create" and "Edit" forms will open, but the fields are disabled—you can't type in them. This visually shows where data would be entered without actually doing it.
* Buttons like "Delete" or "Submit" will show a pop-up message explaining what they would do in a real application, but no data is changed.

It’s essentially a digital walkthrough of the final application. You can see the entire journey a lecturer would take to submit a claim and the path a manager would take to approve it. All the code for this interactive prototype has been committed to GitHub (Riba, 2025).

<https://github.com/RibaOrearabetse/PROG6212-POE-Part-1.git>

# 6. Assumptions and Constraints

We made a few assumptions to build this prototype:

* We assumed three main user types: Lecturers, Coordinators, and Managers.
* We assumed the claim process would be straightforward: submit, review, and approve/reject.

The biggest constraint was that this had to be a non-functional prototype. It’s all about the visual design and user flow. The real magic—like saving to a database, user login, and actual calculations—will be built in the next part of the project (Sommerville, 2016).

# 7. Conclusion

In this first part, we've successfully laid the entire groundwork for the CMCS. We've planned the project, designed a solid database structure, and built a fully navigable prototype that shows exactly how the final application will look and feel.

This click-through model proves our design works from a user experience perspective and sets us up perfectly for Part 2, where we will bring it to life with real functionality (Pressman and Maxim, 2020).

# Reference list

Connolly, T. and Begg, C. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management*. 6th ed. Harlow: Pearson.

Microsoft. (2023). *.NET documentation*. Available at: <https://learn.microsoft.com/dotnet/> [Accessed 17 Sept. 2025].

Pressman, R.S. and Maxim, B.R. (2020). *Software Engineering: A Practitioner’s Approach*. 9th ed. New York: McGraw-Hill.

Riba, O. (2025). *PROG6212-POE-Part-1 [GitHub Repository]*. Available at: <https://github.com/RibaOrearabetse/PROG6212-POE-Part-1.git> [Accessed 17 Sept. 2025].

Sommerville, I. (2016). *Software Engineering*. 10th ed. Harlow: Pearson.

UNISA. (2022). *Independent Contractors Claim Policy*. Pretoria: University of South Africa.

Van der Westhuizen, D. (2021). ‘Digital transformation in South African higher education’, *South African Journal of Higher Education*, 35(4), pp. 120–135.

W3Schools. (2025). *Bootstrap Tutorial*. Available at: <https://www.w3schools.com/bootstrap/> [Accessed 17 Sept. 2025].