**EZEKIEL ISAAC NARAYANASAMI**

**ST10359128**

**PROG6212**

**POE**

**PART1**

1. **DOCUMENTATION**

**Introduction**

When designing the Contract Monthly Claim System, my goal was to build something that feels simple, professional, and easy to use for both lecturers and managers. The system is meant to replace manual claim submissions with a digital solution, so it had to look trustworthy, clear, and user-friendly. I also made sure that my design decisions matched the requirements from the background and the marking rubric (Microsoft, 2023).

**Database Structure**

The database was designed using a relational model because this structure keeps data consistent and avoids duplication (Date, 2019). The main entities are Lecturer, Claim, Document, and Approval. A lecturer can submit many claims, but each claim belongs to one lecturer. A claim can have multiple supporting documents, such as receipts or timesheets. Every claim must also have one approval, which shows whether it was verified by the Programme Coordinator or the Academic Manager. This design makes it easy to track claims and ensures accountability.

**GUI Layout and Navigation**

For the layout, I chose a role-based interface. This means that when a lecturer logs in, they only see what they need which is: submitting new claims, uploading documents, and checking the status of old claims. Coordinators and managers, have their own dashboard where they can view pending claims, approve or reject them, and add remarks.

The navigation is built around a top navigation bar that always shows the app name, user role, and logout button. On the left side, there is a sidebar menu that changes depending on the role. For example, lecturers see “Dashboard,” “Submit Claim,” and “My Claims,” while managers see “Verify Claims” and “Reports.” The middle of the page is the content area where forms and tables are displayed.

**Colours, Themes, and Style**

I went with a modern and minimal theme to give the system a clean look. The primary colour is a deep blue which makes the app feel professional and reliable. I used a teal accent colour for buttons and links so that interactive elements stand out. The background is a mix of white for content panels and dark grey for headers and the sidebar. Success messages are green, warnings are orange, and errors are red. These colours give clear visual feedback to users.

For fonts, I used a simple style because it is easy to read on screens. The design uses lots of white space to make forms and tables less crowded. This helps users focus on the important actions without being confused(Nielsen Norman Group, 2020).

**Conclusion**

I made these choices because I wanted the system to be easy to learn and consistent. A lecturer should know almost instantly where to click to submit a claim, and a manager should quickly see pending items. The colours and layout are not just for looks as they provide feedback and guide users through the process. By combining a relational database with a role-based, modern interface, the CMCS can reduce errors, save time, and keep everyone on track (Nielsen Norman Group, 2020).

1. **UML CLASS DIAGRAM FOR DATABASES**

|  |
| --- |
| **LECTURER** |
| + LecturerID : Int  + Name : String  + Email : String  + HourlyRate : Decimal |

|  |
| --- |
| **CLAIM** |
| + ClaimID : Int  + LecturerID : Int  + HoursWorked : Int  + ClaimAmount : Decimal  + Status : String |

**1.. \***

**1.. 1**

**1.. \***

|  |
| --- |
| **DOCUMENT** |
| + DocumentID : Int  + ClaimID : Int  + FileName : String  + FilePath : String |

|  |
| --- |
| **APPROVAL** |
| + ApprovalID : Int  + ClaimID : Int  + ApprovedBy : String  + ApprovalDate : DateTime  + Remarks : String |

1. **PROJECT PLAN**

|  |  |  |  |
| --- | --- | --- | --- |
| **TASK** | **DEPENDANCY** | **DURATION** | **TIMELINE** |
| Requirements Analysis | - | 3 Days | Week 1 |
| Database Design | Requirements | 4 Days | Week 2 |
| UML Diagram | Database | 3 Days | Week 2 |
| GUI Prototype Design | Requirements | 5 Days | Week 3 |
| MVC Skeleton Setup | Database, UML | 7 Days | Week 4 |
| Documentation Writing | Ongoing | 5 Days | Week 4 – 5 |

1. **GUI / UI**

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

**REFERENCES:**

Microsoft. (2024) ASP.NET Core MVC Overview. Available at: <https://learn.microsoft.com/en-us/aspnet/core/mvc/overview>

(Accessed: 14 September 2025).

W3Schools. (2025) Bootstrap 5 Tutorial. Available at: <https://www.w3schools.com/bootstrap5/>

(Accessed: 14 September 2025).

Fowler, M. (2004) UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd edn. Addison-Wesley.

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

Microsoft. (2024) Razor Pages and Views in ASP.NET Core. Available at: <https://learn.microsoft.com/en-us/aspnet/core/mvc/views/razor>

(Accessed: 14 September 2025).

Sommerville, I. (2015) Software Engineering. 10th edn. Pearson Education.