Prog6212

Dr.Mpho Gololo

Assignment 1

ST10259834

Contents

[Question 1 3](#_Toc209026607)

[Question 2 5](#_Toc209026608)

[Question 3 6](#_Toc209026609)

[Question 4 9](#_Toc209026610)

[References 12](#_Toc209026611)

# Question 1

The Contract Monthly Claim System (CMCS) has been designed to simplify and improve how Independent Contractor (IC) lecturers submit their monthly claims, and how Programme Coordinators and Academic Managers verify and approve them. The system aims to remove unnecessary manual processes, reduce delays, and give all users a clear, transparent view of claim progress. From the start, the design focused on usability, clarity, and scalability, with a structure that can support more advanced features in later stages.

**Design Choices**

The system follows a layered design to separate the interface, business logic, and data management. This will make the application easier to maintain, upgrade, and test as it grows. For the front end, I chose either Windows Presentation Foundation (WPF) or ASP.NET Core MVC because both offer strong support for building rich, responsive interfaces in a .NET environment. The GUI will use a role-based design so that each user sees only the features relevant to them:

* Lecturers will access claim forms, upload supporting documents, and view claim statuses.
* Programme Coordinators and Academic Managers will see dashboards listing pending claims for review, approval, or rejection, with comment options.

This approach supports clear navigation, reduces user error, and ensures sensitive actions are only available to the correct roles.

**Database Structure**

The database will include five key entities: Lecturer, Claim, Document, Approval, and UserRole. A Lecturer can create many Claims, and each Claim can link to multiple Documents as proof. The Approval entity tracks the verification and approval process, including who approved it and when. The UserRole entity will manage user permissions. This structure ensures data integrity, supports clear tracking of claims, and will allow for detailed reporting and auditing later.

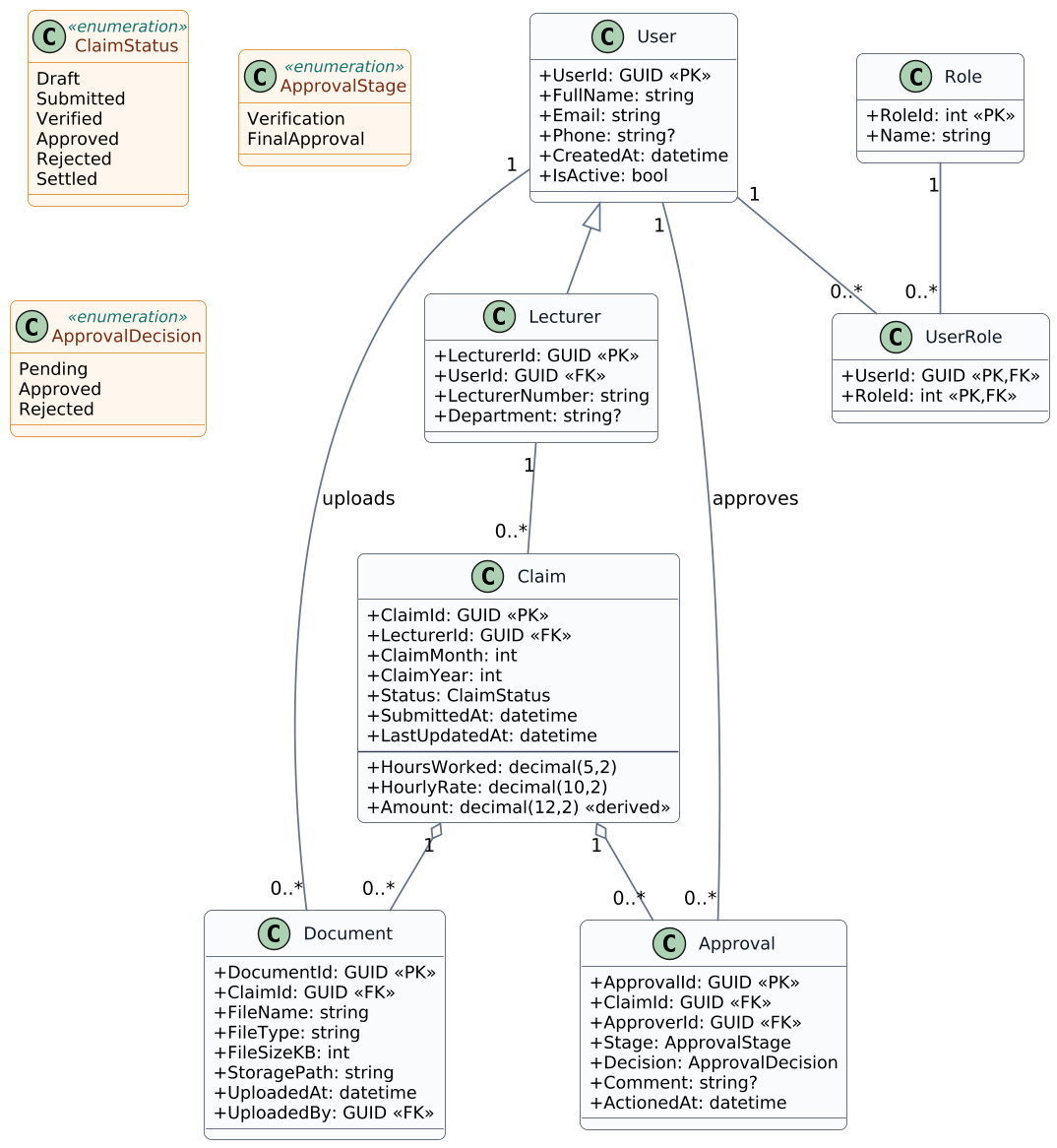
**GUI Layout**

The GUI will use a clean, dashboard-style layout. Lecturers will have a submission page with simple forms, file uploads, and a table showing all their claims and statuses. Coordinators and managers will have a verification page listing all claims, with buttons to approve or reject and a status tracker. A consistent top navigation bar, colour-coded statuses, and clear labelling will make the interface intuitive and quick to learn.

**Assumptions and Constraints**

It is assumed that users have basic computer literacy and stable internet access. Because this is only a prototype, no login or security features will be included yet — these will be implemented later. The main constraint is that the system at this stage will be non-functional and only show the design and layout. This ensures the design can be reviewed and refined before development begins.

# Question 2

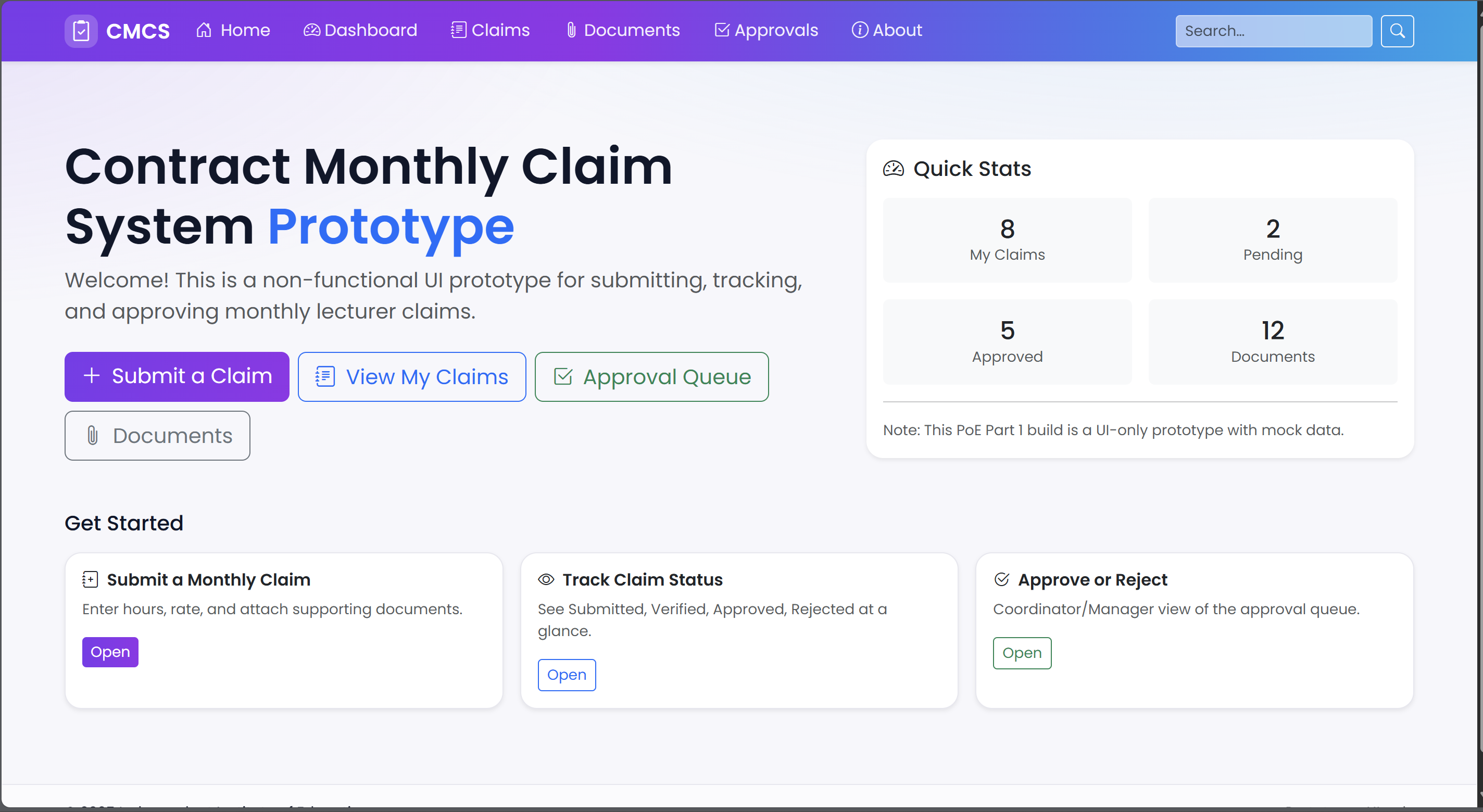


# Question 3

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Tasks & Deliverables** | **Prerequisites** | **Estimated Timeline** |
| Part 1 – Prototype UI | Requirement Gathering • Identify user roles and outline core features (claims, approvals, document uploads). | None | Week 1 |
| Part 1 – Prototype UI | Conceptual Data & UML Modelling • Create a relational schema and UML class diagram for core entities. | Requirement gathering complete | Week 1 |
| Part 1 – Prototype UI | User Interface Wireframing • Design page layouts and navigation flows using Bootstrap concepts. | UML diagram completed | Week 2 |
| Part 1 – Prototype UI | Build Static Front-End Screens • Develop non-functional MVC pages (Home, Dashboard, Claims, Approvals, Documents). | Wireframes finalised | Week 2 |
| Part 1 – Prototype UI | Compile Design Documentation • Write a brief report on design rationale, structure, and assumptions. | UI prototype built | Week 3 |
| Part 2 – Functional Web App | Create Claim Submission Module • Implement a claim form (hours, rate, notes) and save entries to the database. | Database schema ready | Week 4 |
| Part 2 – Functional Web App | Develop Approval Workflow • Build coordinator/manager screens to review, approve, or reject claims. | Claim submission working | Week 4–5 |
| Part 2 – Functional Web App | Enable Document Uploads • Allow file uploads (PDF/DOCX/XLSX) linked to claims. | Claim submission working | Week 5 |
| Part 2 – Functional Web App | Add Claim Status Tracking • Show claim statuses (Pending, Approved, Rejected) and update them dynamically. | Approval workflow complete | Week 6 |
| Part 2 – Functional Web App | Implement Testing and Error Handling • Write unit tests and add user-friendly error handling. | All main features done | Week 6 |
| Part 3 – Automation & POE | Automate Lecturer View • Auto-calculate totals (hours × rate) and validate fields with client-side scripts. | Part 2 finished | Week 7 |
| Part 3 – Automation & POE | Automate Coordinator/Manager Actions • Add rule-based validation and automated status changes for approvals. | Lecturer automation complete | Week 7 |
| Part 3 – Automation & POE | Automate HR Functions • Generate reports/invoices for approved claims and manage lecturer data. | Approval system complete | Week 8 |
| Part 3 – Automation & POE | Prepare Presentation • Create a PowerPoint showing system architecture, features, and workflows. | System feature-complete | Week 9 |
| Part 3 – Automation & POE | Version Control & Submission • Ensure 10+ Git commits and submit the full repository and documentation. | All development complete | Week 9 |

# Question 4

Screenshots of 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.

A screenshot of a computer

AI-generated content may be incorrect.

# References

Visual Paradigm. (n.d.) *UML Class Diagram Tutorial*. Available at: [https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/](https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/?utm_source=chatgpt.com) (Accessed: 17 September 2025). [visual-paradigm.com](https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/?utm_source=chatgpt.com)

Agile Modeling. (n.d.) *UML Class Diagrams: Diagramming Guidelines*. Available at: [https://agilemodeling.com/style/classdiagram.htm](https://agilemodeling.com/style/classdiagram.htm?utm_source=chatgpt.com) (Accessed: 17 September 2025). [agilemodeling.com](https://agilemodeling.com/style/classdiagram.htm?utm_source=chatgpt.com)

Microsoft Learn. (2025) *Configure an ASP.NET Core web app for authentication and authorization*\*. Available at: [https://learn.microsoft.com/en-us/entra/identity-platform/tutorial-web-app-dotnet-sign-in-users](https://learn.microsoft.com/en-us/entra/identity-platform/tutorial-web-app-dotnet-sign-in-users?utm_source=chatgpt.com) (Accessed: 17 September 2025). [Microsoft Learn](https://learn.microsoft.com/en-us/entra/identity-platform/tutorial-web-app-dotnet-sign-in-users?utm_source=chatgpt.com)

GetBootstrap. (n.d.) *Theming Bootstrap*. Available at: [https://getbootstrap.com/docs/4.0/getting-started/theming/](https://getbootstrap.com/docs/4.0/getting-started/theming/?utm_source=chatgpt.com) (Accessed: 17 September 2025). [getbootstrap.com](https://getbootstrap.com/docs/4.0/getting-started/theming/?utm_source=chatgpt.com)

PureCode.ai. (2024) *Mastering Bootstrap Theming: Create a Unique Aesthetic*. Available at: [https://blogs.purecode.ai/blogs/bootstrap-theming](https://blogs.purecode.ai/blogs/bootstrap-theming?utm_source=chatgpt.com) (Published: 17 January 2024) (Accessed: 17 September 2025). [blogs.purecode.ai](https://blogs.purecode.ai/blogs/bootstrap-theming?utm_source=chatgpt.com)