Student Name and Surname: Kiara Israel

Student number: ST10277747

Module Code: PROG6212

Assessment Type: POE PART 1

Due date: 18th September 2025

Report on My CMCS GUI Interface:

CMCS Design Document

In this Essay I will explain my Claims Management and Control System (CMCS). I have compiled my own CMCS which consists of Claims, and I have included a very intricate and well thought of design and development. The main goal of the system is to make it easier for lecturers to submit claims, for program and academic managers to examine them, and for administrators to keep track of the supporting evidence. With an emphasis on three main areas the database design, the graphical user interface which is represented as GUI layout, and the application of Bootstrap and responsive design principles this article describes the design choices taken during development. Additionally, it draws attention to the boldness and constraints that influenced execution (Anglia Ruskin University, 2025).

The Usability of my GUI is as follows, Using NoSQL and no migrations I have designed the data base just to illustrate the index and design of my web app. The database design adheres to a relational model. To meet the needs of the system, 2 entities were introduced. LecturerID is stored in the Claims model and as well as the name of said Lecturer. The SubmitClaim entity, records HoursWorked, HourlyRate, TotalAmount, SubmissionDate, and LecturerId , LecturerName , Additional Notes, Supporting Document and Claim Amount as well as a status bar which show whether the claim is pending.This works as the lecturer selects Submit New Claim and it will take the lecturer to the LecturerID where it can be typed out and the Name of the lecturer.The the lecturer will type in the HoursWorked and HourlyRate which will be multiplied together and which will show the TotalAmount. This is automatically generated when the lecturer types in the numbers and the amount can be changed as many times as possible the lecturer changes (Anglia Ruskin University, 2025).

The SupportingDocument object controls uploaded files linked to claims if the user or lecturer does not want to upload a file, he / she can simply click the button “No Thanks” if the do want to upload a file they will click upload and it will take them to their file explorer where they can upload a file and then submit claim and the claim will be submitted without a document. AcademicManager and ProgramManager are represented in a controller for logic and that can show the fully working system and a view of index has been created that represent academic and program managers and the tasks they entail. The tasks include the buttons for accept, reject and Approve Manager and the same for the coordinator as this will update the Status of the lecturer’s claim. (Anglia Ruskin University, 2025).

I have restricted the use of foreign key, and normalization was used up to Third Normal Form (3NF) to prevent redundancy. For instance, each assertion needs to cite a reliable instructor. Additionally, input validation is enforced at the database level via model-level annotations like [Required], [EmailAddress], and [StringLength]. During database design, I have thought about various outcomes, such as the idea that each claim is the property of a single lecturer, and that a claim should have more than one supporting document, but I stuck to just one, and that managers only accept or reject claims without changing rates or hours (Anglia Ruskin University, 2025).

I build this GUI/database using Visual Studio in ASP.NET Core MVC (Model View Controller) With Razor views. Bootstrap 5 was used for styling, the design and the colour scheme of my GUI with only a small amount of Cascading Styling Sheets (CSS) added. The design prioritized accessibility is that it's easier for lecturers to verify claims, and approve/reject them as well as submit claims and clarity as it makes sense and is straight forward with no intricate design adaptations added. The system was kept uniformly organized, with connections to the Submit Claim, Academic Manager, Program Manager, Supporting Documents, and Lecturer Management sections accessible on 2 pages from a navigation bar on the layout page. Version information and a contact link are included in a footer, and all content is aligned and spaced using Bootstrap's container class (Anglia Ruskin University, 2025).

The user experience is defined by 2 important interfaces. Lecturers can submit their hourly rates and hours worked in form fields styled with.form-control on the Submit Claim page. While supporting papers can be supplied by a file input control, the total sum is automatically calculated by a primary button. Users are informed of issues with validation alerts, which are styled with Bootstrap's.alert-danger. Approve and reject buttons are styled with.btn-success and.btn-danger, respectively, and all submitted claims are shown in a striped table format on the Academic Manager site. This functionality is mirrored on the Program Manager page, although it is limited to claims unique to a program. Lastly, I added in a Bootstrap modal for adding new entries and forms allow administrators to maintain lecturer information on the Lecturer site (Anglia Ruskin University, 2025).

Bootstrap was selected as the front-end framework because of its responsive grid structure, professional-looking default components, and mobile-first design philosophy. Tables and forms are neatly aligned across a range of screen sizes thanks to the 12-column grid structure, and on smaller devices, the content stacks vertically. Form fields were formatted consistently using.form-group and.form-control, and.invalid-feedback was used to make validation signals display in red. For better aesthetics, contemporary design elements like form-floating labels were also included. While buttons are consistent across the application—primary for submissions, success for approvals, and danger for rejections—tables present claims with hover highlighting for convenience of inspection. Icons from

Bootstrap Icons, such as crosses for rejection and checkmarks for acceptance, were incorporated to increase clarity (Anglia Ruskin University, 2025).

Two important factors in the GUI design were responsiveness and accessibility. Through the usage of, the layout changes to fit desktops, tablets, and mobile devices. On smaller screens, the navigation will collapse into a hamburger menu thanks to container-fluid and.navbar-expand-lg. Screen reader compatibility is further enhanced by built-in ARIA roles and attributes, guaranteeing that a broad spectrum of users can access the system (Anglia Ruskin University, 2025).

The implementation process was impacted by several constraints. It was assumed that lecturers would be the only ones responsible for uploading supporting papers, that managers would log in with the appropriate role to see their dashboards, and that hourly rates would be institutionally determined but manually submitted by lecturers. The absence of NoSQL or third-party APIs and the simplified role-based authentication without integrating enterprise identity providers and not using SQL as the database was my constraint (Anglia Ruskin University, 2025).

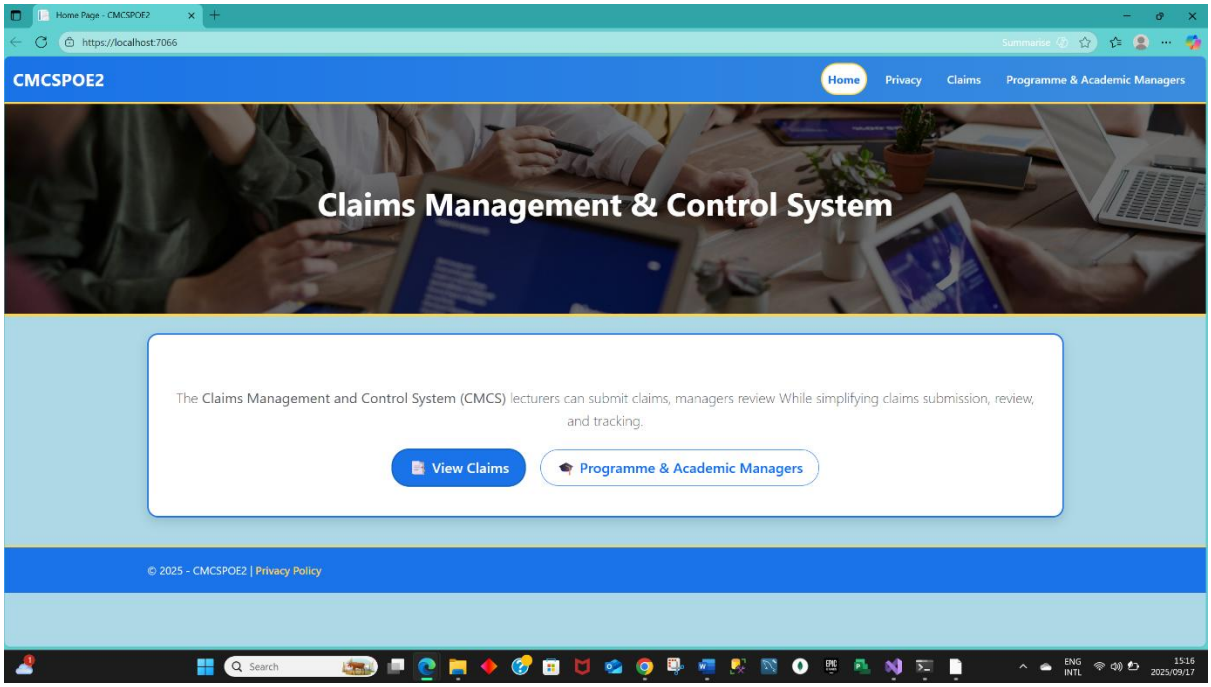
I chose MVC Architecture because it was easier and there are many reasons to support the design decisions taken during development. Models handle data, views handle presentation, and controllers handle logic, all of which improve maintainability. To offer an object-relational mapping (ORM) layer that reduced the use of SQL, because of its responsive design, accessibility capabilities, and ease of integration, Bootstrap 5 was chosen to guarantee that the system functions flawlessly. Lastly, as total amounts are constantly determined from hours and rates, automated claim computations were used to minimize human mistake and ensure financial accuracy (Anglia Ruskin University, 2025).

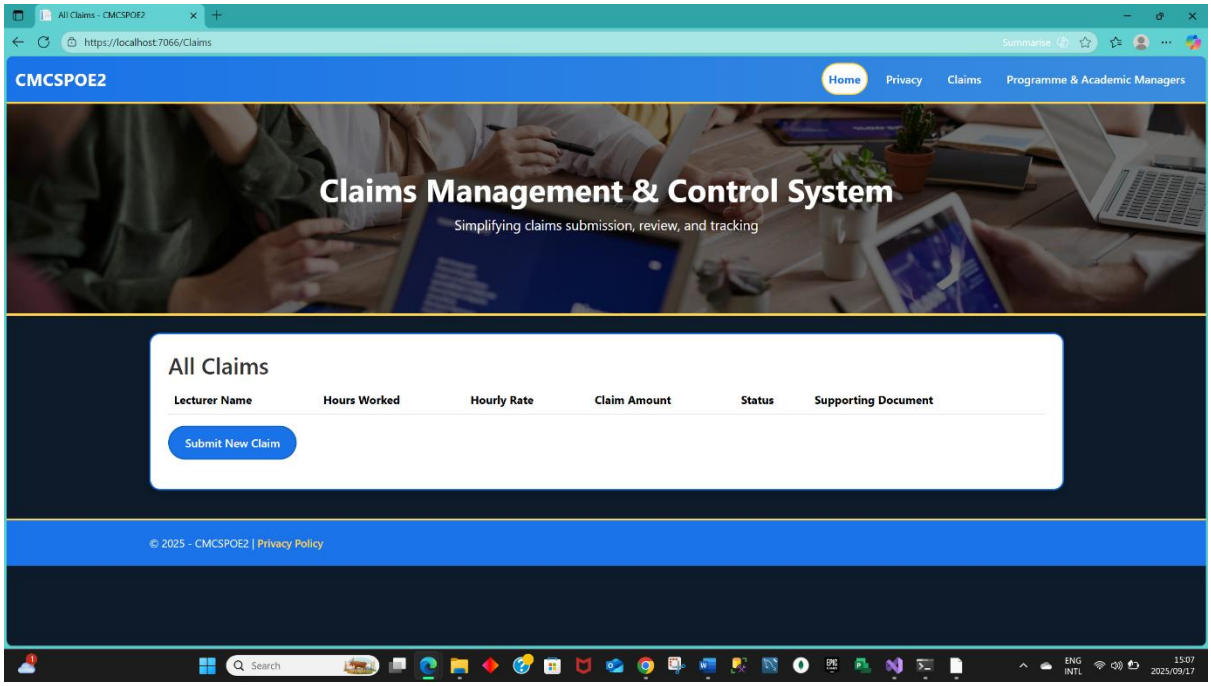
In conclusion for my MVC the system creates a professional and maintainable solution by balancing usability considerations, technical limitations, and functional needs (Anglia Ruskin University, 2025).

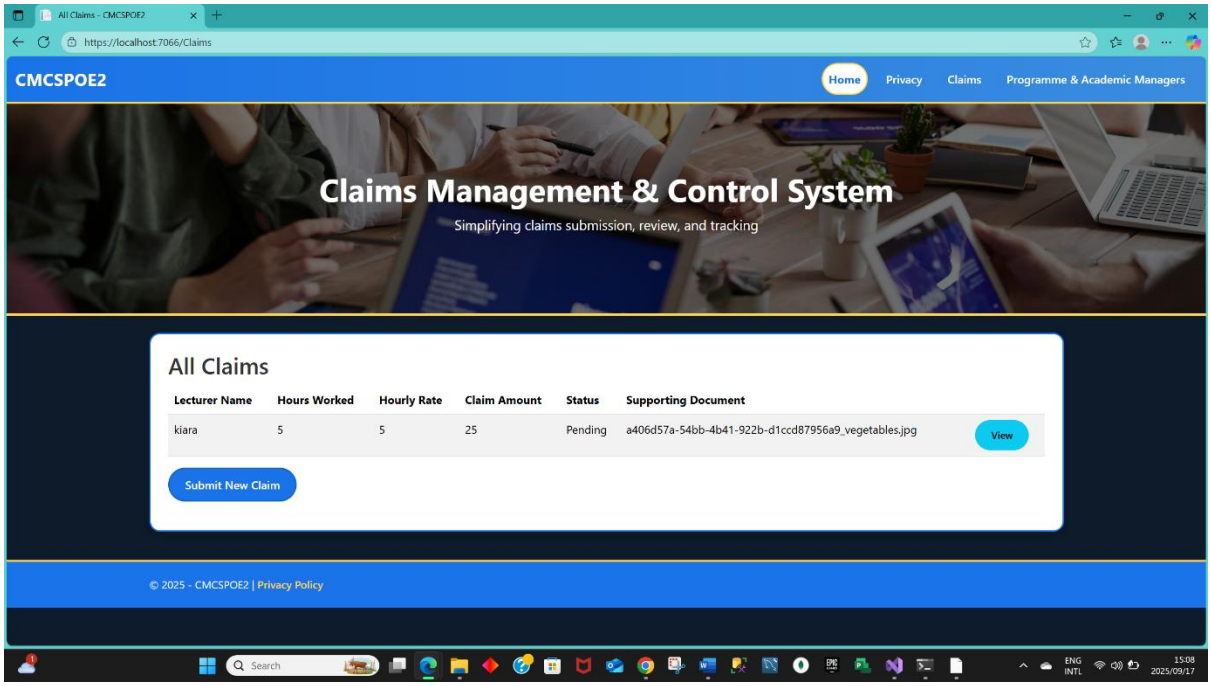
Total Words: 911

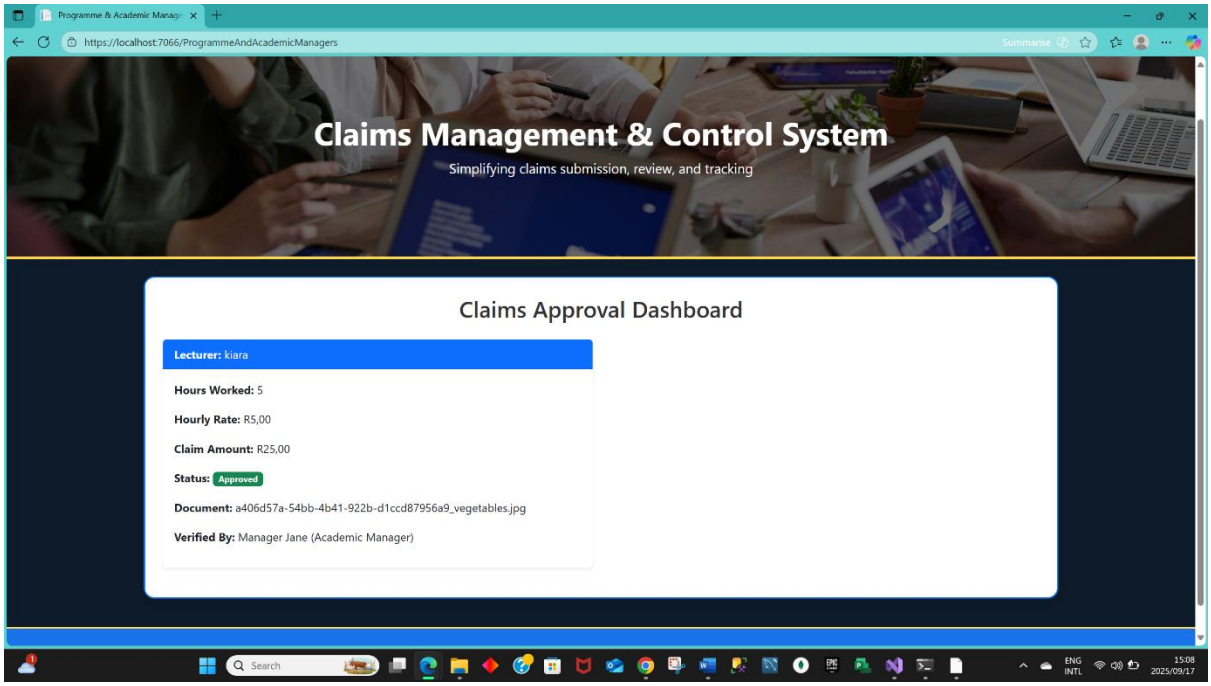
Screenshots of GUI Interface: From Opening Page till the Dashboard

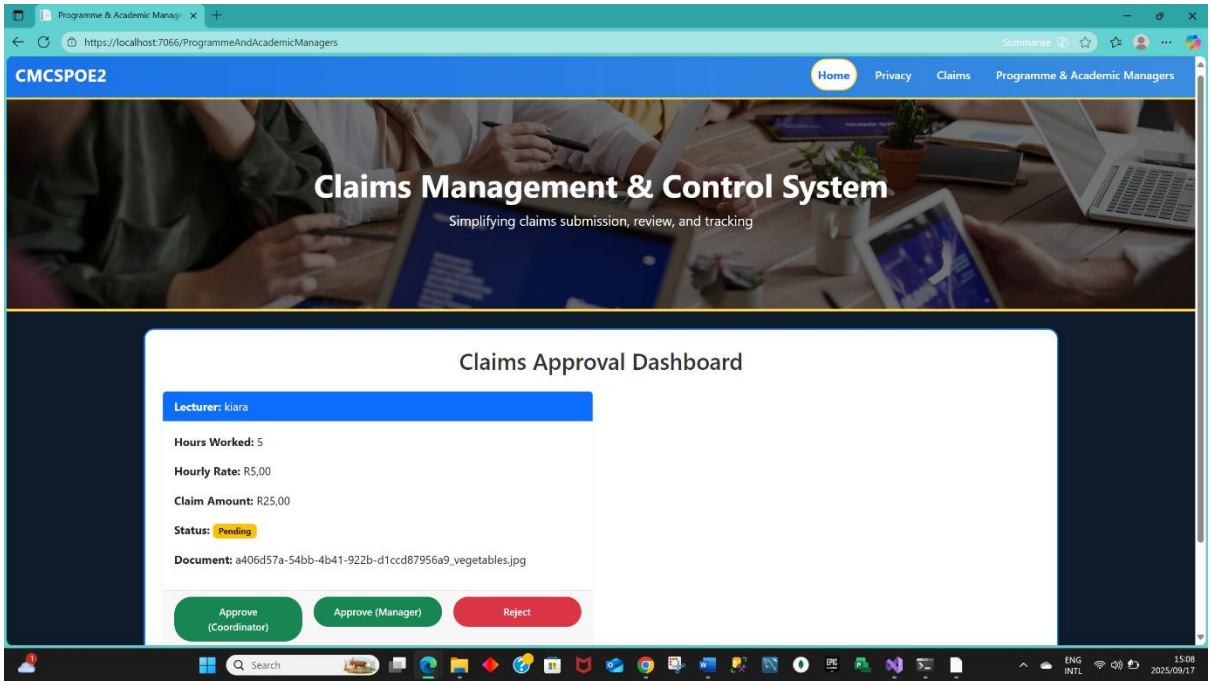
(Caulfield, 2020)



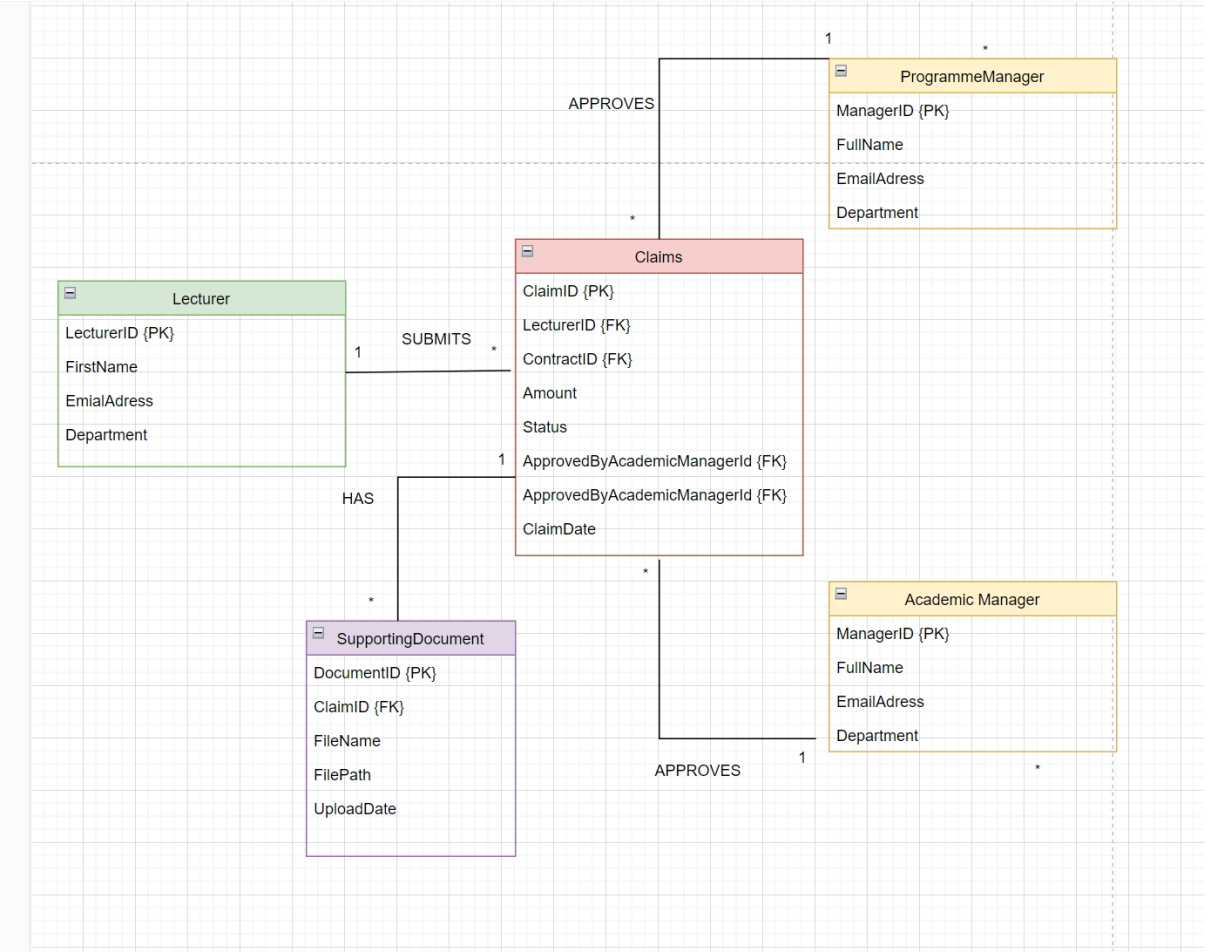



All Claims - CMCSPOE2
https://localhost:7066/Claims
CMCSPOE2
Home
Privacy
Claims
Programme & Academic Managers
Claims Management & Control System
Simplifying claims submission, review, and tracking
All Claims
Lecturer Name
Hours Worked
Hourly Rate
Claim Amount
Status
Supporting Document
kiara
5
5
25
Pending
a406d57a-54bb-4b41-922b-d1ccd87956a9_vegetables.jpg
View
Submit New Claim
© 2025 - CMCSPOE2 | Privacy Policy


https://localhost:7066/ProgrammeAndAcademicManagers
Claims Management & Control System
Simplifying claims submission, review, and tracking
Claims Approval Dashboard
Lecturer: kiara
Hours Worked: 5
Hourly Rate: R5,00
Claim Amount: R25,00
Status: Approved
Document: a406d57a-54bb-4b41-922b-d1ccd87956a9_vegetables.jpg
Verified By: Manager Jane (Academic Manager)


Programme & Academic Manage
https://localhost:7066/ProgrammeAndAcademicManagers
Summarise
CMCSPOE2
Home
Privacy
Claims
Programme & Academic Managers
Claims Management & Control System
Simplifying claims submission, review, and tracking
Claims Approval Dashboard
Lecturer: kiara
Hours Worked: 5
Hourly Rate: R5,00
Claim Amount: R25,00
Status: Pending
Document: a406d57a-54bb-4b41-922b-d1ccd87956a9_vegetables.jpg
Approve (Coordinator)
Approve (Manager)
Reject
Search
ENG
INTL
15:08
2025/09/17

Screenshots of Class Diagram:

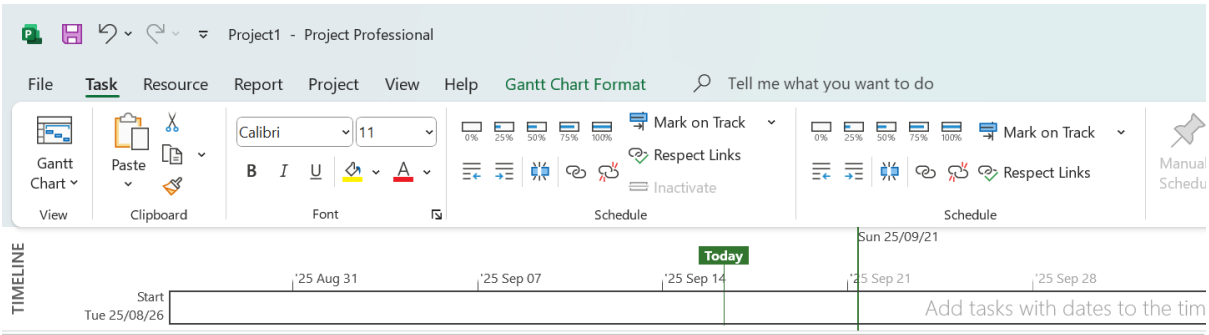


(Caulfield, 2020)

Explanation:

My UML Class Diagrams shows what attributes each model has and its multiplicities and the verbs that go along with it as well as the 1 to many multiplicities and arrows going along with them.

Screenshots of WBS: (Caulfield, 2020)

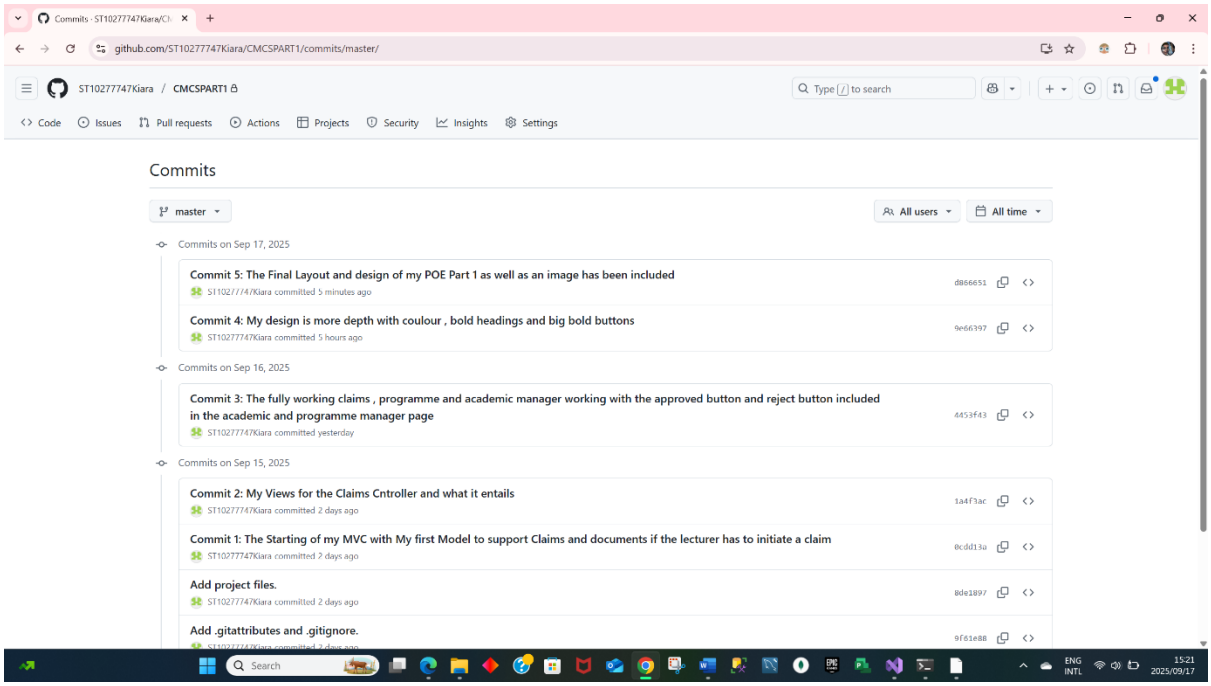


GANTT CHART

	WBS	Task Name	Deliverables	Duration	Start	Finish
1	**0**	**CMCS**	**Project plan & deliverables**	**50 days?**	**Tue 25/08/26**	**Mon 25/11/03**
2	**1**	**Project Initiation**	**Initiation phase outputs**	**3 days?**	**Tue 25/08/26**	**Thu 25/08/28**
3	1.1	Identify Project Team	Team list	2 days	Tue 25/08/26	Wed 25/08/27
4	1.2	Define Project Objectives	Objectives document	2 days	Wed 25/08/27	Thu 25/08/28
5	1.3	Confirm Project Scope	Scope approval document	2 days	Wed 25/08/27	Thu 25/08/28
6	**2.**	**Project Planning**	**Approved project plan**	**6 days?**	**Fri 25/08/29**	**Fri 25/09/05**
7	2.1	Develop Work Breakdown Structure (WBS)	WBS document	2 days?	Fri 25/08/29	Mon 25/09/01
8	2.2	Develop Schdeule and Milestones	Project schedule	1 day?	Thu 25/09/04	Thu 25/09/04
9	2.3	Resource Allocation	Resource plan	2 days?	Thu 25/09/04	Fri 25/09/05
10	**3.**	**Requirements Gatherings**	**Requirements document**	**7 days?**	**Mon 25/09/08**	**Tue 25/09/16**
11	3.1	Conduct lecturer consultations	Consultation notes	1 day?	Wed 25/09/10	Wed 25/09/10
12	3.2	Collect supporting documents	Uploaded documents	1 day?	Fri 25/09/12	Fri 25/09/12
13	3.3.	Define user roles (lecturer/student)	User role	1 day?	Tue 25/09/16	Tue 25/09/16
14	**4.**	**System Design**	**Design specifications**	**7 days?**	**Wed 25/09/17**	**Thu 25/09/25**
15	4.1	Design System Architecture	Detailed system architecture document	1 day?	Thu 25/09/18	Thu 25/09/18
16	4.2	Design Database Schema	Entity Relationship Diagram (ERD) & table schemas	3 days?	Fri 25/09/19	Tue 25/09/23
17	4.3	Develop User Interface Designs	Role-specific interface mockups	1 day?	Thu 25/09/25	Thu 25/09/25
18	**5.**	**Development**	**Functional system modules**	**20 days?**	**Fri 25/09/26**	**Thu 25/10/23**
19	5.1	Backend development (claims workflow)	Claim module	1 day?	Thu 25/10/09	Thu 25/10/09
20	5.2	Backend development (document upload)	Document upload module	4 days?	Fri 25/10/10	Wed 25/10/15

20	5.2	Backend development (document upload)	Document upload module	4 days?	Fri 25/10/10	Wed 25/10/15
21	5.3	Frontend – Lecturer dashboard	Lecturer dashboard	4 days?	Fri 25/10/10	Wed 25/10/15
22	5.4.	Frontend – Academic manager portal	Manager portal	2 days?	Thu 25/10/16	Fri 25/10/17
23	5.5	Role-based access control (RBAC)	User role permissions	3 days?	Fri 25/10/17	Tue 25/10/21
24	5.6	System integration	Integrated modules	2 days?	Tue 25/10/21	Wed 25/10/22
25	**6.**	**Testing & Quality Assurance**	**QA reports**	**8 days?**	**Thu 25/10/23**	**Mon 25/11/03**
26	6.1	Unit Testing (claims& docs)	Unit test results	3 days	Thu 25/10/23	Mon 25/10/27
27	6.2	System testing (user roles& claims)	System test results	3 days?	Tue 25/10/28	Thu 25/10/30
28	6.3	User Acceptance Testing (UAT)	UAT sign-off	2 days?	Fri 25/10/31	Mon 25/11/03
29	6.4	Workflow Testing	Workflow test report	1 day?	Mon 25/11/03	Mon 25/11/03

Screenshot of GitHub Commits:



Reference List:

Caulfield, J. (2020) *Reference a Website in Harvard Style | Templates & Examples*. Available at: https://www.scribbr.co.uk/referencing/harvard-website-reference/ (Accessed: 17 September 2025).

Anglia Ruskin University (2025) *Harvard System of Referencing Guide*. Available at: https://library.aru.ac.uk/referencing/harvard.htm (Accessed: 17 September 2025).