

Dianca Jade Naidu
ST10261874
7 September 2024

2024



Report: Prototype of the Contract Monthly Claim System

PROGRAMMING 2B
PORTFOLIO OF EVIDENCE
PART 1

Contents

1. Introduction	2
Overview of the CMCS	2
Purpose	2
Scope of the Prototype Design:	2
2. Design Documentation	2
Detailed Design Choices	2
Assumptions and Constraints	15
3. UML Class Diagram for Databases	16
Overview of UML Class Diagram:	16
Data Requirements of the CMCS	16
UML Class Diagram	17
4. Project Plan	18
Project Plan Overview	18
Timeline for Prototype Development:	18
Realistic and Achievable Milestones:	18
5. GUI Interface Design	19
Front-End Prototype Features:	19
System Reliability and Consistency:	20
6. Version Control Strategy	20
GitHub Link	20
Commit History	20
Descriptive Commit Messages	21
7. Conclusion	22
Reflection on the Prototype	22
Future Development Phases:	22
8. Annexures	22
AI Usage Breakdown:	22
9. References	23

1. Introduction

Overview of the CMCS:

The Contract Monthly Claim System (CMCS) is designed to speed up the submission and approval of claims for an array of roles, such as admins, academic managers, programme coordinators, and lecturers. Lecturers may upload supporting documentation, track their claims' progress, and submit claims via the system. Administrators supervise user obligations and permissions in addition to verifying and approving claims; program coordinators and academic managers can verify and approve claims.

Purpose:

The prototype attempts to reduce administrative overhead and enhance the transparency of the claims process by offering a streamlined, user-friendly interface for filing, organising, and approving claims.

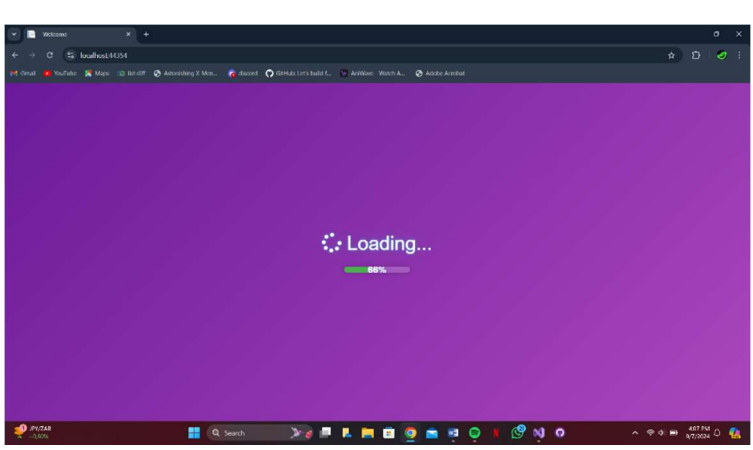
Scope of the Prototype Design:

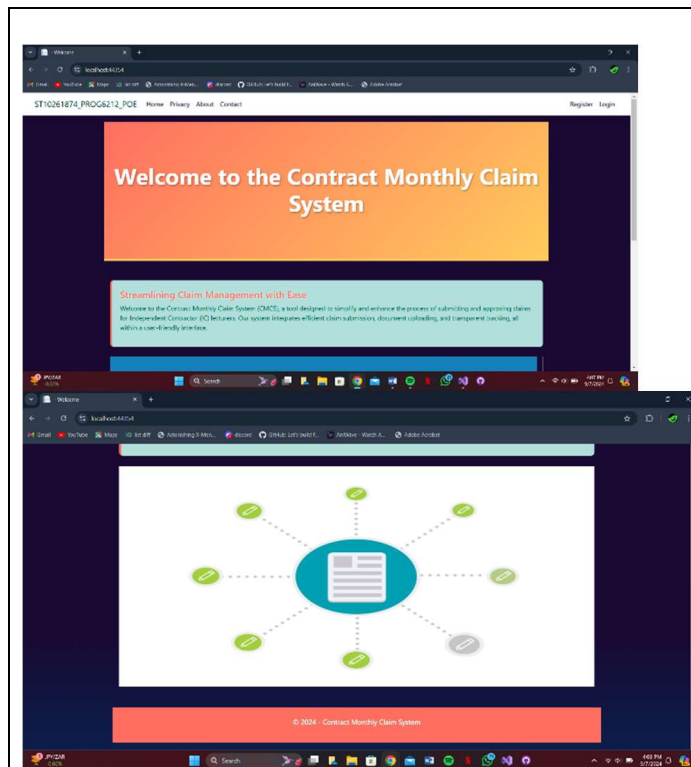
Role-based access control, document upload, claims tracking, and claim submission are all included in the prototype. Advanced analytics and comprehensive administrative reports are not included, but they will be included in other parts of the POE.

2. Design Documentation

Detailed Design Choices

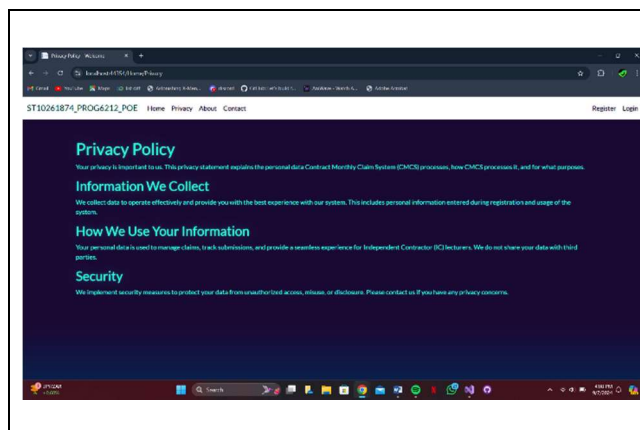
Screenshots of GUI

	<h3><u>Loading Screen</u></h3> <ul style="list-style-type: none">• As soon as the app starts, users can see a splash screen.• This loading page looks visually appealing to users which allures users from the start and keeps them looking forward to the rest of the application.
--	--



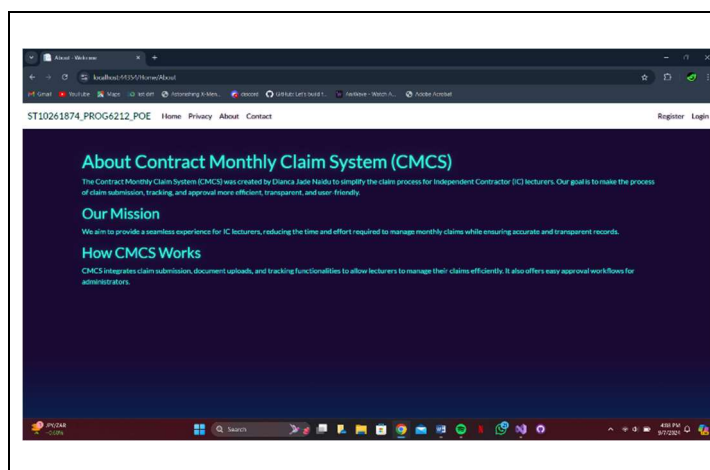
Home Page

- As you can see, no one has logged in, so this is from the perspective of any user.
- The use of different colours is once again used for the purpose of being captivating to one's eye.
- The home page also has the usage of a Carousel. I have added 3 GIFs which automatically change. However, a user can choose to view the next GIF if they want.



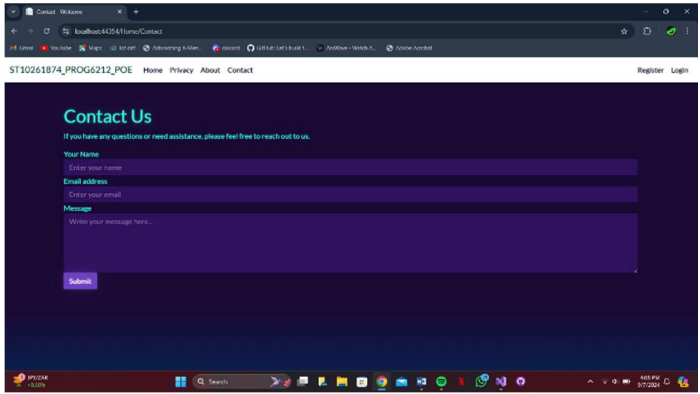
Privacy Page

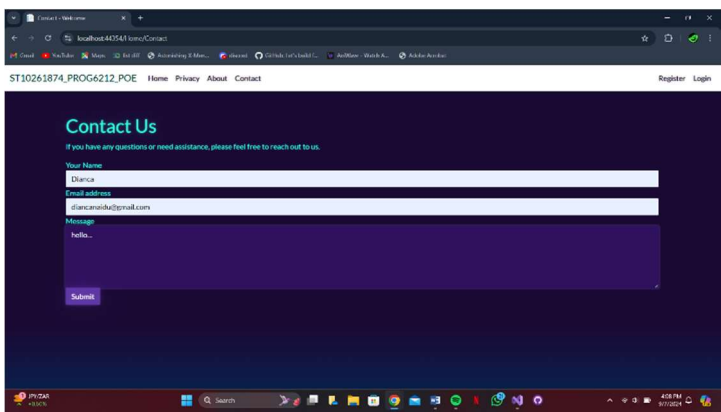
- This is a very important page as it ensures users that their privacy and security is being kept safe.

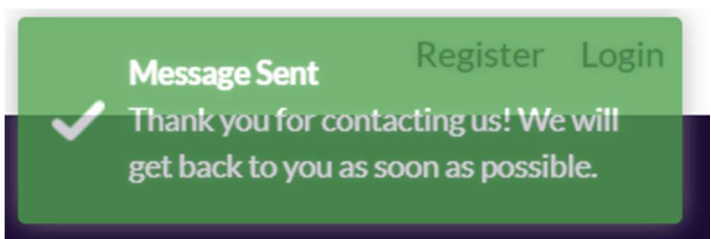


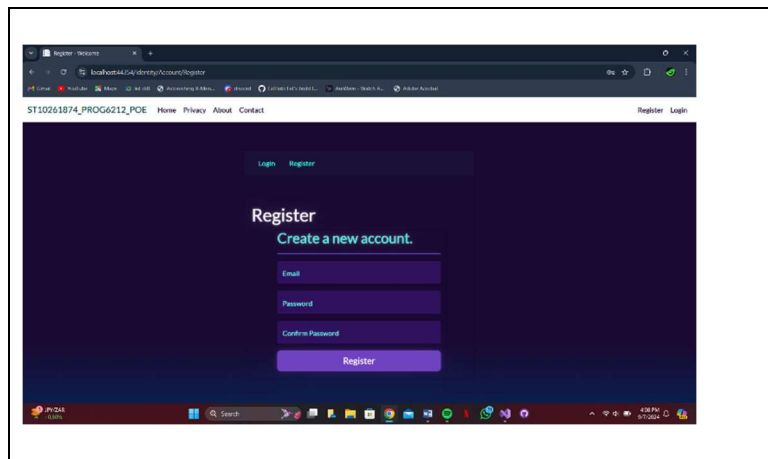
About Page

- This page gives information about what this application is about, what the mission is and how the app works.

	<h3><u>Contact Us Page</u></h3> <ul style="list-style-type: none"> • For the Contact Us Page, user input is required. • This allows for users to interact with the system even before logging in.
---	---

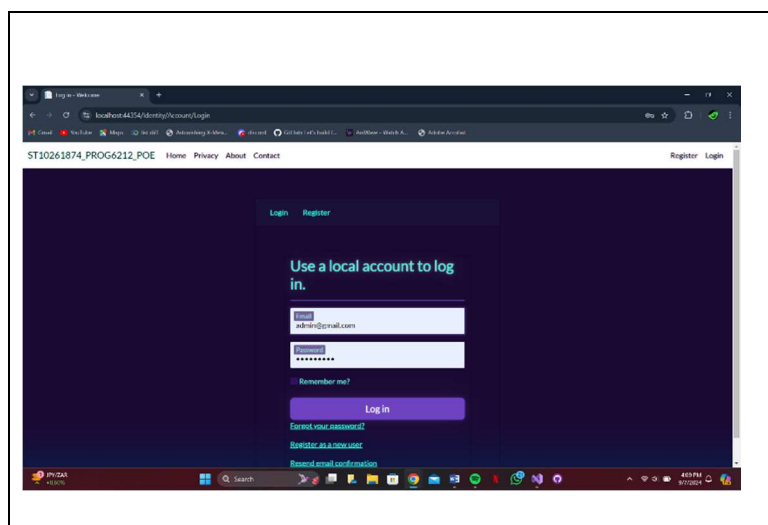
	<h3><u>Contact Us Page</u></h3> <ul style="list-style-type: none"> • Here shows an example of a user basically “logging a ticket” to get assisted.
--	---

	<h3><u>Contact Us Page</u></h3> <ul style="list-style-type: none"> • As you can see, a toastr pops up on the top right hand corner of the page. • This is when the user clicks on the Submit button after filling in their details and message.
---	---



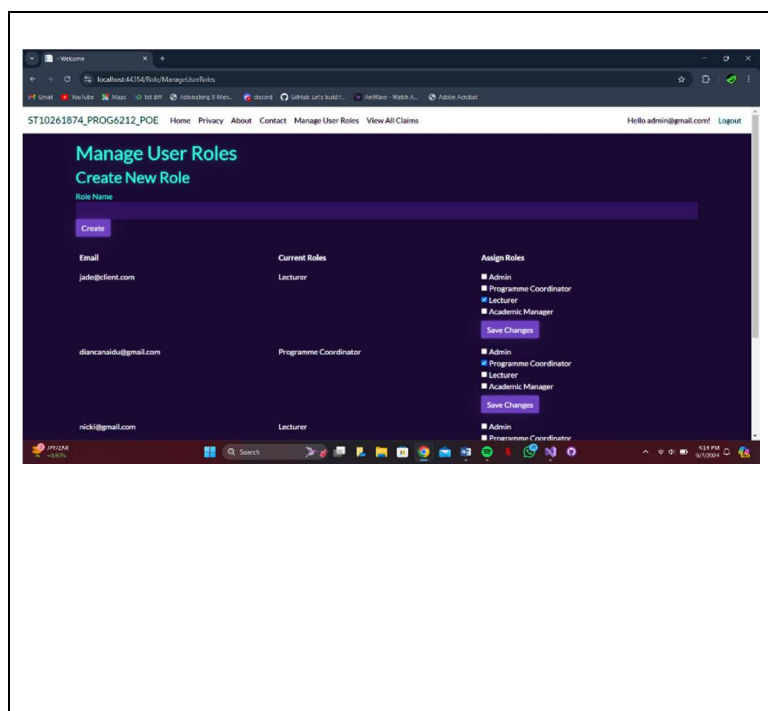
Register Page

- The design of the register page is very simple and user friendly.
- This less complex layout makes it very easy for users to register for the first time.




Login Page

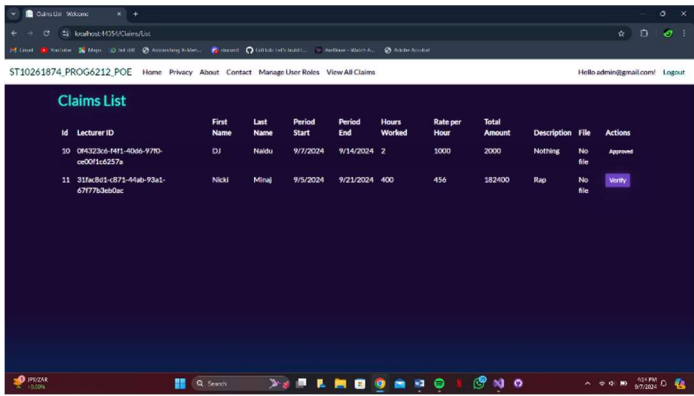
- The design of the login page is very similar to the register page, which again is to make logging in quick and easy for users.

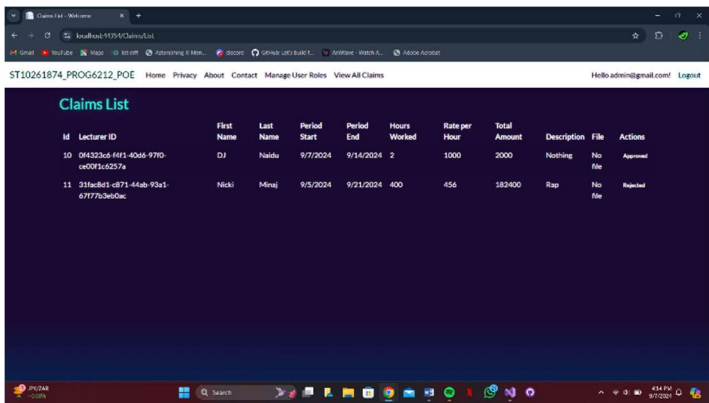


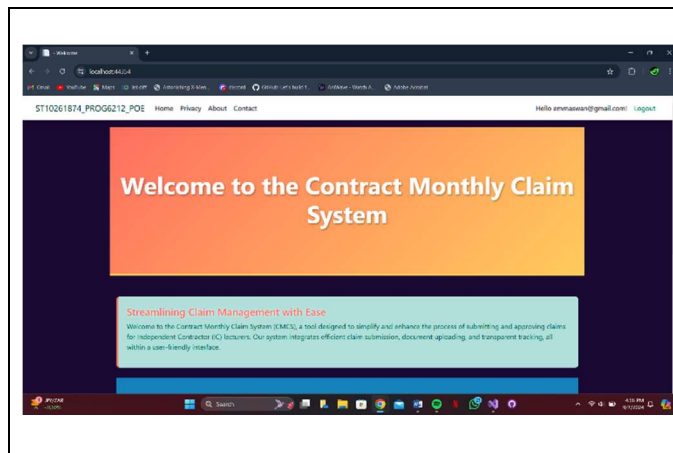
Manage User Roles Page

- As shown, I have logged in as Admin which is the only user who is able to view the Manage User Roles Page.
- This page is to create roles as well as to assign roles to users.
- The use of CheckBoxes makes it easier for the admin to assign roles.
- Also, my reasoning for using a CheckBox instead of a ComboBox for instance, is simply because with CheckBoxes, you are

	<p>able to choose more than one option (Shaw, 2024). For example, a user can be both a lecturer as well as an academic manager.</p>
---	---

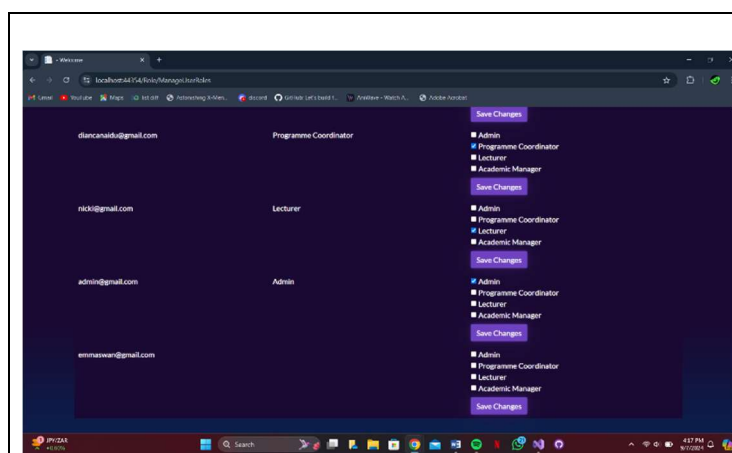
	<p>View All Claims Page</p> <ul style="list-style-type: none"> The admin has access to view the list of claims that a lecturer has made. The layout of this page is simple and clear. Headings are in bold font which makes it look neat and organised.
--	--

	<p>View All Claims Page</p> <ul style="list-style-type: none"> The admin also has access verify, approve and reject claims as you can see.
---	--



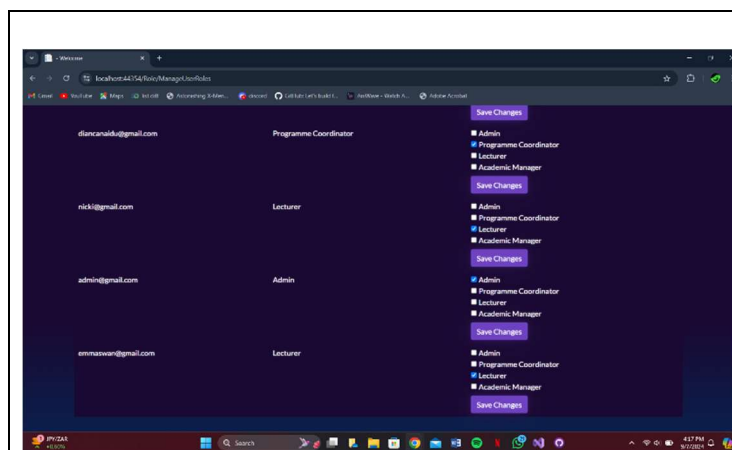
Home Page as a new user

- A user that has not been assigned a role will still be able to log in but will not be able to view much.
- They will have to wait for the Admin to assign a role to them.



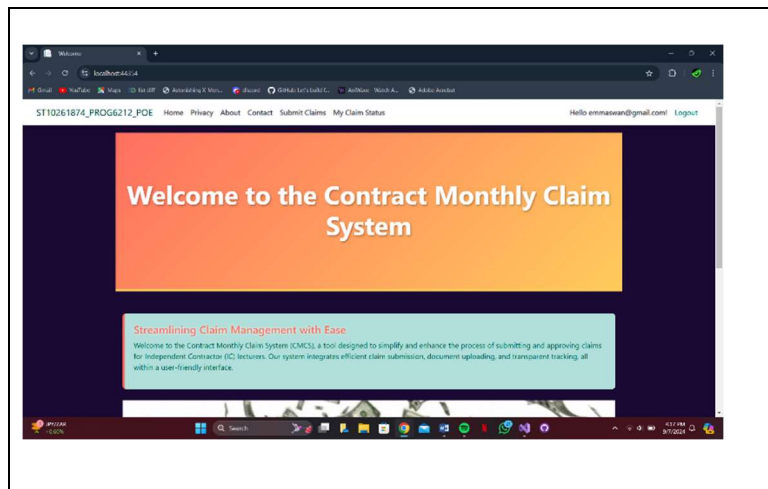
Assign a new user

- The screenshot on the left shows the Manage User Roles page.
- As you can see, emmaswan@gmail.com has not been assigned a role as yet.



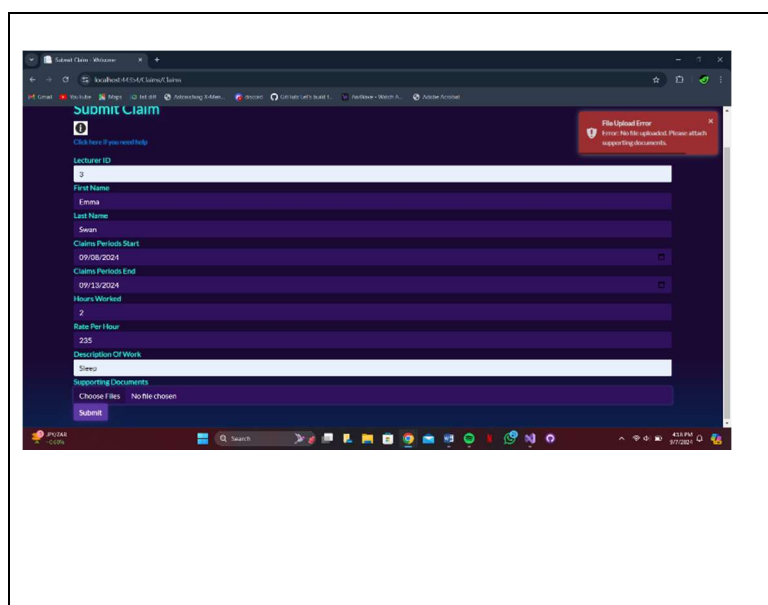
Assign a new user

- The screenshot on the left now shows that a role has been assigned to this user.
- The CheckBox of Lecturer is now ticked.



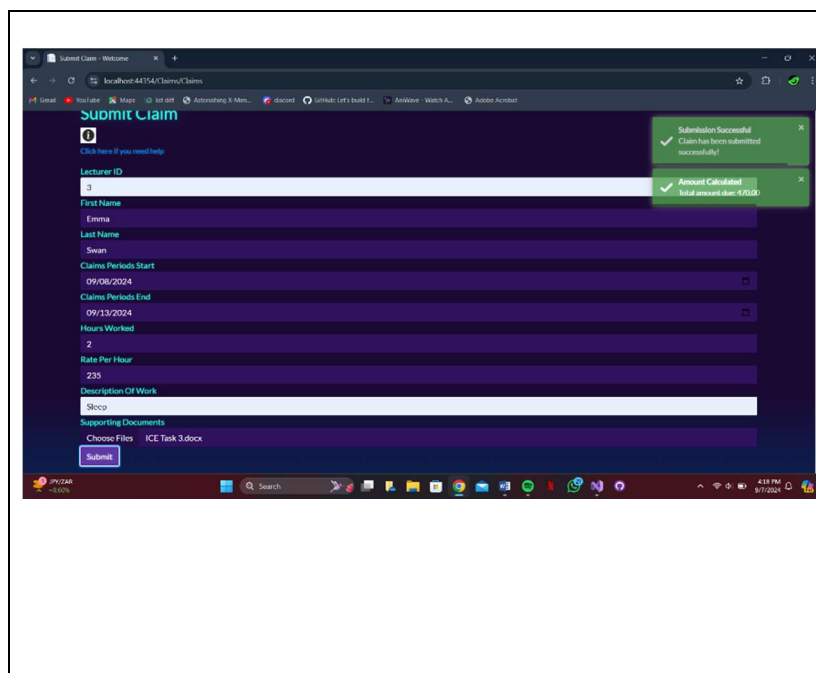
Home Page as Lecturer

- Now that emma@emma@gmail.com has been assigned a role, they can now log in and are able to view additional pages.
- The role of a lecturer is to have access to the Submit Claims page and My Claim Status page.



Submit Claims Page

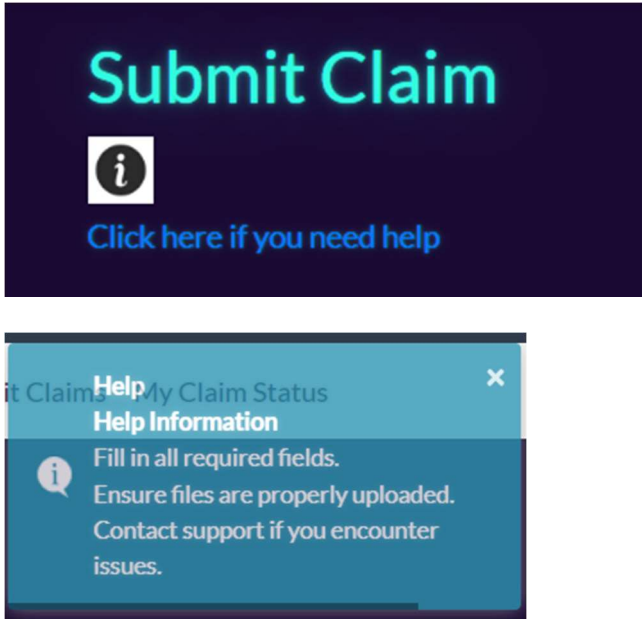
- This is the view of the Submit Claims Page.
- A toastr pops up on the top right hand corner of the page when a user does not fill in all details and clicks on the submit button.
- For example, I had purposely not chosen to upload a file. Hence, that error message popped up.

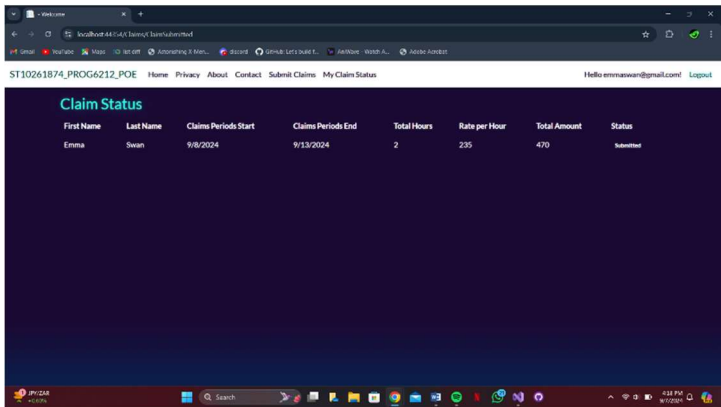


Submit Claims Page

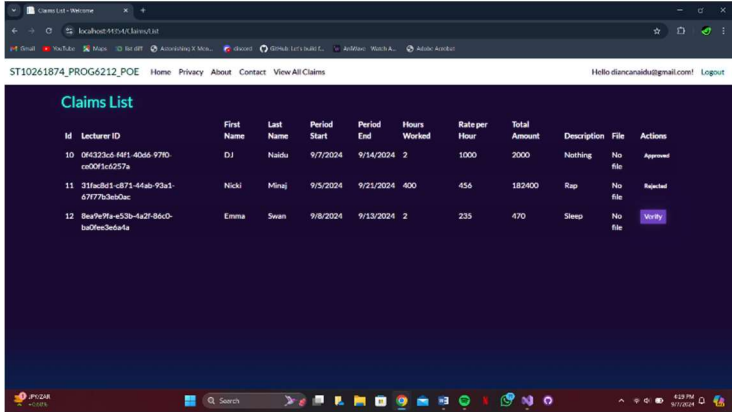
- I have now chosen a file to upload and as soon as I clicked on the “Submit” button, the toasts in green popped up.
- The Green colour indicates success and the red toastr which was shown previously showed an error.
- The use of different colours, those two colours specifically, makes it easy for the

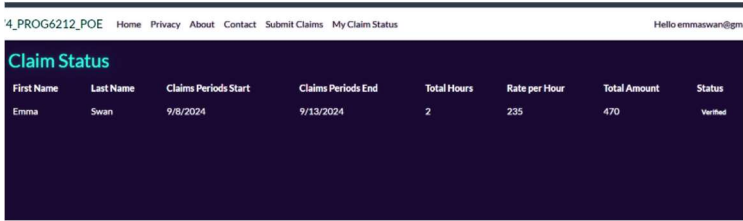
	<p>user to understand and differentiate between the types of messages.</p> <ul style="list-style-type: none"> The total amount has also been calculated accordingly.
--	---

	<p>Submit Claims Page</p> <ul style="list-style-type: none"> This page can be mildly confusing for a first time Lecturer. This is why I have added an image of an icon which users can click on. As soon as the user clicks on this icon, a toastr will pop up which will assist the user. I have chosen to put this in a blue colour so that users do not get confused and think that it is either an error or success message.
--	--

	<p>Submit Claims Page</p> <ul style="list-style-type: none"> As soon as the user clicks submit and gets the successful toastr messages, they will automatically be rerouted to the Claim Status page. As you can see, the Status shows “Submitted”. This status will change according to what the Admin, Academic Manager or Programme Coordinator chooses.
---	--

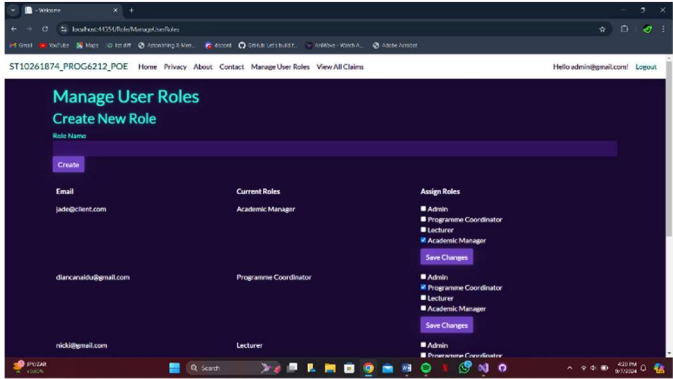
	<ul style="list-style-type: none"> The total amount that has been calculated is also now shown in this table.
--	--

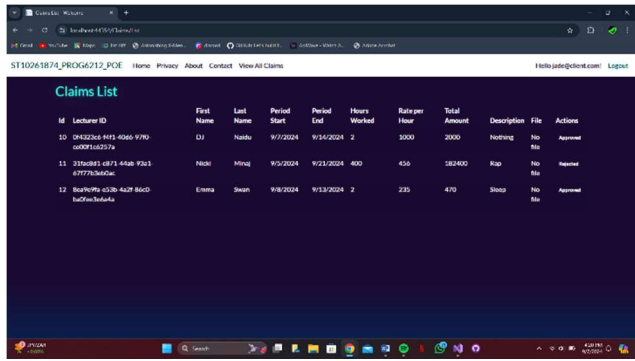
	<p>View All Claims Page</p> <ul style="list-style-type: none"> I have now logged in as the Programme Coordinator. As you can see, there is a “Verify” button now visible.
--	---

	<p>Claim Status Page</p> <ul style="list-style-type: none"> As soon as the Programme Coordinator clicked on the verify button, the status of the claim changed to Verified.
---	--

	<p>View All Claims Page</p> <ul style="list-style-type: none"> Once either the Programme Coordinator, Academic Manager or Admin clicks on the “Verify” button, both the Approve and Reject buttons show up. Once again, I have used the colours Red and
---	---

	Green to make it user friendly.
--	---------------------------------

	<p>Manage User Roles Page</p> <ul style="list-style-type: none"> For demonstration purposes, I have now logged in as Admin to assign the role of Academic Manager.
--	---



ID	Lecturer ID	First Name	Last Name	Period Start	Period End	Hours Worked	Rate per Hour	Total Amount	Description	File	Actions
10	043226d-64f1-4066-970d-000f16227a	DJ	Naidu	9/7/2024	9/14/2024	2	1000	2000	Nothing	No file	Approved
11	3766082-0471-4548-93d1-43777b340e	Nishi	Ming	9/5/2024	9/21/2024	400	456	182400	Rip	No file	Approved
12	8a0e9f9-653e-4a28-86cd-ba29ee3e4e4a	Emma	Swan	9/8/2024	9/13/2024	2	235	470	Stop	No file	Approved

View All Claims Page

- I have now logged in as Academic Manager and Approved the Status.
- This shows that both Programme Coordinator and Academic Manager do not have to approve or reject claims. As long as one of them does an action, it will still work.
- This reduces repetitive work.

	<h3>My Claim Status Page</h3> <ul style="list-style-type: none"> This is the view of the Claim Status page as a lecturer. Each screenshot is taken from two different lecturer's views. A lecturer can only view their own claims which ensures security and protection of data (IBM, n.d.).

Screenshots of Database

- Data in the table:

Table: dbo.AspNetUsers-

	Id	UserName	NormalizedUs...	Email	NormalizedEm...	EmailConfirmed	PasswordHash	SecurityStamp	
	0f4323c6-f4f1-4...	jade@client.com	JADE@CLIENT...	jade@client.com	JADE@CLIENT...	True	AQAAAAIAAYa...	WWSJGGCQT37...	7
	2dd53811-3a24...	diancanaidu@g...	DIANCANAIDU...	diancanaidu@g...	DIANCANAIDU...	True	AQAAAAIAAYa...	QEAPNXIMOB...	2
	31fac8d1-c871-...	nicki@gmail.com	NICKI@GMAIL...	nicki@gmail.com	NICKI@GMAIL...	True	AQAAAAIAAYa...	4TWBNNPZXM...	6
	6f5cbf4b-9e71-...	admin@gmail.c...	ADMIN@GMAIL...	admin@gmail.c...	ADMIN@GMAIL...	True	AQAAAAIAAYa...	S7NN2GSJGDU...	6
	8ea9e9fa-e53b-...	emmaswan@g...	EMMASWAN@...	emmaswan@g...	EMMASWAN@...	True	AQAAAAIAAYa...	JVCKP2QYMO7...	6

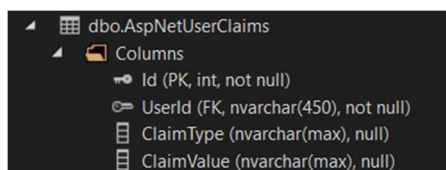
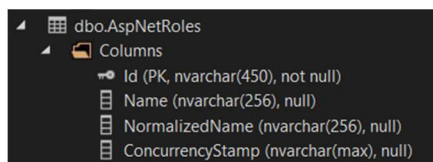
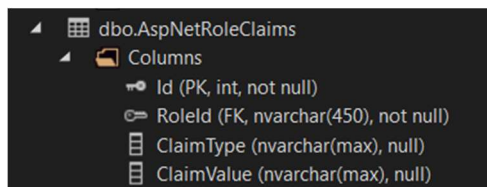
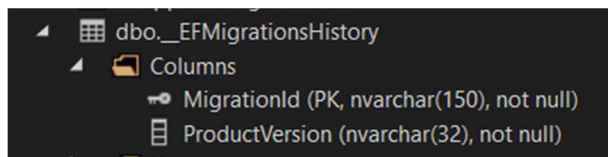
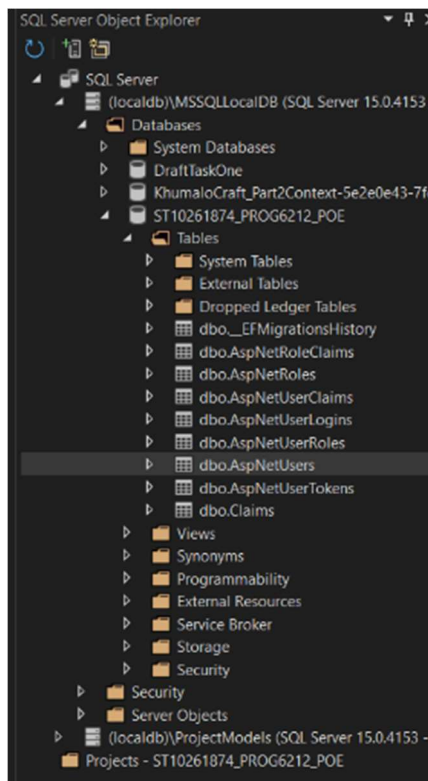
Table: dbo.Claims-

Id	FirstName	LastName	ClaimsPeriodsS...	ClaimsPeriodsE...	HoursWorked	RateHour	TotalAmount	Desc
6-f4f1-4...	DJ	Naidu	9/7/2024 12:00:...	9/14/2024 12:0...	2	1000	2000	Noth
1-c871-...	Nicki	Minaj	9/5/2024 12:00:...	9/21/2024 12:0...	400	456	182400	Rap
fa-e53b-...	Emma	Swan	9/8/2024 12:00:...	9/13/2024 12:0...	2	235	470	Sleep

Table: dbo.AspNetRoles-

Id	Name	NormalizedNa...	ConcurrencySt...
2490f519-7b2e-...	Admin	ADMIN	NULL
5e5fe93c-0ad7-...	Programme Co...	PROGRAMME C...	NULL
65d57d57-a1ee...	Lecturer	LECTURER	NULL
b663f263-6b35...	Academic Man...	ACADEMIC MA...	NULL

- List of all tables-



dbo.AspNetUserLogins
Columns
LoginProvider (PK, nvarchar(128), not null)
ProviderKey (PK, nvarchar(128), not null)
ProviderDisplayName (nvarchar(max), null)
UserId (FK, nvarchar(450), not null)

dbo.AspNetUserRoles
Columns
UserId (PK, FK, nvarchar(450), not null)
RoleId (PK, FK, nvarchar(450), not null)

dbo.AspNetUsers
Columns
Id (PK, nvarchar(450), not null)
UserName (nvarchar(256), null)
NormalizedUserName (nvarchar(256), null)
Email (nvarchar(256), null)
NormalizedEmail (nvarchar(256), null)
EmailConfirmed (bit, not null)
PasswordHash (nvarchar(max), null)
SecurityStamp (nvarchar(max), null)
ConcurrencyStamp (nvarchar(max), null)
PhoneNumber (nvarchar(max), null)
PhoneNumberConfirmed (bit, not null)
TwoFactorEnabled (bit, not null)
LockoutEnd (datetimeoffset(7), null)
LockoutEnabled (bit, not null)
AccessFailedCount (int, not null)

dbo.AspNetUserTokens
Columns
UserId (PK, FK, nvarchar(450), not null)
LoginProvider (PK, nvarchar(128), not null)
Name (PK, nvarchar(128), not null)
Value (nvarchar(max), null)

dbo.Claims
Columns
Id (PK, int, not null)
LecturerID (nvarchar(max), not null)
FirstName (nvarchar(max), not null)
LastName (nvarchar(max), not null)
ClaimsPeriodsStart (datetime2(7), not null)
ClaimsPeriodsEnd (datetime2(7), not null)
HoursWorked (float, not null)
RateHour (float, not null)
TotalAmount (float, not null)
DescriptionOfWork (nvarchar(max), not null)
Status (nvarchar(max), not null)
VerifiedBy (nvarchar(max), not null)
VerifiedOn (datetime2(7), null)
IsApproved (bit, not null)
IsVerified (bit, not null)
IsRejected (bit, not null)

Assumptions and Constraints

Assumptions:

1. User Roles and Permissions:
 - Only Lecturers can submit claims, view the status of their claims, and upload supporting documents.
 - Programme Coordinators and Academic Managers have access to verify, approve, or reject claims submitted by lecturers.
 - Administrators manage user roles, ensuring that permissions are correctly assigned to each user as well as have access to verify, approve, or reject claims submitted by lecturers.
2. Access Control:
 - Role-based access control is fully functional, and each user can only perform the actions associated with their role (e.g., only Admin can modify roles).
3. Claims Calculation:
 - The calculation of claim amounts is based on rates per hour and the amount of hours the lecturer has worked.
4. Supporting Documents:
 - Lecturers are expected to upload supporting documents in standard formats (e.g., PDFs, Word documents, images) when submitting claims.
5. System Availability:
 - The system assumes constant availability and uptime for claims submission and approval processes.

Constraints:

1. Authentication:
 - Users must be authenticated via a login system, and any unauthorized user cannot access the claims submission or approval features.
2. File Upload Limits:
 - Uploaded supporting documents must adhere to file size restrictions (e.g., maximum file size of 5MB) to ensure efficient storage management (Nixon, 2022).
3. Role Management:
 - Only administrators can assign or change user roles, ensuring a controlled and secure environment where permissions cannot be altered by unauthorized users.
4. Scalability:
 - The current system is designed for small to medium-sized educational institutions, and future scaling will require changes to the database structure and cloud storage capacity.

3. UML Class Diagram for Databases

Overview of UML Class Diagram:

The UML Class Diagram is centered around the base class, which is called `ApplicationUser`, from which all user types inherit. This shows inheritance (Bell, 2023). These user types include `Admin`, `Lecturer`, `AcademicManager`, and `ProgrammeCoordinator`. The `Claims` class represents the claims submitted by lecturers and is linked to other roles that manage, review or submit claims.

Data Requirements of the CMCS

Classes:

`ApplicationUser`, `Lecturer`, `Admin`, `Academic Manager`, `Programme Coordinator`, `Claims`

Attributes:

ApplicationUser:

`UserID: int {Primary Key}`
`FirstName: string`
`LastName: string`
`Email: string`
`Password: string`
`Role: string`

Lecturer:

`LecturerID: int {Primary Key}`
`UserID: int {Foreign Key}`
`TotalClaimsSubmitted: int`

Admin:

`AdminID: int {Primary Key}`
`UserID: int {Foreign Key}`

Academic Manager:

`AcademicManagerID: int {Primary Key}`
`UserID: int {Foreign Key}`
`TotalClaimsApproved: int`
`TotalClaimsRejected: int`

ProgrammeCoordinator:

`ProgrammeCoordinatorID: int {Primary Key}`
`UserID: int {Foreign Key}`
`TotalClaimsApproved: int`
`TotalClaimsRejected: int`

Claims:

ClaimsID: int {Primary Key}

LecturerID: int {Foreign Key}

ClaimsPeriodsStart: DateTime

ClaimsPeriodsEnd: DateTime

HoursWorked: double

RateHour: double

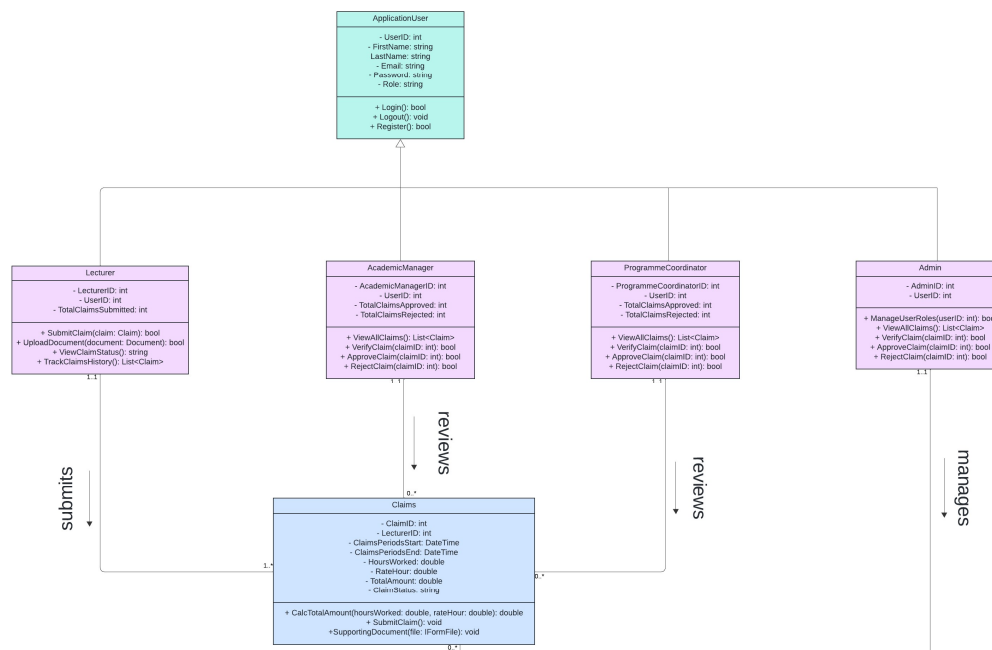
TotalAmount: double

ClaimStatus: string

Relationships:

- Admin, ProgrammeCoordinator, AcademicManager and Lecturer inherit from ApplicationUser.
- One and only One Lecturer SUBMITS One or Many claims. A Lecturer cannot submit zero claims.
- One and only One AcademicManager REVIEWS Zero or Many claims.
- One and only One ProgrammeCoordinator REVIEWS Zero or Many claims.
- One and only One AcademicManager MANAGES Zero or Many claims.

UML Class Diagram



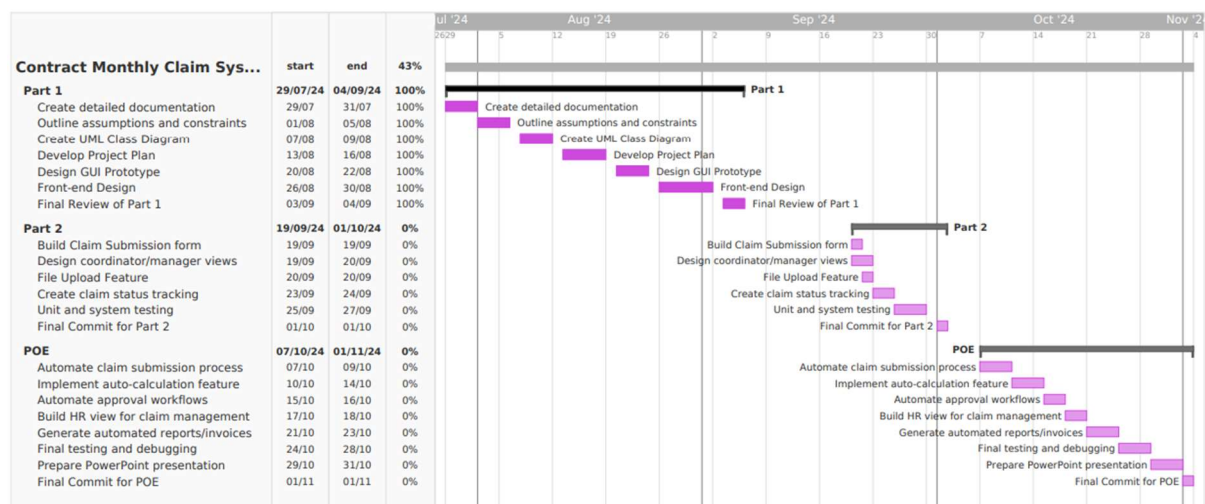
4. Project Plan

Project Plan Overview:

The tasks needed to finish the prototype between July 22, 2024, and November 1, 2024 are listed in the project plan. Important tasks include outlining assumptions and constraints, deciding on design choices, putting the system's functionality into practice, and testing. Dependencies are included into each step to guarantee a seamless transition from one activity to the next. Milestones for class diagrams, GUI design, providing functionality for different roles, and system testing are all included in the plan. This guarantees that all necessary parts are finished on schedule, resulting in the prototype's final submission.

Timeline for Prototype Development:

Gantt Graph



Realistic and Achievable Milestones:

Part 1

1. UML Class Diagram completed (09-Aug-24): All classes, attributes and relationships are finalised.
2. Front-end Design Complete (30-Aug-24): Whatever is required for part 1 is completed.
3. Final Review of Part 1 (4-Sep-24): Changes made, errors fixed and documentation finalised.

Part 2

1. Views for all roles completed (20-Sep-24): Views for Admin, Lecturer, Programme Coordinator and Academic Manager are complete.
2. Unit and system testing (27-Sep-24): System has been tested for bugs and any additional changes that need to be made.

3. Final Commit for Part 2 (1-Oct-24): Everything required for Part 2 is complete and ready to push the final commit to GitHub

POE

1. Final Unit and System Testing (28-Oct-24): Made changes based on feedback from lecturer, added new features, website is complete with no bugs.
2. Prepare PowerPoint Presentation (31-Oct-24): Final Presentation has been done to show how the website functions and to showcase it.
3. Final Commit for Part 3 (1-Nov-24): Everything required for Part 3 is complete and ready to push the final commit to GitHub

5. GUI Interface Design

Front-End Prototype Features:

❖ Claims Submission Functionality:

Lecturers are able to submit their claims using the Claims Submission capability. Lecturers enter the necessary data (such as hours worked, rates, etc.) using an easy-to-use form. One click submits claims, automating the submission procedure and ensuring user-friendliness. In order to improve the user experience, lecturers are notified by Toastr notifications when their claim is successfully submitted or if there are any difficulties encountered during the submission process.

❖ Document Upload Feature:

Using the Document Upload function, instructors may support their claims with supporting documents like schedules or records. Several file types are supported via the upload interface, and upon successful upload, a Toastr notification is displayed. When processing claims, this feature makes sure the approval team gets all the paperwork needed for validation.

❖ Approval Process for Claims:

The interface designed for the Approval Process for Claims is intended for use by Program Coordinators, Academic Managers, and Admins who are in charge of verifying, accepting, or rejecting claims that have been filed. Users can easily get the detailed information about a claim by clicking on it.

❖ Transparent Claim Status Tracking:

The Claim Status Tracking feature allows lecturers to view the progress of their submitted claims. They can log in to see real-time updates, with claims marked as “Submitted”, “Approved” or “Rejected”.

System Reliability and Consistency:

The system uses strong backend procedures to maintain data integrity (Cote, 2021) and constant performance even when several people submit, review, and approve claims.

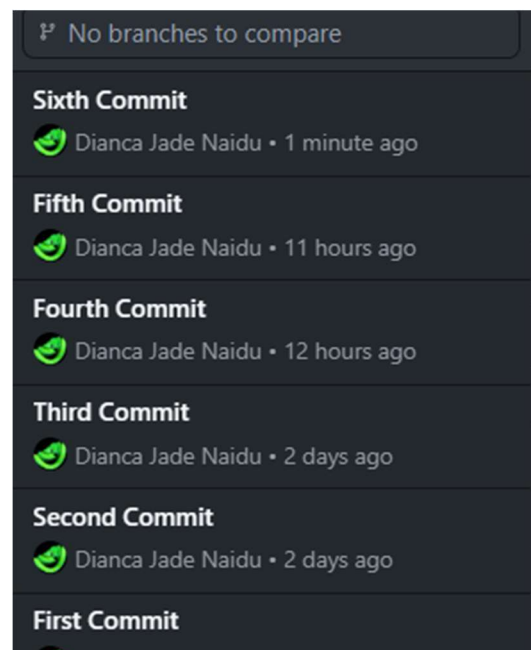
Key components that provide system consistency and dependability (geeksforgeeks, 2024) are as follows:

Access control based on roles to stop illegal access to private claim data. Transaction management is necessary to prevent data loss or inconsistency throughout system operations, both during claim submission and approval (Schneider & Smalley, 2024). Toastr's error handling and notifications make sure users are notified of any problems in real time and prevent system breakdowns.

6. Version Control Strategy

GitHub Link: <https://github.com/VCWVL/prog6212-part-1-ST10261874.git>

Commit History



Descriptive Commit Messages

First Commit ↕
Added most controllers, views and models

Second Commit ↕
Made additional improvements to the displaying, such as adding buttons, making it user friendly

Third Commit ↕
Customised home page, made it much more appealing to users

Fourth Commit ↕
Added more toasts, also added a help icon to the claims page

Fifth Commit ↕
Added extensive comments to the entire program.

Sixth Commit ↕
Final changes made to the program

7. Conclusion

Reflection on the Prototype:

The essential features needed for the CMCS are effectively demonstrated by the prototype. There are still quite a few improvements and additional features that are yet to be added based on feedback from users as it has not yet been published. However, this will only happen at later stages. As of now, the prototype is more than what is required and it is quite exceptional.

Future Development Phases:

- Functionality: Adding more functionality based on user feedback
- User Feedback Incorporation: Refining the user interface based on feedback.

8. Annexures

AI Usage Breakdown:

- Splash Screen: Used AI (ChatGPT) to help me understand how to use the progress bar.
- GUI Design Assistance: Used AI (ChatGPT) to help me with the colour scheme and font, as well as fix certain errors such as brackets being in the wrong place.

9. References

- Bell, D., 2023. *The UML 2 class diagram*. [Online]
Available at: <https://developer.ibm.com/articles/the-class-diagram/>
[Accessed 10 September 2024].
- Cote, C., 2021. *What Is Data Integrity and Why Does It Matter?*. [Online]
Available at: <https://online.hbs.edu/blog/post/what-is-data-integrity>
[Accessed 10 September 2024].
- geeksforgeeks, 2024. *Consistency in System Design*. [Online]
Available at: <https://www.geeksforgeeks.org/consistency-in-system-design/>
[Accessed 10 September 2024].
- IBM, n.d. *What is data security?*. [Online]
Available at: <https://www.ibm.com/topics/data-security>
[Accessed 10 September 2024].
- Nixon, C., 2022. *The key to effective storage management: How to identify errors and resolve them before they affect your business*. [Online]
Available at: <https://celerity-uk.com/blog/the-key-to-effective-storage-management-how-to-identify-errors-and-resolve-them-before-they-affect-your-business/>
[Accessed 10 September 2024].
- OpenAI, n.d. *ChatGPT*. [Online]
Available at: <https://chatgpt.com/>
- Schneider, J. & Smalley, I., 2024. *What is transaction management?*. [Online]
Available at: <https://www.ibm.com/topics/transaction-management>
[Accessed 10 September 2024].
- Shaw, K., 2024. *C# CheckBox Control*. [Online]
Available at: <https://www.naukri.com/code360/library/c-sharp-checkbox-control>
[Accessed 10 September 2024].