



ST. PAUL'S UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

PROJECT REPORT

WAMIETECH PORTFOLIO:

A scalable , dynamic content management platform for project showcase and service streamlining

BY

WAMILIMO HILLARY

BSCLMR288922

SUPERVISOR: SELF SUPERVISED

Project Report submitted in partial fulfillment of the requirements for the award of the Degree in
Computer science

11th April, 2025

Contents

INTRODUCTION	1
A case scenario.....	1
Problem statement	1
General objective	1
Specific objectives	1
Justification: cons of a static portfolio supporting the need for a dynamic system.....	2
Targeted audience.....	2
System Features.	4
1. Landing Page key features.....	4
Teams and testimonial.....	7
About Section.....	8
Services	8
Projects.....	9
Authentication Section.....	9
Client panel	10
Admin dashboard	10
Project management.....	11
Responsive Design.....	12
USE OF COMPLEX DATA TYPES AND OOP CONCEPTS.....	13
1. List and array	13
JSONB as a Complex Data Type	13
Abstraction.....	14
Encapsulation	14
Role-Based Access Control (RBAC)	14
Meeting Scheduling and Notification System where Database Triggers are used.....	14
Feature Overview	14
Triggers Implemented	14
Conclusion	16
REFERENCES	16

Figure 1:Landing page	4
Figure 2:Buy me coffee	5
Figure 3:Hire me modal.....	5
Figure 4:Sidebar	6
Figure 5:My team	7
Figure 6:Testimonial	7
Figure 7:About section	8
Figure 8:Services dynamic.....	8
Figure 9:Projects section	9
Figure 10: Authentication page.....	9
Figure 11:Clients dashboard	10
Figure 12:Admin dashboard.....	10
Figure 13:Project management.....	11
Figure 14: editing a user.....	11
Figure 15:Sample deletion	12
Figure 16:Responsiveness	12
Figure 17 Jsonb used in app.	13
Figure 18:project structure	16

INTRODUCTION

A case scenario

In a mixed class of IT and non-IT students, such as computer science and nursing, an update is required on everyone's portfolio within five minutes like adding a new skill learned in class. The IT student may manage, though it's still time-consuming with static code. For the nurse with limited technical skills, her static, hard-coded portfolio means she must either hire a developer or risk breaking the site if she edits it herself. Even the IT student, in a real-world scenario like a job interview, may need to urgently update a service or project but without a dynamic interface, he's stuck. Static portfolios delay updates, risk errors, and don't support income-generating features. And why have a website that uses up your time and money but doesn't bring any return? A dynamic portfolio system solves all this: it allows real-time content updates through a simple UI, and adds tools like bookings, scheduling, and secure payments turning a portfolio from just a showcase into a self-managed, revenue-generating platform for both IT and non-IT users.

Problem statement

Most personal portfolios, especially for non-IT users, are static and difficult to update without technical skills. This leads to delays in updates in critical times like having a client, increased maintenance costs, missed opportunities, and no potential for income generation. There is a need for a dynamic portfolio system that allows easy content updates, supports service management, and enables revenue-generating features.

General objective

To design and develop a dynamic web-based portfolio system that streamlines service management, project showcasing, and user-friendly content updates through an intuitive interface.

Specific objectives

1. To develop a UI-based content management system for dynamic display and efficient updating of services, projects, and profile information.
2. To implement a scheduling feature that enables project coordination and insight-driven interactions with clients.
3. To integrate a booking and financial management module that supports service requests, payments, and transaction tracking.
4. Demonstrate OOP & advance database concepts

Justification: cons of a static portfolio supporting the need for a dynamic system

1. **Limited Flexibility** - Any content change requires manual editing of code, making it difficult to keep the site consistently updated or adaptable to changing needs.
2. **Dependence on Technical Skills** - Non-IT users must rely on developers, which can be both costly and inefficient—especially for simple updates.
3. **Error-Prone Updates** - Direct code modifications increase the risk of mistakes, potentially disrupting the website's functionality, layout, or user experience.
4. **Inability to Integrate Services** - Static sites cannot easily support advanced features such as real-time bookings, client interaction tools, or secure payment gateways.
5. **Poor Scalability** - Static structures lack the flexibility needed to scale with growing content, user interactions, or additional service modules. Manually adding the services listing needs to copy structure which is tiresome. We only need to input and get the service displayed with consistent styles
6. **Missed Revenue Opportunities** - Without dynamic content management and service integration, static portfolios fail to support monetization, client engagement, and timely responsiveness to market demands. Maybe a client was browsing your portfolio services and misses a service he/she wanted then boom you update and he sees. That is a competitive opportunity.

Targeted audience.

Potential Clients - People who are looking for services through the portfolios, whether they need to book a consultation, purchase a service, or interact with the portfolio owner.

Employers - Hiring managers or recruitment agencies looking to assess candidates' skills, projects, and work experience quickly.

Tech Stack Used

1. **PostgreSQL** – for structured, relational database management.
 2. **HTML, CSS, and JavaScript** – for creating a responsive and modern frontend UI.
 3. **Python (Flask Framework)** – for backend logic, routing, and API handling.
 4. **SHA-256 Hashing Algorithm** – for secure password storage.
-

Security Features and Code Quality

1. **Form Validation** – to ensure only clean and expected input reaches the server.
2. **Protection Against SQL Injection** – using separate query files and parameterized SQL queries.
3. **Email Verification** – implemented to prevent fake account creation.
4. **Password Recovery & Reset** – secure token-based mechanism for password resets.
5. **Environment Variables (.env)** – used to safely store sensitive credentials and configuration data.
6. **Code Modularity** – clean, scalable, and maintainable structure using Flask Blueprints and clear separation of concerns.

System Features.

1. Landing Page key features

Features a clean navigation bar and a hero section introducing the admin, with a typewriter effect displaying messages like hey folks my name is wamilimo, rubbed and write another am a software engineer based in Kenya etc. Call-to-action buttons (View CV, Learn More) guide users to key sections.

Buy Me Coffee Feature

A modal form for MPESA donations with fields for MPESA number and amount.

Sidebar

Includes admin details and quick navigation links.

Hire Me Section

Encourages collaboration with direct contact links (Call Me, WhatsApp, Email, etc.) and motivational tagline.

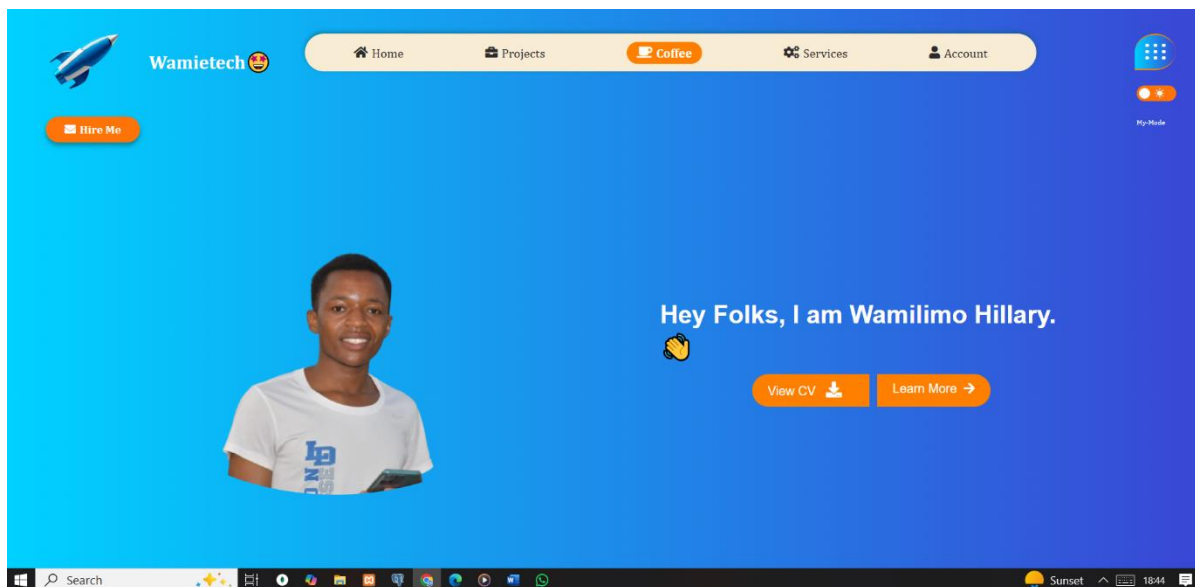


Figure 1:Landing page

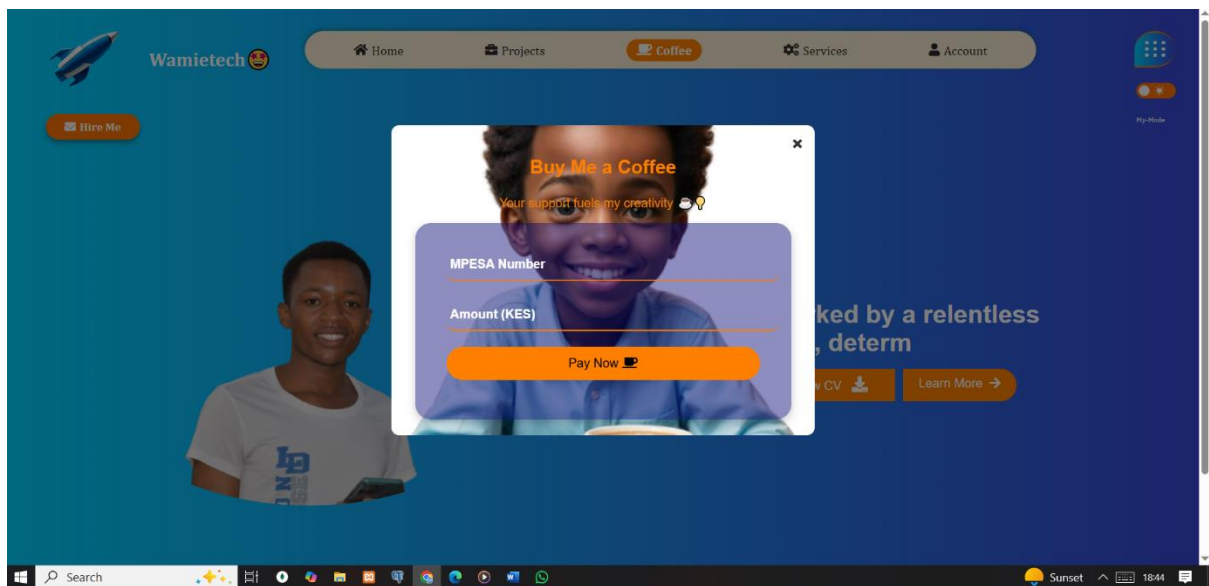


Figure 2:Buy me coffee

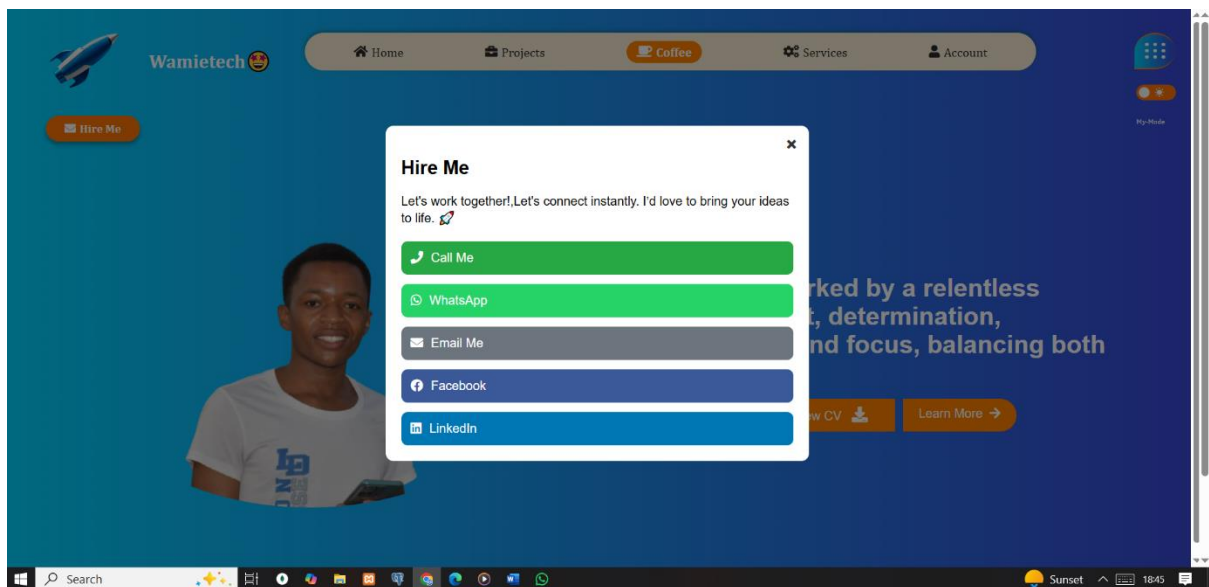


Figure 3:Hire me modal

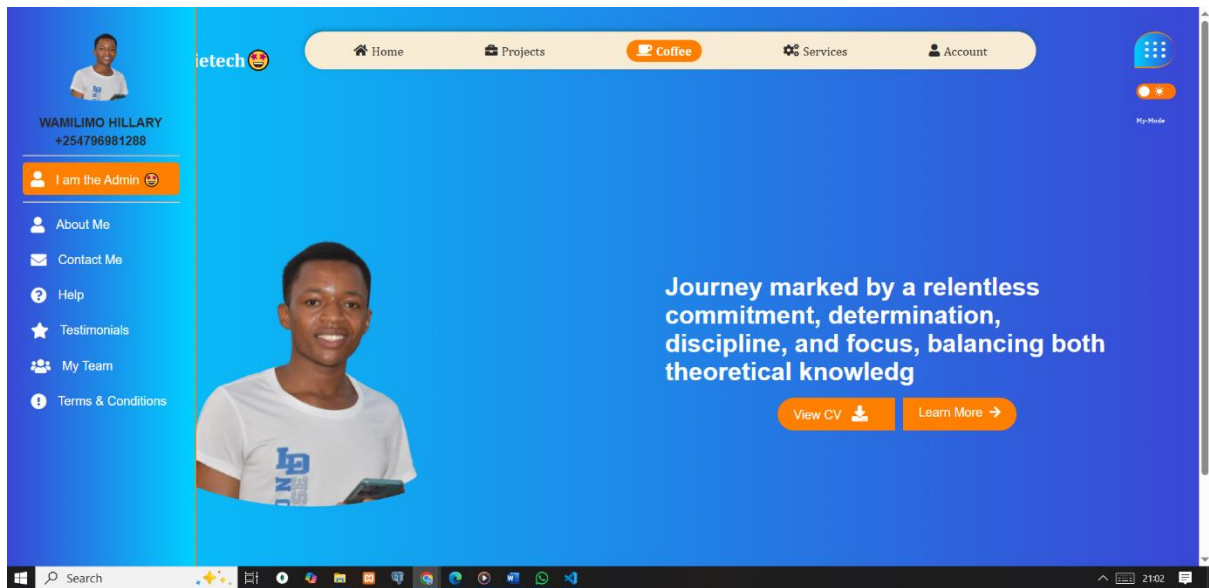


Figure 4:Sidebar

Teams and testimonial

Teams and testimonial images and details respectively are dynamically fetched . when a testimonial or team member added. Its dynamically added to the list. You can click testimonial button and read dynamically fetched testimonial.

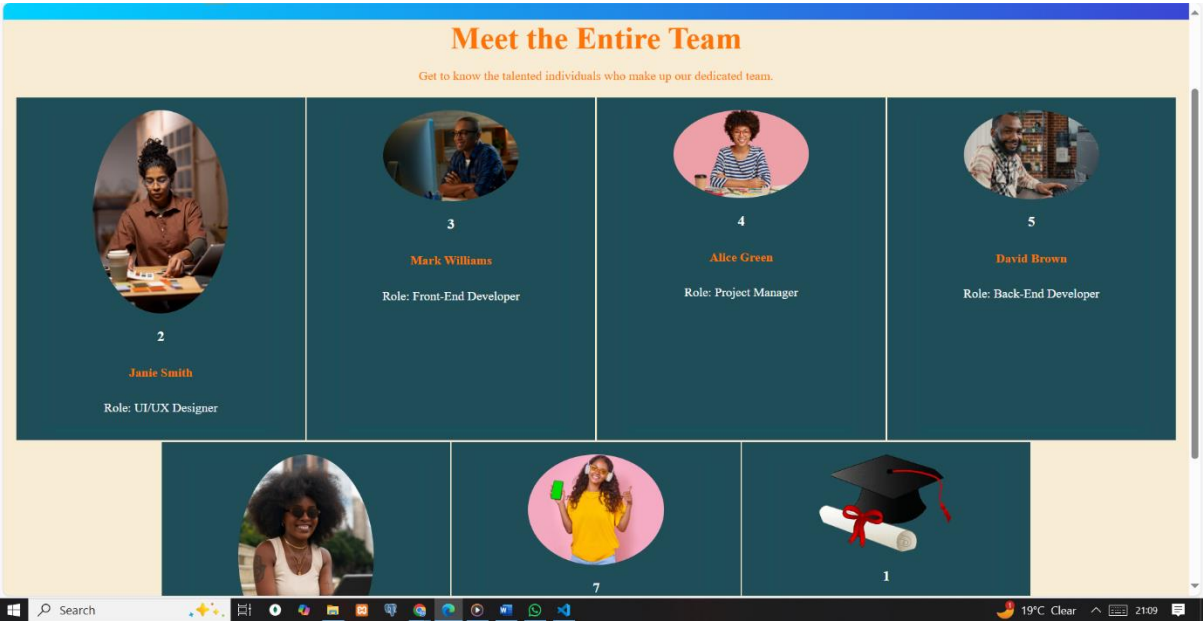


Figure 5:My team

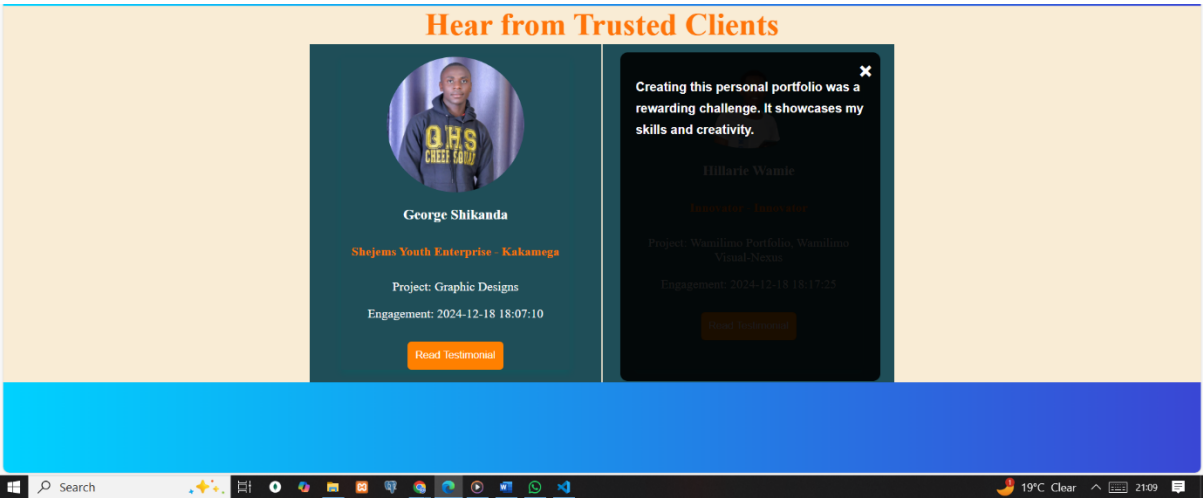


Figure 6:Testimonial

Testimonial and an open testimonial text dynamically fetched.

About Section

The About section provides background information about the user, including their professional journey, skills, and interests. This section helps to personalize the portfolio and gives visitors insight into the user's career and achievements.



Figure 7: About section

Services

The Services section lists the professional services offered by the user. The section is dynamically fetched from the database to ensure that the list stays current and accurate.

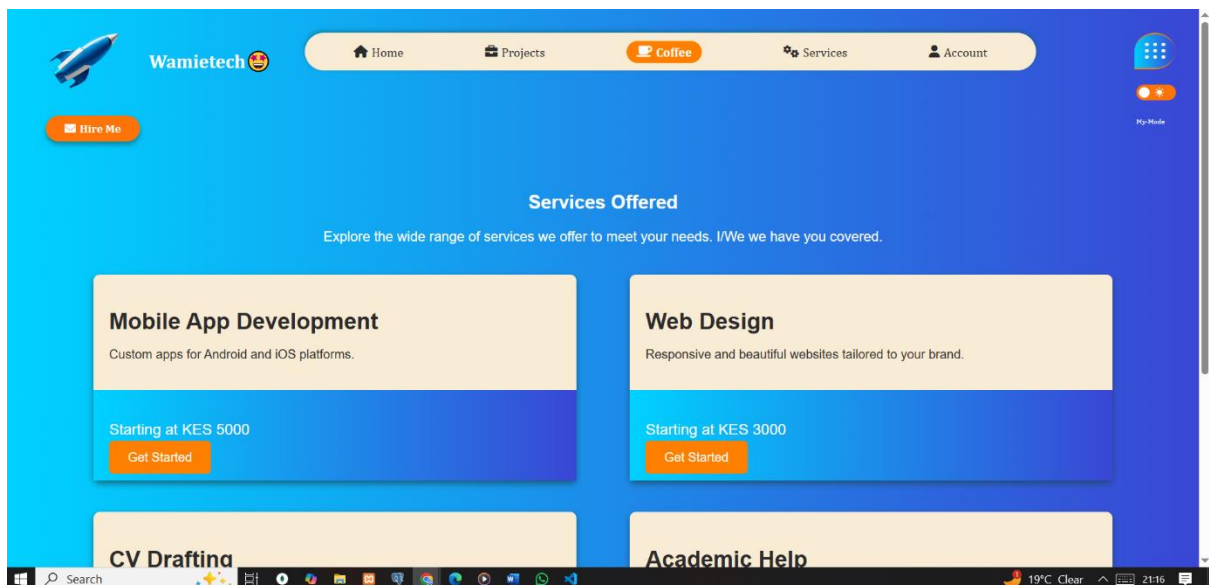
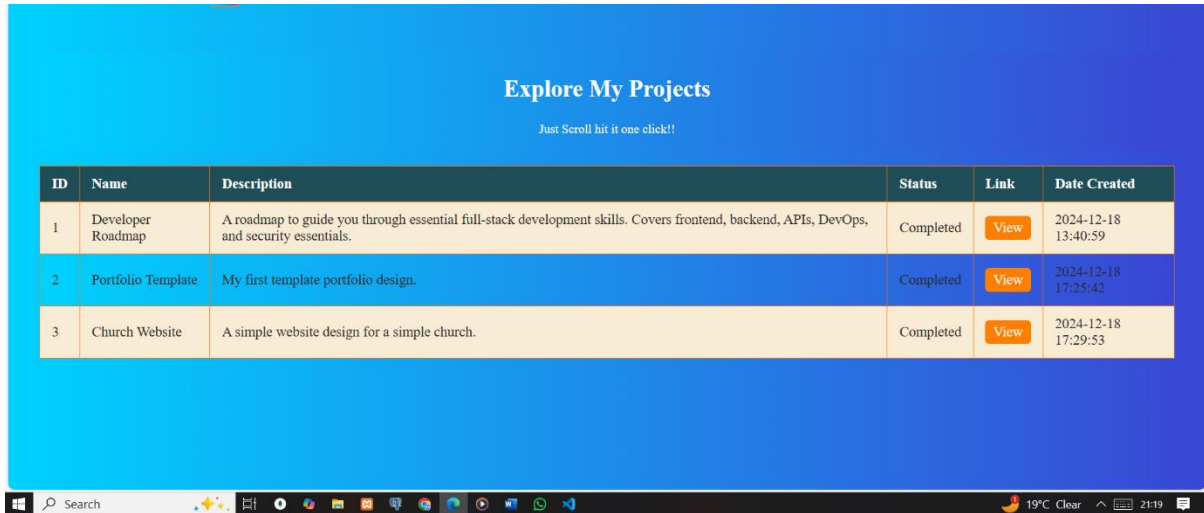


Figure 8: Services dynamic

Projects

This section dynamically displays projects worked on. The section is dynamically populated from the database, allowing the user to update their projects easily.

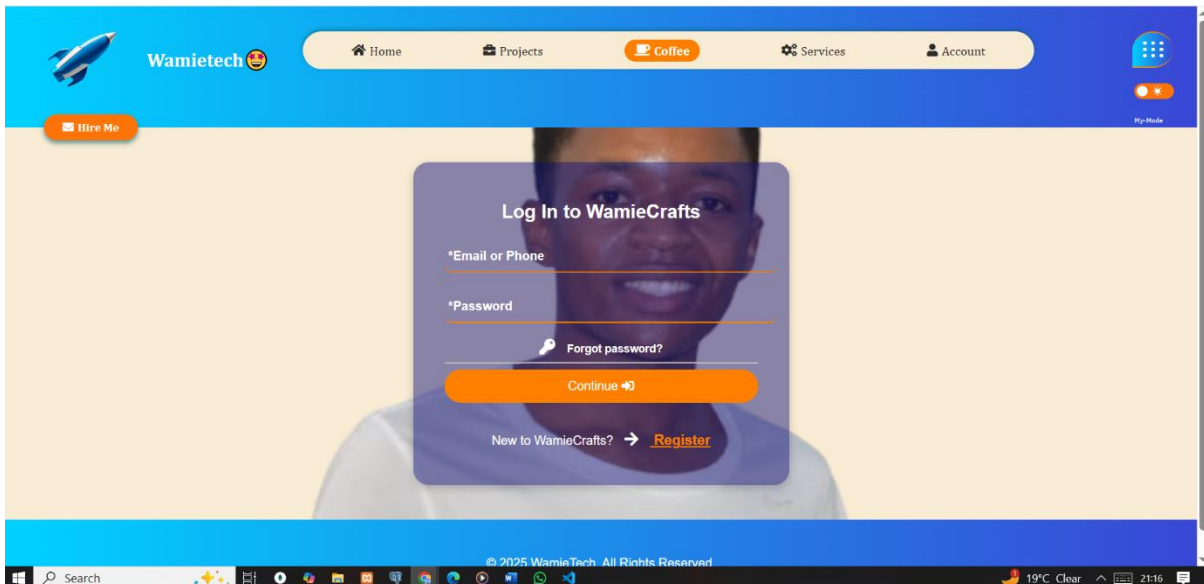


ID	Name	Description	Status	Link	Date Created
1	Developer Roadmap	A roadmap to guide you through essential full-stack development skills. Covers frontend, backend, APIs, DevOps, and security essentials.	Completed	View	2024-12-18 13:40:59
2	Portfolio Template	My first template portfolio design.	Completed	View	2024-12-18 17:25:42
3	Church Website	A simple website design for a simple church.	Completed	View	2024-12-18 17:29:53

Figure 9: Projects section

Authentication Section

Simple and clean login/register interface with input fields for email and password. Includes “Forgot Password” link and feedback messages for invalid credentials



Wamietech

Home Projects Coffee Services Account

Hire Me

Log In to WamieCrafts

*Email or Phone

*Password

[Forgot password?](#)

[Continue](#)

New to WamieCrafts? → [Register](#)

© 2025 WamieTech. All Rights Reserved

Figure 10: Authentication page

Client panel

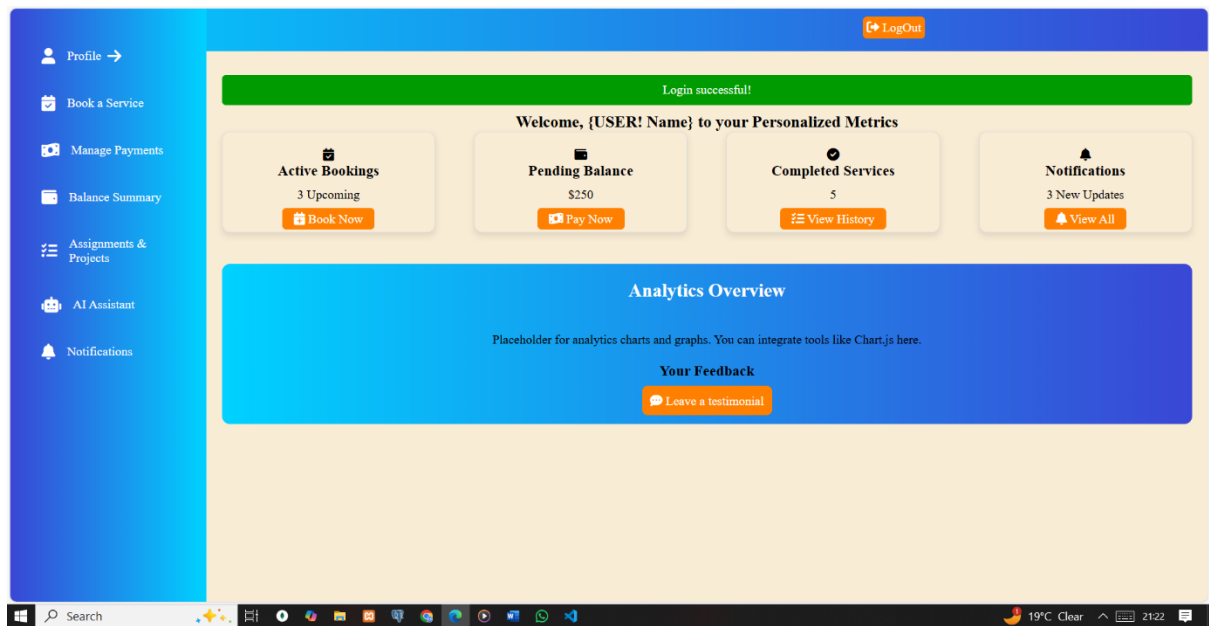


Figure 11: Clients dashboard

Admin dashboard

The Admin Panel allows manage portfolio content easily. From this panel, the user can add, edit, or remove projects, update the services offered, and modify contact details. The admin interface is designed to be intuitive and efficient, making portfolio maintenance simple.

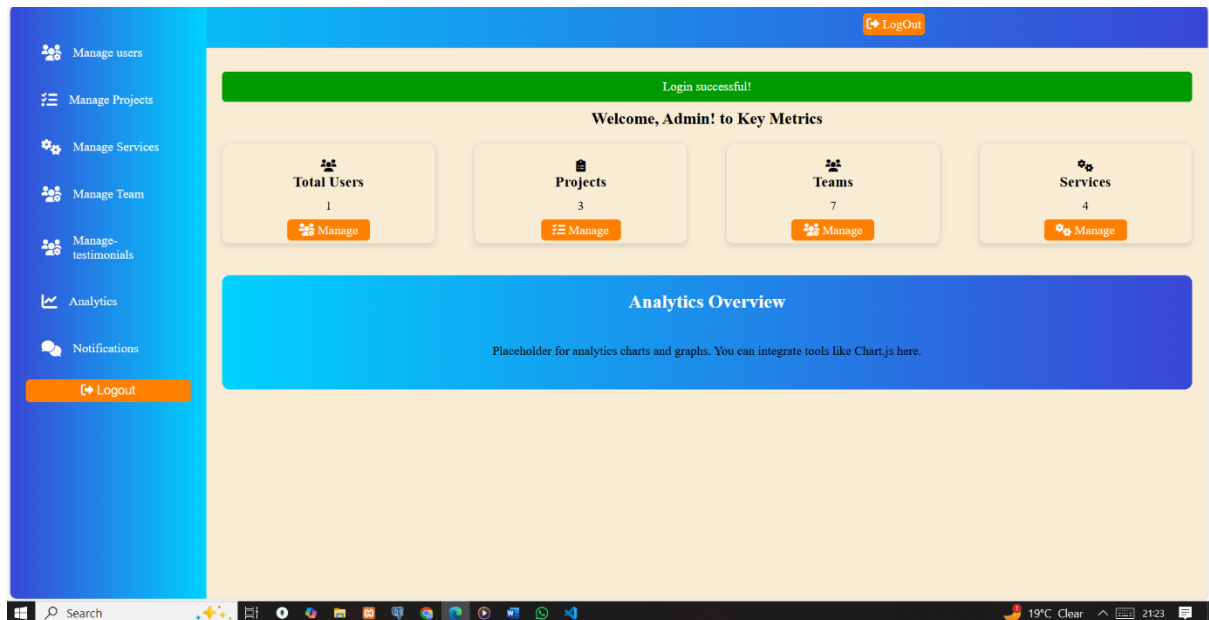
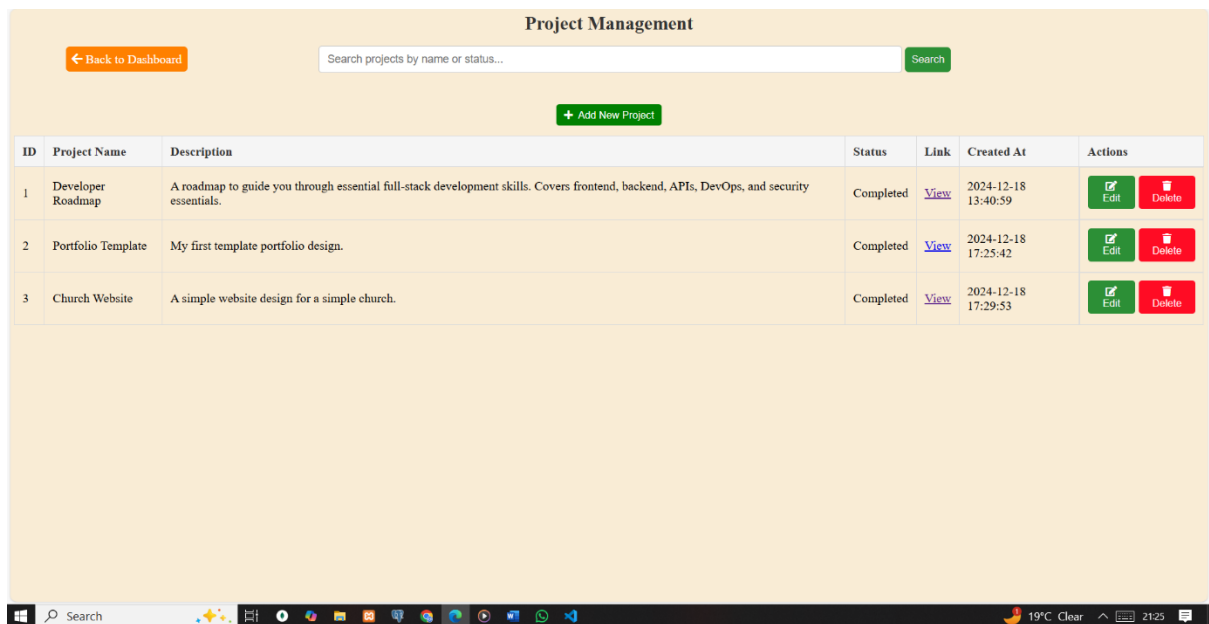


Figure 12: Admin dashboard.



Project management

Figure 13: Project management

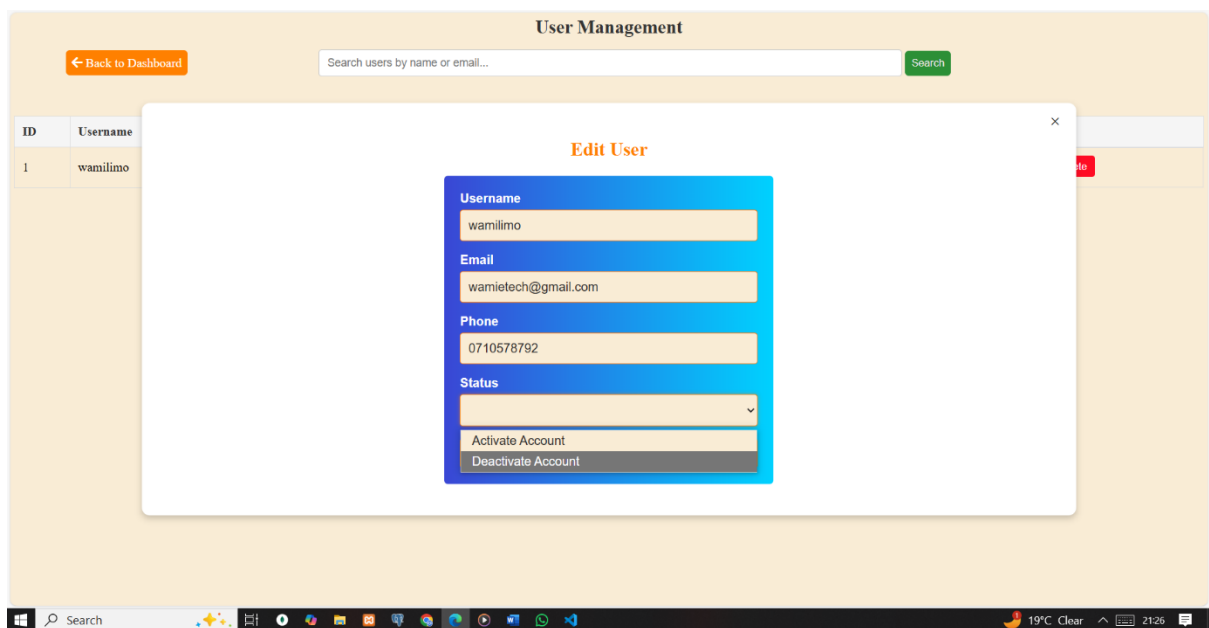


Figure 14: editing a user

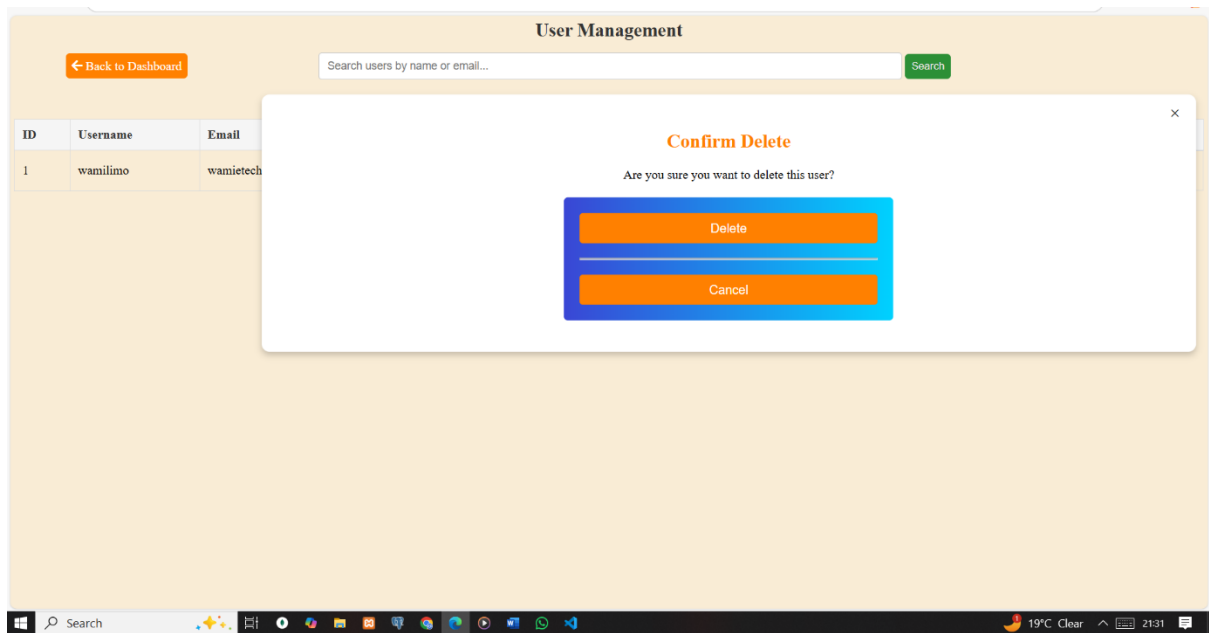


Figure 15: Sample deletion

Responsive Design

The portfolio is fully responsive, ensuring it is optimized for various screen sizes, from desktops to mobile devices. The layout adapts dynamically to ensure a smooth user experience across different devices, enhancing accessibility and usability.

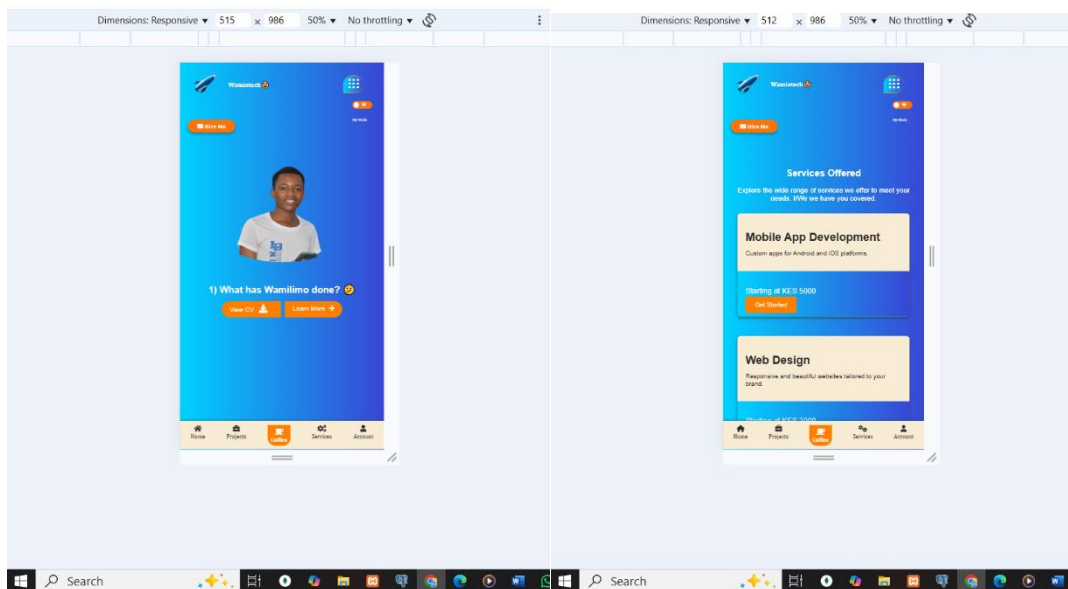


Figure 16: Responsiveness

USE OF COMPLEX DATA TYPES AND OOP CONCEPTS

1. List and array

The application retrieves typewriter messages from the database as individual TEXT rows and converts them into a LIST complex data type. This list is then passed to the frontend as a JavaScript array, enabling dynamic rendering in the typewriter effect. This approach combines ease of admin editing with effective use of complex data types.

JSONB as a Complex Data Type

To demonstrate PostgreSQL's support for complex data types, JSONB was used to store multiple referee entries in a single column. This enabled flexible and efficient querying without relying on foreign keys. Sample queries included extracting entire JSON arrays as well as specific fields like email addresses from nested objects.

The screenshot displays a PostgreSQL database interface with two panels. The left panel shows SQL queries for creating, inserting, and querying a table named 'referees_section'. The right panel shows the results of a query that extracts specific fields from the JSONB data.

Query 1 (Left Panel):

```
1 CREATE TABLE referees_section (  
2   id SERIAL PRIMARY KEY,  
3   title VARCHAR(100),  
4   referees JSONB  
5 );  
6  
7 DROP TABLE referees_section  
8  
9 INSERT INTO referees_section (title, referees) VALUES (  
10  'Main Referees',  
11  '[  
12    {  
13      "full_name": "Lucy ndungu",  
14      "position": "Lecturer",  
15      "organization": "SPU University",  
16      "email": "Indungu@spu.ac.ke",  
17      "phone": "+1234567890",  
18      "relationship": "Supervisor"  
19    },  
20    {  
21      "full_name": "ziroh kitsao",  
22      "position": "Mentor",  
23      "organization": "TanadaaTelecommunicationCompany",  
24      "email": "ziroh@gmail.com",  
25      "phone": "+9876543210",  
26      "relationship": "Project Lead"  
27    }  
28  ]',  
29 );  
30  
31
```

Query 2 (Right Panel):

```
38  
39 SELECT  
40   elem->>'full_name' AS full_name,  
41   elem->>'position' AS position,  
42   elem->>'organization' AS organization,  
43   elem->>'email' AS email,  
44   elem->>'phone' AS phone,  
45   elem->>'relationship' AS relationship  
46 FROM referees_section,  
47      jsonb_array_elements(referees) AS elem;  
48  
49
```

Data Output (Right Panel):

full_name	position	organization	email	phone	relationship
Lucy ndun.	Lecturer	SPU University	Indungu@spu.ac.ke	+1234567890	Supervisor
ziroh kitsao	Mentor	TanadaaTelecommunicationCompany	ziroh@gmail.com	+9876543210	Project Lead

Figure 17 Jsonb used in app.

Abstraction

Typewriter Messages

The frontend displays typewriter messages without knowing how they are retrieved or stored in the database. The complex logic of fetching and formatting data is hidden behind a simple function.

MPESA Payment

The payment form allows users to donate without exposing the underlying API communication and payment processing logic. The backend handles all complexity, presenting a simple interface to the user.

Project and Service Updates

Admins update content through simple forms (like "Edit user"). The forms interact with backend logic for validation and database updates without the admin needing to understand the underlying processes.

Encapsulation

During **payment for services**, the user's balance is encapsulated and can only be accessed or updated through specific methods `pay()`. The balance cannot be directly modified, ensuring that all updates are handled securely and accurately, maintaining data integrity.

Role-Based Access Control (RBAC)

WamieTech Portfolio uses **Role-Based Access Control (RBAC)** with two roles: **Admin** and **User**. Admins can manage and modify all content, while Users can only view the portfolio and services. This ensures secure and controlled access to the system.

Meeting Scheduling and Notification System where Database Triggers are used.

Feature Overview

The system integrates **Google Meet-based scheduling** to streamline communication and ensure structured, conflict-free meetings between clients and admins. Key triggers were implemented to automate meeting management and notifications.

Triggers Implemented

1. Auto-set Meeting End Time

Trigger - Automatically calculates and sets the meeting **end_time** to **1 hour** after the **start_time** when a meeting is scheduled.

Why I Did This - This ensures that all meetings have a fixed duration, avoiding overruns and ensuring a structured meeting flow.

2. Meeting Status Update

Trigger - Automatically updates the meeting status to **‘completed’** exactly **1 hour** after the **start_time**.

Why I Did This - This helps in marking the meeting as completed without requiring manual updates. It ensures the admin is alerted when a meeting has finished and can proceed with the next one on time.

3. Automated Meeting Notifications

Trigger: Sends automated notifications at specific times:

- ❖ **10 minutes before the meeting** starts.
- ❖ **5 minutes before the meeting** ends.
- ❖ **Upon meeting completion**.

Why I Did This - This automation ensures that all participants receive timely reminders and updates about their meetings, improving the overall user experience and keeping everyone informed. It also ensures that admins are aware of meeting completions, helping to manage transitions smoothly.

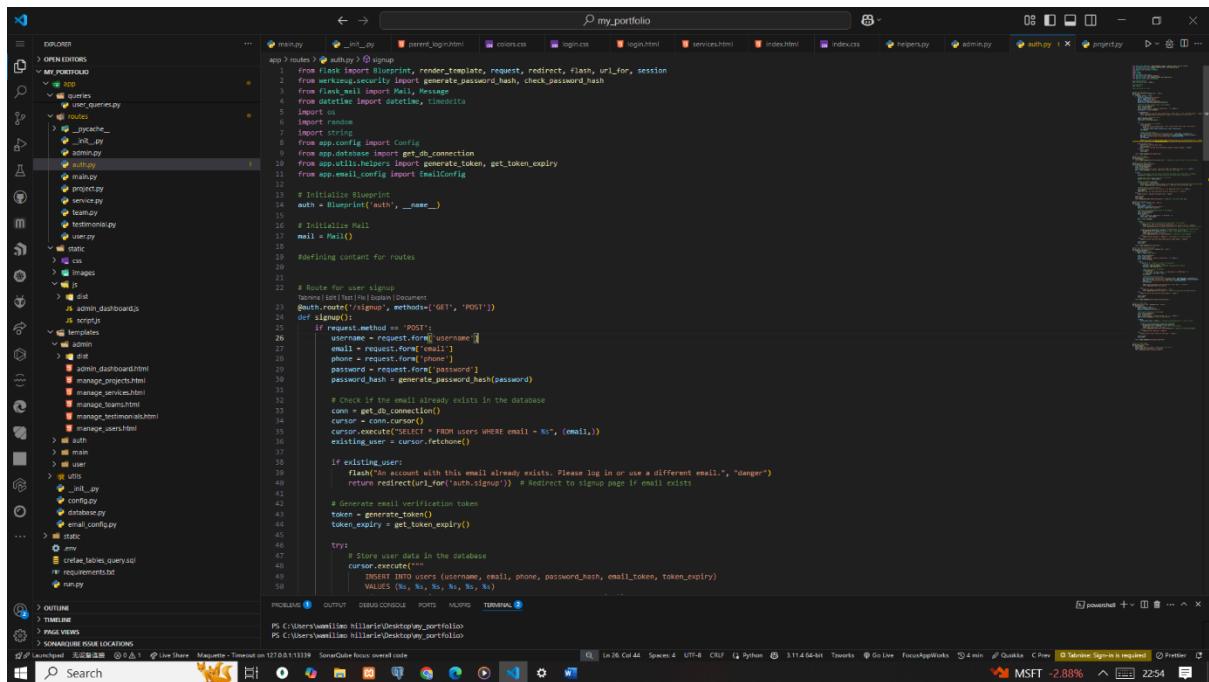


Figure 18:project structure

Conclusion

The portfolio system offers a dynamic platform for showcasing projects and accessing services efficiently. It features role-based access, dynamic content updates, and uses complex data types like jsonb to store referee details independently. Admins manage all content with full CRUD rights, while users enjoy a personalized, simplified interface.

Service management is streamlined, allowing admins to update offerings and pricing dynamically. Automated triggers enhance features like meeting scheduling—ensuring fixed durations, real-time notifications, and status updates—making the system modern, responsive, and well-structured.

REFERENCES

Wamilimo, H. (n.d.). *WamieTech Portfolio* [Source code]. GitHub.
<https://github.com/wamilimohillary/wamietech-portfolio>