PUBLIC PORTFOLIO

A DESIGN PROJECT REPORT

submitted by

KEERTHIVASAN S J

LOGESHWARAN P J

MOHANA RENGAN N

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)

Samayapuram – 621 112

DECEMBER, 2024

PUBLIC PORTFOLIO

A DESIGN PROJECT REPORT

submitted by

KEERTHIVSAN S J (811722104074)

LOGESHWARAN P J (811722104082)

MOHANA RENGAN N (811722104090)

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)

Samayapuram – 621 112

DECEMBER, 2024

K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled **"PUBLIC PORTFOLIO"** is bonafide work of **KEERTHIVASAN S J (811722104074), LOGESHWARAN P J (811722104082), MOHANA RENGAN N (811722104090)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
Dr. A Delphin Carolina Rani M.E., Ph.D.,
HEAD OF THE DEPARTMENT
Professor
Department of CSE
K Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

SIGNATURE
Mr. A. Malarmannan, M.E.,
SUPERVISOR
Assistant Professor
Department of CSE
K Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

Submitted for the viva-voice examination held on ……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We jointly declare that the project report on **"PUBLIC PORTFOLIO"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of Bachelor Of Engineering. This project report is submitted on the partial fulfilment of the requirement of the awardof Degree of Bachelor Of Engineering.

Signature

KEERTHIVASAN S J

LOGESHWARAN P J

MOHANA RENGAN N

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and indebtness to our institution **"K RAMAKRISHNAN COLLEGE OF TECHNOLOGY"**, for providing us with the opportunity to do this project.

We are glad to credit honorable chairman **Dr. K RAMAKRISHNAN, B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering adequate duration to complete it.

We would like to thank **Dr. N VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project with full satisfaction.

We whole heartily thank **Dr. A DELPHIN CAROLINA RANI, M.E., Ph.D.,** Head of the Department, **COMPUTER SCIENCE AND ENGINEERING** for providing her support to pursue this project.

We express our deep and sincere gratitude and thanks to our project guide **Mr. A MALARMANNAN, M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course. We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

Public Portfolio Management System is a web-based platform designed to streamline how individuals manage and showcase professional profiles. In the digital era, a dynamic and well-structured portfolio is vital for career growth and personal branding. This system allows users to create, update, and share portfolios effortlessly, highlighting skills, education, certifications, work experience, and project achievements. With real-time customization, users can maintain portfolios that reflect their evolving journeys. The platform supports collaborative input, enabling endorsements, feedback, and recommendations to enhance credibility. Public accessibility ensures users can share profiles with employers, collaborators, and peers. Built with a robust architecture, the system uses frameworks like Django or Flask for efficient backend processing, responsive web design for user-friendly interfaces, and secure databases for reliable data storage. Future features include profile analytics and customizable layouts, aiming to redefine professional networking by empowering users to enhance their digital presence and seize new opportunities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	FULL FORM
CRUD	Create, Read, Update, Delete
ORM	Object-Relational Mapping
MVT	Model-View-Template
UI	User Interface
DBMS	Database Management System
CSRF	Cross-Site Request Forgery
URLconf	URL Configuration

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

The concept of creating a web-based portfolio management system originates from the growing need for individuals to showcase their professional profiles dynamically. Traditional resume-building methods often lack flexibility and fail to leverage modern technology for user-friendly interfaces. Over time, advancements in web technologies and frameworks like Django have paved the way for highly customizable, scalable, and interactive platforms. The Public Portfolio Management System leverages Django (a Python-based web framework) to provide a comprehensive, open-source solution for individuals to create, update, and share their professional portfolios effortlessly. Unlike conventional methods, this system integrates modern design principles with features like ATS (Applicant Tracking System)-friendly resume templates and enhanced accessibility. It addresses the demands of a digital-first era by enabling public viewing and encouraging collaboration.

With the rise of digital employment platforms and job-hunting tools, this project focuses on unifying professional representation and job-seeking capabilities in one platform. The open-source nature ensures community-driven improvements, fostering innovation and inclusivity.

1.2 OVERVIEW

The Public Portfolio Management System is an open-source, web-based application designed to facilitate the creation and sharing of professional profiles. This system allows users to:

- Add key information such as education, work experience, skills, and certifications.
- Generate ATS-friendly resume templates suitable for modern recruitment systems.
- Update their portfolios dynamically, ensuring relevance and accuracy.
- Enable public access to profiles, making it easier for potential employers and collaborators to explore them.

Built on Django, the system is robust, scalable, and secure. Users interact with a simple yet powerful interface that ensures smooth navigation and ease of use.

The table below outlines some of the primary features

Feature	Description	Purpose
Dynamic Portfolio Updates	Allows users to edit and update their profiles in real-time.	Keeps information up-to-date.
ATS-friendly Templates	Provides resume templates optimized for Applicant Tracking Systems.	Enhances job application success.
Public Profile Sharing	Offers public URLs for easy sharing of portfolios.	Simplifies professional networking.
Django Framework	Utilizes the Django framework for secure, scalable, and efficient development.	Ensures reliability and performance.

Table 1.1 Primary Features

1.3 PROBLEM STATEMENT

The challenges faced by job seekers often stem from traditional methods of creating and managing resumes and portfolios. Key issues include:

- Lack of standardized and ATS-compatible resume formats, leading to rejection in digital recruitment processes.
- Limited accessibility to professional portfolio management systems for individuals without technical expertise.
- Absence of a centralized, public platform to showcase skills and qualifications effectively.

The Public Portfolio Management System addresses these issues by providing an open-source, user-friendly platform that bridges the gap between job seekers and recruiters.

1.4 OBJECTIVE

The primary objective of the project is to develop an innovative platform that simplifies the process of creating, managing, and sharing professional portfolios. Key goals include:

- Providing a unique platform for job seekers to enhance their visibility in the competitive job market.

- Offering ATS-friendly resume templates that align with modern recruitment standards.
- Encouraging collaboration and feedback by enabling public access to profiles.
- Leveraging the open-source community for continuous improvement and innovation.
- Minimizing costs by using open-source technologies and eliminating proprietary dependencies.

1.5 IMPLICATION

The implementation of the Public Portfolio Management System has far-reaching implications for job seekers and recruiters alike:

1. **Enhanced Accessibility:** Open-source nature ensures inclusivity for users across demographics.
2. **Efficiency in Recruitment:** ATS-friendly resumes streamline the hiring process for employers.
3. **Global Reach:** Publicly accessible profiles allow individuals to connect with opportunities worldwide.
4. **Collaboration and Feedback:** Public access encourages peer reviews and collaborative updates.
5. **Future-proof Design:** Scalable architecture ensures the platform evolves with technological advancements.

Example Use Case:

A software developer creates a portfolio on the system, adding their skills, certifications, and project experience. Using the built-in tools, they generate an ATS-friendly resume to apply for jobs. Simultaneously, their public profile attracts recruiters, who can view their qualifications directly.

The table below illustrates the benefits for stakeholders:

Stakeholder	**Benefit**
Job Seekers	Easy-to-use platform, ATS-friendly resumes, public visibility.
Recruiters	Access to standardized, well-organized profiles.
Open-source Community	Opportunity to contribute to the project, enhancing its functionality and reach.

Table 1.2 Benefits for Stakeholders

CHAPTER 2

LITERATURE SURVEY

2.1 Job-Seeker Platforms with Open-Source Collaboration

AUTHORS: Lin, Chen, and Zhang

YEAR: 2023

This research focuses on creating platforms that empower job seekers through open-source tools. The authors present a Django-based system for building portfolios and resumes, emphasizing public accessibility and collaboration. The study introduces features such as job application tracking, skill validation, and public portfolio sharing. The authors discuss the role of the open-source community in enhancing platform capabilities. Challenges related to data integrity and scalability are addressed by leveraging cloud-based storage and Django's ORM (Object-Relational Mapping).

2.2 Dynamic Web-Based Portfolio System

AUTHORS: Sharma, Gupta, and Rao

YEAR: 2022

This paper emphasizes the need for dynamic web-based systems for managing and showcasing professional portfolios. The authors propose a system that allows users to build and customize their portfolios online. The study focuses on creating a modular, scalable architecture using modern web technologies like Django. Key features include real-time updates, public access to profiles, and integration with third-party APIs for skills verification. The system enhances user experience by offering intuitive interfaces and streamlining portfolio sharing. Challenges such as data security and user authentication are also discussed, with proposed solutions leveraging Django's in-built security mechanisms.

2.3 Human-Centric Portfolio Systems

AUTHORS: Ahmed and Khan

YEAR: 2021

This paper explores the development of portfolio management systems that prioritize user needs. By conducting surveys and usability testing, the authors identify critical features desired by job seekers, such as easy editing, multimedia integration, and public access. The study introduces a system built on Django, which enables users to create personalized portfolios with a focus on simplicity and accessibility. The system also includes modules for exporting ATS-friendly resumes. The paper concludes with recommendations for making the platform more inclusive, such as integrating language support and accessibility features for individuals with disabilities.

2.4 Modern Approaches to Resume Building

AUTHORS: Davis and Lee

YEAR: 2020

This paper discusses the evolution of resume-building tools, focusing on user-friendly design and ATS compliance. The authors propose a system that automates the resume creation process while maintaining a high degree of customization. The tool, built using Django, offers pre-designed templates optimized for different industries. The study highlights the role of community feedback in improving the platform's usability. Advanced features, such as machine learning-based skill suggestions, are also explored.

2.5 Open-Source ATS-Friendly Resume Generator

AUTHORS: Patel, Mehta, and Desai

YEAR: 2020

This study investigates the importance of ATS-friendly resume templates in modern recruitment processes. The authors developed an open-source resume builder that supports multiple ATS-optimized formats. The paper highlights the significance of creating resumes that can bypass automated filters used by hiring systems. Using Python and Django, the proposed tool ensures flexibility, scalability, and user-friendliness. The system offers pre-built templates, allowing users to tailor their resumes to specific job roles. The research also analyzes feedback from recruiters, underscoring the positive impact of ATS compliance on hiring outcomes.

2.6 Collaborative Portfolio Platforms

AUTHORS: Wilson and Smith

YEAR: 2019

This research paper introduces a collaborative platform where professionals can build and share portfolios. It addresses the challenge of portfolio customization by enabling real-time collaboration between users and peers. Built using open-source frameworks, the platform allows community-driven enhancements. The authors focus on the integration of public visibility features, allowing users to share profiles with potential employers seamlessly. Security concerns, such as unauthorized access and data breaches, are addressed using encryption and access control policies. The findings reveal significant improvements in user engagement and professional networking.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Several platforms currently exist for managing professional portfolios, each with unique features. However, they often have limitations, such as restricted access, paid services, or lack of ATS-friendly templates.

Below are some examples of existing solutions and their challenges:

Existing Systems	Features	Limitations
LinkedIn	Online professional networking with portfolio features.	Limited resume customization and no ATS optimization for downloaded resumes.
Canva	Design tool for creating portfolios and resumes.	Requires paid subscription for advanced features; lacks public access and sharing options.
Wix Portfolio	Website builder with customizable portfolio templates.	Limited ATS compliance; requires technical skills for customization.
Novoresume	Offers ATS-friendly resume templates.	Restricted to resumes only; no dynamic profile management or public viewing.
GitHub Pages	Hosting platform for developer portfolios.	Tailored to developers; lacks features for other professions.

Table 3.1 Existing System

3.2 PROPOSED SYSTEM

The Public Portfolio Management System is designed as an open-source, web-based application to overcome the limitations of existing systems. This platform provides:

1. Dynamic Portfolio Management: Users can create, update, and manage their portfolios in

real time.

2. ATS-Friendly Resume Templates: Pre-designed templates optimized for Applicant Tracking Systems ensure higher compatibility with recruitment tools.

3. Public Profile Access: Users can share a public URL for easy access by recruiters or collaborators.

4. Customizability and Scalability: Built on Django, the system offers robust and scalable solutions.

5. Open-Source Collaboration: Encourages continuous improvement by allowing contribution developer community.

3.3 CLASS DIAGRAM OF PROPOSED SYSTEM

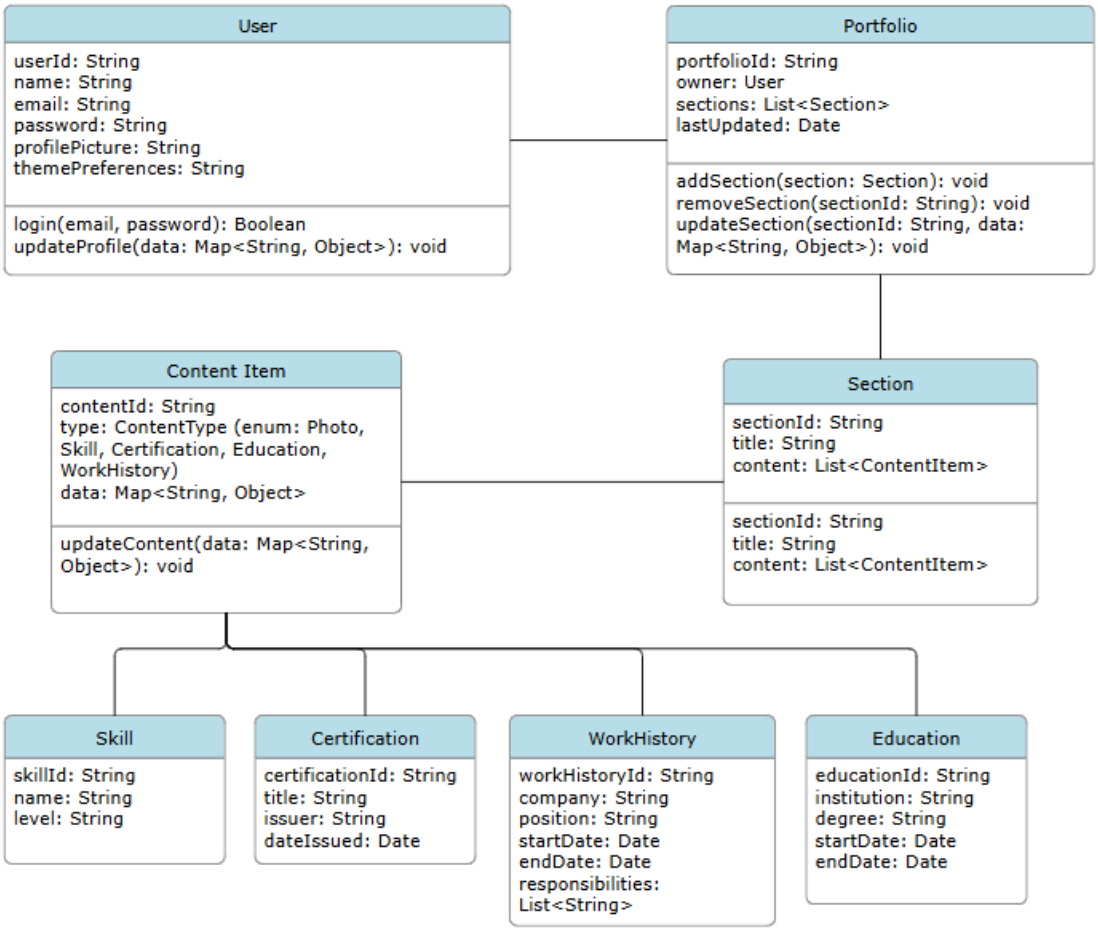

Figure 3.1 Class Diagram

3.4 FLOWCHART

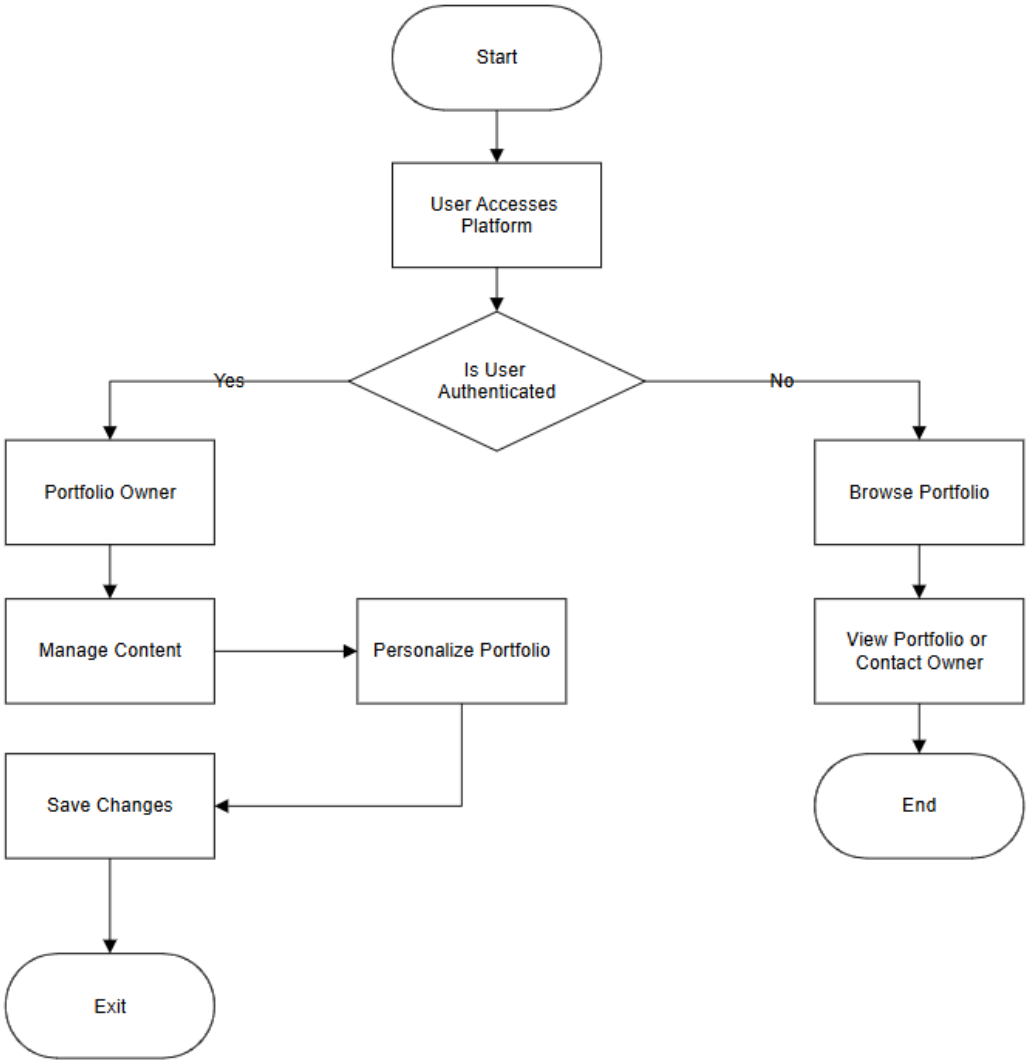

Figure 3.2 Flowchart Diagram

3.5 PROCESS CYCLE

1. User Registration/Login: The user registers or logs into the system.
2. Profile Creation: Users input personal details, skills, education, and work experience.
3. Resume Generation: ATS-friendly resumes are generated based on the input.
4. Profile Sharing: Public URLs are created for easy sharing.
5. Feedback and Updates: Users receive feedback from collaborators and update their profiles.

3.6 ACTIVITY DIAGRAM

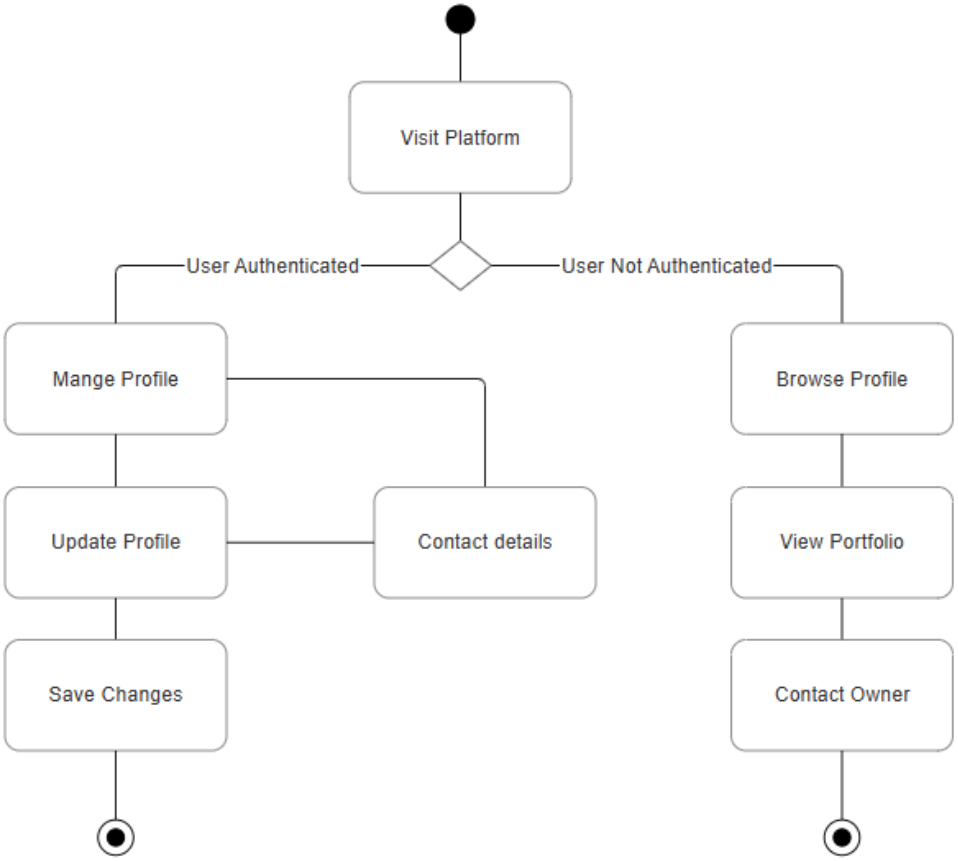

Figure 3.3 Activity Diagram

CHAPTER 4

MODULES

4.1 USER AUTHENTICATION MODULE

This module handles Login and Sign-Up functionalities for users. It ensures secure access to the platform through authentication mechanisms.

Features:

1. User Registration:

New users can create an account by providing their email, username, and password.

Email verification is implemented to prevent unauthorized access.

2. Login:

Existing users can log in using their registered credentials.

Multi-factor authentication can be added for enhanced security.

3. Password Recovery:

Provides an option for users to reset their password through email in case of forgotten credentials.

4. Session Management:

Ensures active sessions are tracked and logged out automatically after inactivity.

5. Benefits:

Protects user data through encryption and secure protocols.

Simplifies account management with a user-friendly interface.

4.2 HOME PAGE MODULE

The Home Page serves as the central hub for users, providing a comprehensive overview of their portfolio.

Features:

1. Profile Overview:

Displays personal details such as name, contact information, and profile picture.

2. Dynamic Sections:

Users can edit, update, and organize the following sections:

Experience: Work history with descriptions of roles and responsibilities.

Education: Academic background, degrees, and certifications.

Projects: Showcase of completed and ongoing projects.

Skills: Technical and soft skills with proficiency levels.

Languages Known: List of languages with proficiency levels.

Certificates: Uploaded certifications and achievements.

Benefits: Keeps all portfolio details centralized and easily accessible.

Allows real-time updates to maintain accuracy and relevance.

4.3 PUBLIC PROFILE MODULE

Features:

1. Customizable URL:

Users can generate a unique and professional public link for their profile.

2. Privacy Settings:

Controls to manage which sections of the profile are visible to the public.

3. Responsive Design:

Ensures the public profile is viewable on various devices, including desktops, tablets, and smartphones.

Benefits:

Enhances professional visibility and networking opportunities.

Simplifies the process of sharing resumes and portfolios with recruiters.

4.4 ATS-FRIENDLY RESUME GENERATOR MODULE

This module focuses on creating professional resumes that comply with Applicant Tracking Systems (ATS).

Features:

1. Pre-designed Templates:

Offers multiple ATS-friendly templates tailored for different industries.

2. One-Click Export:

Generates resumes in PDF format, ready for applications.

3. Customization Options:

Users can rearrange sections and adjust formatting to fit specific job requirements.

Benefits:

Increases the chances of passing ATS scans during job applications.

Saves time and effort in creating professional resumes.

4.5 EDIT AND UPDATE MODULE

This module allows users to edit and update their profile information dynamically.

Features:

1. Inline Editing:

Users can edit information directly within the profile sections without navigating to separate pages.

2. Validation Checks:

Ensures that all entries are formatted correctly and meet predefined criteria.

3. Version Control:

Tracks changes and provides an option to revert to previous versions of the profile.

Benefits:

Encourages continuous improvement and keeps profiles relevant.

Reduces the effort needed for major updates or corrections.

4.6 ADMINISTRATION MODULE

This module is designed for system administrators to manage users and maintain the platform.

Features:

1. User Management:

View, modify, or deactivate user accounts as needed.

2. Content Moderation:

Monitor and approve public profiles to maintain platform quality.

3. Analytics and Reporting:

Provides insights into user activity, popular templates, and public profile traffic.

Benefits:

Ensures the smooth operation of the platform.

Helps administrators make data-driven decisions for future updates.

4.7 FEEDBACK AND COLLABORATION MODULE

This module encourages user interaction and feedback, fostering collaboration and improvement.

Features:

1. Feedback Forms:

Allows users to provide feedback on templates, features, and overall user experience.

2. Collaboration Tools:

Facilitates input from mentors, peers, and recruiters to improve profiles.

3. Community Contributions:

Open-source nature allows developers to suggest and implement new features.

Benefits:

Enhances the platform's functionality based on real user input.

Creates a community-driven approach to portfolio management.

CHAPTER 5

SYSTEM SPECIFICATIONS

5.1 Hardware Requirements

5.1.1 Desktop/Laptop:

Processor: A dual-core CPU such as Intel Core i3 or an equivalent processor. This ensures the platform operates smoothly and efficiently for both developers and users.

RAM: A minimum of 4 GB is recommended to handle the Django-based web application, including multi-tasking for database queries, file uploads, and real-time interactions.

Storage: At least 50 GB of free disk space is required to store application files, user data, backups, and logs.

Display: The platform is designed to be visually optimized for displays with a resolution of 1280x800 pixels or higher, ensuring a seamless user interface experience.

5.1.2 Mobile Devices:

RAM: Devices with a minimum of 2 GB RAM can easily access the application through web browsers, ensuring compatibility across low- to high-end devices.

Operating System (OS): The system supports the latest versions of iOS and Android, providing broad compatibility for users across different ecosystems.

Display: A 720p resolution ensures clear visualization of profiles, templates, and other graphical elements.

5.2 Software Requirements

The system is browser-based, making it accessible across multiple platforms without needing additional installations. Supported browsers include:

Google Chrome: For its speed and compatibility with modern web technologies.

Mozilla Firefox: For its robust performance and enhanced privacy features.

Microsoft Edge: Ensures compatibility with Windows devices and integrates seamlessly with other Microsoft services.

Safari (Mac OS): Optimized for Mac OS users, ensuring fluid interactions.

5.2.1 Internet Connection:

A stable internet connection is mandatory for accessing, updating, and sharing portfolios. Features like real-time data synchronization and public access require uninterrupted connectivity.

5.2.2 Development Tools:

Code Editor: Tools like Visual Studio Code, Sublime Text, or PyCharm are ideal for customizing the system's Django-Python codebase. They provide syntax highlighting, debugging features, and Git integration to streamline development.

Django Framework: The platform relies on Django's robust features for rapid development, scalability, and security.

Database: The system can integrate with databases like SQLite (for local testing) and PostgreSQL or MySQL (for deployment).

5.3 Additional Requirements

5.3.1 Backend Environment:

Python 3.8 or higher is required to run Django and associated libraries efficiently.

Virtual environment tools like `venv` or `virtualenv` are used to manage dependencies and isolate the development environment.

5.3.2 Frontend Technologies:

HTML5 and CSS3 for responsive design and smooth user interfaces.

JavaScript frameworks (e.g., jQuery or React) to enhance interactivity on profile pages.

5.3.3 Deployment Environment:

A cloud hosting service like AWS, Heroku, or DigitalOcean for deploying the web application, ensuring high availability and scalability.

5.3.4 Security Measures:

SSL certificates to encrypt data between the user and server.

Authentication modules provided by Django for secure logins and data access.

5.4 Benefits of the System Specifications

1. Cross-Platform Accessibility: With the web-based approach, users can access their portfolios from any device, regardless of the operating system, as long as they meet the hardware and software requirements.

2. Cost-Effective: By relying on open-source technologies like Django, Python, and SQLite, the system minimizes costs while maximizing performance and reliability.

3. Scalability: The use of cloud hosting services and Django's modular architecture allows the platform to handle an increasing number of users without compromising performance.

4. User-Friendly Interface: The high-resolution display and responsive design ensure that the application provides an intuitive experience for both creators and viewers.

CHAPTER 6

METHODOLOGY

6.1 Conceptual Framework

The conceptual framework forms the backbone of the Public Portfolio Website, focusing on the alignment of user needs, system goals, and technical capabilities.

Purpose of the Platform

- To provide a centralized space where individuals can showcase their skills, achievements, and work experience.
- To serve as a professional networking tool, enabling users to share their profiles easily with potential employers, clients, or collaborators.
- To empower individuals with limited technical expertise to create professional portfolios effortlessly.

Scope of the Platform

1. Personal Portfolios: Targeted at students, freelancers, and professionals looking to establish a digital presence.

2. Professional Growth: Aids in career development by offering features like resume generation and analytics.

3. Global Accessibility: Designed for a diverse user base, accommodating multiple languages and regional preferences.

Core Features Explained

1. Customizable Profiles: Users can create profiles tailored to their personal or professional needs.

2. Public Profile Sharing: Unique URLs allow users to showcase their profiles on social media, emails, and job platforms.

3. Interactive Dashboard: Provides insights into profile performance, such as visitor statistics and engagement levels.

6.2 Technology Stack

The technology stack is a carefully chosen blend of modern tools and frameworks that ensure performance, scalability, and user satisfaction.

Frontend Development

React.js: Enables the creation of dynamic, component-based interfaces. Offers fast rendering and real-time updates.

CSS Frameworks: Tailwind CSS allows rapid styling, and Bootstrap ensures responsiveness across devices.

Backend Development

Django: Provides a robust framework for implementing business logic. Handles user authentication, form validation, and data processing.

Database and Storage

SQLite: A reliable, scalable database for storing user data.

Hosting and Deployment

AWS: Provides scalability and reliability with services like EC2 and S3.

CI/CD Pipelines: Tools like GitHub Actions automate testing and deployment, ensuring seamless updates.

6.3 Data Flow and Architecture

User Interaction Layer

Data Input: Users fill out forms for personal details, education, work experience, and portfolio projects. Real-time validation ensures error-free data submission.

Dynamic Profile Rendering: Profiles are generated dynamically, allowing for personalized layouts and styles.

Backend Processing

Data Handling: The backend validates and processes data before storing it securely in the database. Role-based access control ensures only authorized users can make changes.

Resume Generation: Uses templates to create professional resumes in real-time.

Public Sharing and Privacy Controls

Unique URL Generation: Users receive a sharable link to their public profile.

Privacy Management: Profiles can be customized to control what information is visible to the public.

6.4 User Interface and Experience Design

The design principles prioritize simplicity, functionality, and inclusivity to create a superior user experience.

Design Principles

Simplicity: Minimalist design to avoid overwhelming users. Clear navigation paths and logical organization of content.

Consistency: Uniform styles for buttons, fonts, and colors ensure a cohesive look.

Inclusivity: Features like screen readers, high-contrast themes, and keyboard navigation cater to all users.

Innovative Features

Real-Time Feedback: Users receive instant notifications for incomplete fields or successful updates.

Profile Customization: Users can adjust colors, fonts, and layouts to reflect their personality.

Responsive Design

Mobile Optimization: Ensures the platform is accessible on all devices, from desktops tosmartphones.

Adaptive Layouts: Media queries and flexible grids guarantee a seamless experience on different screen sizes.

6.5 Security and Privacy Measures

Authentication and Access Control

Secure Login: Supports multi-factor authentication (MFA) for added security.

Role-Based Access Control (RBAC): Admins manage data, while regular users control their profiles.

Data Encryption and Storage

Encryption: Data is encrypted both in transit (using SSL/TLS) and at rest (using AES-256).

Storage Compliance: Adheres to GDPR and CCPA for global data privacy standards.

Regular Audits and Monitoring

Security Audits: Regular testing to identify and mitigate vulnerabilities.

Activity Logs: Tracks changes and access for accountability.

6.6 Benefits and Unique Features

The platform offers unparalleled advantages to its users.

Professional Growth

Resume Optimization: ATS-friendly templates increase job application success rates. **Networking**: Public profiles make it easy to share professional details with potential employers.

Engagement Metrics: Users can view the number of profile visits and downloads. **Insights for Improvement:** Suggestions for profile enhancement based on visitor behaviour.

Global Reach and Inclusivity

Accessibility Features: Tools for users with disabilities ensure inclusivity.

6.7 Testing and Deployment Strategy

The system undergoes rigorous testing and a systematic deployment process to ensure reliability.

Testing Phases

Unit Testing: Ensures individual components function as intended.

Integration Testing: Validates interaction between the frontend, backend, and database. User Acceptance Testing (UAT): Collects feedback from target users to refine the system.

Deployment Process

Server Configuration: Deployed on cloud platforms with auto-scaling capabilities.

Monitoring: Tools like New Relic monitor performance and uptime.

Future Maintenance

Feature Updates: Regularly introducing new features based on user feedback.

Bug Fixes: Prompt resolution of issues to maintain smooth operations.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

The development of the Public Portfolio Website marks a significant step toward empowering individuals with a professional online presence. The system successfully integrates essential features such as profile creation, resume generation, and public profile sharing, while prioritizing user experience and data security. Its intuitive design ensures accessibility for users of all technical backgrounds, making it a versatile tool for students, professionals, and freelancers alike. By adhering to modern web standards and leveraging robust technologies like Django and PostgreSQL, the platform delivers a seamless and efficient experience.

The Public Portfolio Website not only meets its core objectives but also demonstrates scalability and adaptability, making it suitable for future expansions. Its role in facilitating professional growth and networking underscores its value in today's competitive job market. With rigorous testing and security measures in place, the platform is poised to handle real-world demands while safeguarding user data. Overall, this project exemplifies the potential of technology in transforming traditional approaches to personal branding and career development.

7.2 Future Enhancements

While the Public Portfolio Website achieves its current goals, there are several avenues for future enhancements that can further elevate its impact:

1. AI-Driven Recommendations: Implementing AI algorithms to provide personalized suggestions for improving user profiles based on industry trends and recruiter preferences.

2. Integration with Social Media Platforms: Allowing users to link their portfolios with LinkedIn, GitHub, and other platforms to enhance their professional visibility.

3. Multilingual Support: Expanding language options to cater to a global audience, making the platform accessible to non-English speakers.

4. Advanced Analytics Dashboard: Offering detailed insights into profile performance, such as visitor demographics and engagement metrics.

5. Mobile Application Development: Extending platform accessibility through dedicated Android and iOS applications, ensuring seamless user experience across devices.

6. Features: Introducing badges and progress trackers to motivate users to complete and optimize their profiles.

7. Collaboration Tools: Enabling team-based projects where users can collaborate, showcase group achievements, and share resources.

APPENDIX - 1

SOURCE CODE

profile.html

```
{% extends '../base/home.html' %}

{% load static %}

{% block stylesheet %}

<link rel="stylesheet" href="{% static 'css/profile.css' %}">

{% endblock %}

{% block main %}

<!-- Profile card -->

<div class="profile-header">

   <div class="profile-cnt">

      <div class="profile">

         <div class="profile-photo-cnt">

            <img src="{{ user_profile.profile_picture.url }}" alt="Profile Picture"
class="profile-photo">

         </div>

         {% if edit_access %}

         <div class="edit-cnt">

            <button type="button" class="edit-btn"><i class="fa-solid fa-pen"></i></button>

         </div>

         {% endif %}

      </div>

      <div class="user-identity-cnt">

         <div class="name-cnt">
```

```
            <h1 class="name">{{ user_profile.name }} <span class="pronouns">({{
user_profile.pronouns }})</span></h1>

            {% if edit_access %}

            <button type="button" class="edit-btn"><i class="fa-solid fa-pen"></i></button>

            {% endif %}

          </div>

          <h3 class="uname"><i class="fa-regular fa-id-badge"></i> {{
user_profile.user.username }}</h3>

          <p class="dob"><i class="fa-solid fa-cake-candles"></i> {{ user_profile.dob }}</p>

          <div class="location">

            <p><i class="fa-solid fa-location-dot"></i> {{ user_profile.district }}</p>

          </div>

          <div class="btn-cnt">

            {% if user_profile.open_to_work %}

            <span class="open-to-work"><i class="fa-solid fa-briefcase"></i> Open to
Work</span>

            {% endif %}

            <a href="{{ user_profile.resume.url }}" download="Resume.pdf"><button
type="button" class="resume"><i class="fa-solid fa-file"></i> Resume</button></a>

          </div>

        </div>

      </div>

      <div class="share-cnt">

        <button type="button" class="share-btn">Share <i class="fa-solid fa-arrow-up-from-
bracket"></i></button>

      </div>
```

```
</div>
<!-- About -->
 {% if user_about %}
<div class="profile-about-cnt">
    <div class="profile-about-title-cnt">
        <h1 class="profile-about-title">{{ user_about.title }}</h1>
        {% if edit_access %}
        <div class="edit-cnt">
            <button type="button" class="edit-btn"><i class="fa-solid fa-pen"></i></button>
        </div>
        {% endif %}
    </div>
    <div class="profle-about-me-cnt">
        <p class="profile-about-me">
            {{ user_about.about_me }}
        </p>
    </div>
</div>
{% endif %}
<!-- Experience -->
{% if user_experience %}
<div class="experiences-cnt">
    <div class="experiences-title-cnt">
        <h1 class="experiences-title">Experience <i class="fa-solid fa-briefcase"></i></h1>
```

```
        {% if edit_access %}
        <div class="add-btn">
            <i class="fa-regular fa-square-plus"></i>
        </div>
        {% endif %}
    </div>
    <div class="experiences-list">
        <!--  -->
        {% for experience in user_experience %}
        <div class="experience-detail-cnt">
            <div class="experience-detail">
                <div class="job-role-cnt">
                    <h2 class="job-role">{{ experience.title }} | {{ experience.company }}</h2>
                    {% if edit_access %}
                    <div class="edit-cnt">
                        <button type="button" class="edit-btn"><i class="fa-solid fa-
pen"></i></button>
                    </div>
                    {% endif %}
                </div>
                <div class="duration">{{ experience.start_date }} - {{ experience.end_date
}}</div>
                <div class="experience-description-cnt">
                    <p class="experience-description">{{ experience.description }}</p>
                </div>
```

```
            </div>
            {% if edit_access %}
            <form class="del_form" action="{% url 'del_user_experienc' experience.id %}"
method="post">
                {% csrf_token %}
                <button type="submit" class="delete-btn">
                    <i class="fa-regular fa-trash-can"></i>
                </button>
            </form>
            {% endif %}
        </div>
        {% endfor %}
        <!--  -->
    </div>
</div>
{% endif %}
<!-- Education -->
{% if user_educations %}
<div class="education-cnt">
    <div class="education-title-cnt">
        <h1 class="education-title">Education <i class="fa-solid fa-graduation-cap"></i></h1>
        {% if edit_access %}
        <div class="add-btn">
            <i class="fa-regular fa-square-plus"></i>
        </div>
```

```
        {% endif %}
    </div>
    <div class="education-list">
        <!--  -->
        {% for education in user_educations %}
        <div class="education-detail-cnt">
            <div class="education-detail">
                <div class="education-information-cnt">
                    <div class="institution-name-cnt">
                        <h2 class="institution-name">{{ education.school_name }}</h2>
                        {% if edit_access %}
                        <div class="edit-cnt">
                            <button type="button" class="edit-btn"><i class="fa-solid fa-
pen"></i></button>
                        </div>
                        {% endif %}
                    </div>
                    <h4 class="degree">{{ education.degree }} <span class="grade">Grade : {{
education.grade }}</span></h4>
                    <p class="area-of-study">{{ education.course }}</p>
                    <p class="duration">{{ education.start_date }} - {{ education.end_date }}</p>
                </div>
            </div>
            {% if edit_access %}
```

```
            <form class="del_form" action="{% url 'del_user_education' education.id %}"
method="post">
                {% csrf_token %}
                <button type="submit" class="delete-btn">
                    <i class="fa-regular fa-trash-can"></i>
                </button>
            </form>
            {% endif %}
        </div>
        {% endfor %}
    </div>
</div>
{% endif %}
<!-- Certificates -->
{% if user_certificates %}
<div class="certifications-cnt">
    <div class="certification-title-cnt">
        <h1 class="certification-title">Certifications <i class="fa-solid fa-award"></i></h1>
        {% if edit_access %}
        <div class="add-btn">
            <i class="fa-regular fa-square-plus"></i>
        </div>
        {% endif %}
    </div>
    <div class="certificate-list">
```

```
<!--  -->
{% for certificate in user_certificates %}
<div class="certificate">
    <div class="certificate-provier-logo-cnt">
        <img src="{% static 'images/icons/certificateicon.png' %}" alt="" class="certificate-provier-logo" width="100px">
    </div>
    <div class="certificate-details-cnt">
        <div class="certificate-details">
            <div class="certificate-name-cnt">
                <h2 class="certificate-name">{{ certificate.title }}</h2>
                {% if edit_access %}
                <div class="edit-cnt">
                    <button type="button" class="edit-btn"><i class="fa-solid fa-pen"></i></button>
                </div>
                {% endif %}
            </div>
            <div class="issued-details">
                <h4 class="issued-by">Issued By: {{ certificate.issued_by }}</h4>
                <p class="issued-by">Issued On: {{ certificate.issued_on }}</p>
            </div>
            <a href="{{ certificate.certificate_url }}"><button type="button" class="view-certificate-btn"><i class="fa-solid fa-arrow-up-right-from-square"></i> View Certificate</button></a>
```

```
            </div>
            {% if edit_access %}
            <form class="del_form" action="{% url 'del_user_certificate' certificate.id %}"
method="post">
                {% csrf_token %}
                <button type="submit" class="delete-btn">
                    <i class="fa-regular fa-trash-can"></i>
                </button>
            </form>
            {% endif %}
          </div>
        </div>
        {% endfor %}
        <!--  -->
    </div>
</div>
{% endif %}
<!-- Projects -->
 {% if user_projects %}
<div class="project-cnt">
    <div class="project-title-cnt">
        <h1 class="project-title">Projects <i class="fa-solid fa-gear"></i></h1>
        {% if edit_access %}
        <div class="add-btn">
            <i class="fa-regular fa-square-plus"></i>
```

```
        </div>
        {% endif %}
    </div>
    <div class="project-list">
        {% for project in user_projects %}
        <div class="project">
            <div class="project-image-cnt">
                <img src="{% static 'images/icons/projecticon.png' %}" alt="" class="project-provier-logo" width="100px">
            </div>
            <div class="project-details-cnt">
                <div class="project-details">
                    <div class="project-name-cnt">
                        <h2 class="project-name">{{ project.title }}</h2>
                        {% if edit_access %}
                        <div class="edit-cnt">
                            <button type="button" class="edit-btn"><i class="fa-solid fa-pen"></i></button>
                        </div>
                        {% endif %}
                    </div>
                    <div class="created-details">
                        <p class="created-on">{{ project.start_date }} - {{ project.end_date }}</p>
                    </div>
                    <div class="certificate-description-cnt">
```

```
                    <p class="certificate-description">{{ project.description }}</p>
                </div>
                <a href="{{ project.project_url }}"><button type="button" class="view-project-btn"><i class="fa-solid fa-arrow-up-right-from-square"></i> View Project</button></a>
            </div>
            {% if edit_access %}
            <form class="del_form" action="{% url 'del_user_project' project.id %}" method="post">
                {% csrf_token %}
                <button type="submit" class="delete-btn">
                    <i class="fa-regular fa-trash-can"></i>
                </button>
            </form>
            {% endif %}
          </div>
        </div>
        {% endfor %}
    </div>
</div>
{% endif %}
<!-- Skills -->
 {% if user_skills %}
<div class="profile-skills-cnt">
    <div class="profile-skills-title-cnt">
```

```
        <h1 class="profile-skills-title">Skills <i class="fa-solid fa-person-
snowboarding"></i></h1>
        {% if edit_access %}
        <div class="edit-cnt">
            <button type="button" class="edit-btn"><i class="fa-solid fa-pen"></i></button>
        </div>
        {% endif %}
    </div>
    <div class="profle-skills-list">
        {% for skill in user_skills %}
        <span class="skills {{ skill.proficiency.lower }}">{{ skill.skill }}</span>
        {% endfor %}
    </div>
    <div class="profeciancy-level-cnt">
        <div class="profeciancy-level">
            <div class="circle beginner-level"></div>
            <p>Beginner</p>
        </div>
        <div class="profeciancy-level">
            <div class="circle intermediate-level"></div>
            <p>Intermediate</p>
        </div>
        <div class="profeciancy-level">
            <div class="circle expert-level"></div>
            <p>Expert</p>
```

```
        </div>
    </div>
</div>
{% endif %}
<!-- Languages Known -->
 {% if user_languages %}
<div class="languages-known-cnt">
    <div class="languages-known-title-cnt">
        <h1 class="languages-known-title">Languages Known <i class="fa-solid fa-earth-asia"></i></h1>
        {% if edit_access %}
        <div class="edit-cnt">
            <button type="button" class="edit-btn"><i class="fa-solid fa-pen"></i></button>
        </div>
        {% endif %}
    </div>
    <div class="languages-known-list">
        {% for language in user_languages %}
        <span class="languages {{ language.proficiency.lower }}">{{ language.language }}</span>
        {% endfor %}
    </div>
    <div class="profeciancy-level-cnt">
        <div class="profeciancy-level">
            <div class="circle beginner-level"></div>
```

```
            <p>Beginner</p>
        </div>
        <div class="profeciancy-level">
            <div class="circle intermediate-level"></div>
            <p>Intermediate</p>
        </div>
        <div class="profeciancy-level">
            <div class="circle expert-level"></div>
            <p>Expert</p>
        </div>
    </div>
</div>
{% endif %}
{% endblock %}
```

views.py

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from django.contrib.auth.models import User
from django.http.response import HttpResponse
from django.urls import reverse
from django.contrib import messages
from .models import *
```

```
def user_login(request):
    if (request.method == "POST"):
        uname_or_email = request.POST.get("uname_or_email")
        password = request.POST.get("password")
        error_message = ""
        if ("@" in uname_or_email):
            user = authenticate(email = uname_or_email, password = password)
        else:
            user = authenticate(username = uname_or_email, password = password)
        # the user variable contains the user name of the user if avilable else it contains None
        if user is not None:
            # this login() function is from the contrib.auth whick logins to the user
            login(request, user)
            return redirect(reverse("user_profile", kwargs={"uname" : user})) # keyword
argument is used to send data requeired for the url
        else:
            error_message = "Invalid User name or Password"
            return render(request, "auth/login.html", context={"error" : error_message})
    return render(request, "auth/login.html")
@login_required(login_url='user_login')
def user_logout(request):
    logout(request)
    return redirect(reverse("user_login"))
def user_signin(request):
    return render(request, "auth/signin.html")
```

```
def home(request):
    if request.user.is_authenticated:
        logged_user = get_object_or_404(UserProfile, user=request.user)
    else:
        logged_user = None
    context = {
        'logged_user': logged_user
    }
    if(request.method == "POST"):
        search = request.POST.get("search")
        user = User.objects.filter(username = search).first()
        if (user is not None):
            return redirect(reverse("user_profile", kwargs={"uname" : search}))
    return render(request, "base/home.html", context)
def user_profile(request, uname):
    user = get_object_or_404(User, username=uname)
    user_profile = get_object_or_404(UserProfile, user=user)
    user_about = user_profile.about_me
    user_experience = user_profile.experiences.all()
    user_educations = user_profile.educations.all()
    user_certificates = user_profile.certificates.all()
    user_projects = user_profile.projects.all()
    user_skills = user_profile.skills.all()
    user_languages = user_profile.languages.all()
```

```
if request.user.is_authenticated:
    logged_user = get_object_or_404(UserProfile, user=request.user)
else:
    logged_user = None
edit_access = uname == request.user.username
context = {
    'edit_access': edit_access,
    'uname': uname,
    'user_profile': user_profile,
    'user_about': user_about,
    'user_experience': user_experience,
    'user_educations': user_educations,
    'user_certificates': user_certificates,
    'user_projects': user_projects,
    'user_skills': user_skills,
    'user_languages': user_languages,
'logged_user': logged_user,
    }
if request.method == "POST":
    search = request.POST.get("search")
    searched_user = User.objects.filter(username=search).first()
    if searched_user:
        return redirect(reverse("user_profile", kwargs={"uname": search}))
    else:
```

```
            return HttpResponse("Profile Not Found!")
    return render(request, "user/profile.html", context)
def personal_details_form(request):
    return render(request, "forms/personal_details_form.html")
@login_required(login_url='user_login')
def del_user_experienc(request,id):
    if request.method == "POST":
        experience = get_object_or_404(Experience, id = id)
        experience.delete()
    return redirect(reverse('user_profile', kwargs={"uname" : request.user.username}))
@login_required(login_url='user_login')
def del_user_education(request,id):
    if request.method == "POST":
        education = get_object_or_404(Education, id = id)
        education.delete()
    return redirect(reverse('user_profile', kwargs={"uname" : request.user.username}))
@login_required(login_url='user_login')
def del_user_certificate(request,id):
    if request.method == "POST":
        certificate = get_object_or_404(Certificate, id = id)
        certificate.delete()
    return redirect(reverse('user_profile', kwargs={"uname" : request.user.username}))
@login_required(login_url='user_login')
def del_user_project(request,id):
```

```
    if request.method == "POST":
        project = get_object_or_404(Project, id = id)
        project.delete()
    return redirect(reverse('user_profile', kwargs={"uname" : request.user.username}))
# Create your views here.
```

urls.py

```
from django.urls import path
from . import views
urlpatterns = [
    path('', views.home, name="home"),
    path('in/<str:uname>/', views.user_profile, name="user_profile"),
    path('login/', views.user_login, name="user_login"),
    path('logout/', views.user_logout, name="user_logout"),
    path('signin/', views.user_signin, name="user_signin"),
    path('personal_details_form/', views.personal_details_form,
name="personal_details_form"),
    path('del_user_experienc/<int:id>', views.del_user_experienc,
name="del_user_experienc"),
    path('del_user_education/<int:id>', views.del_user_education,
name="del_user_education"),
    path('del_user_certificate/<int:id>', views.del_user_certificate,
name="del_user_certificate"),
    path('del_user_project/<int:id>', views.del_user_project, name="del_user_project"),
]
```

APPENDIX - 2

SCREENSHOTS

SAMPLE OUTPUT

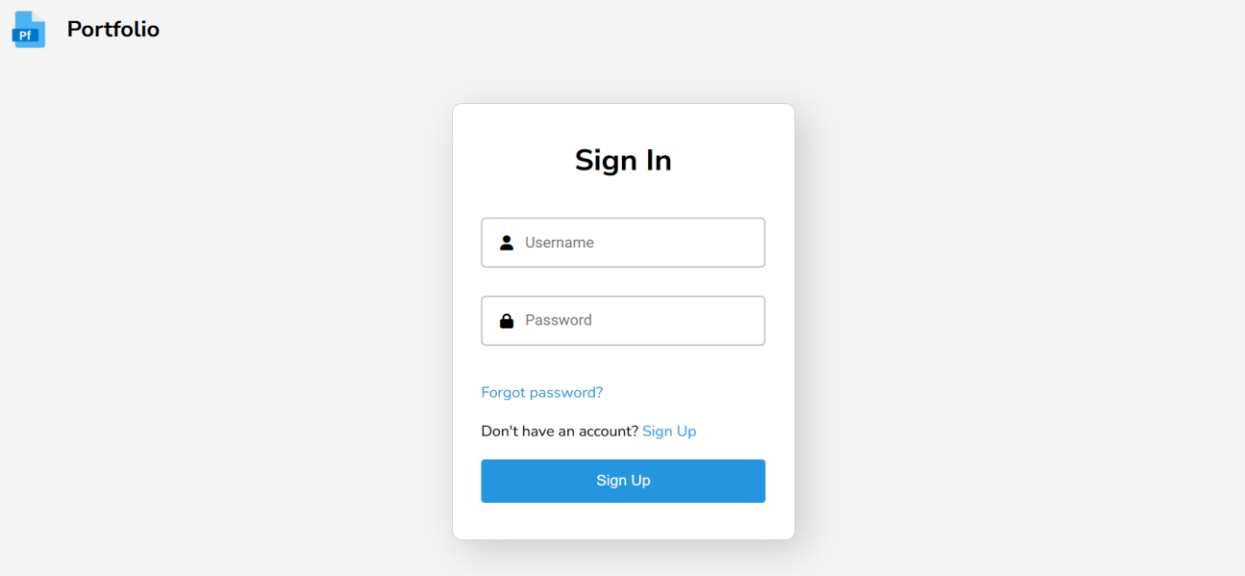

Figure A.2.1 Sign in Page

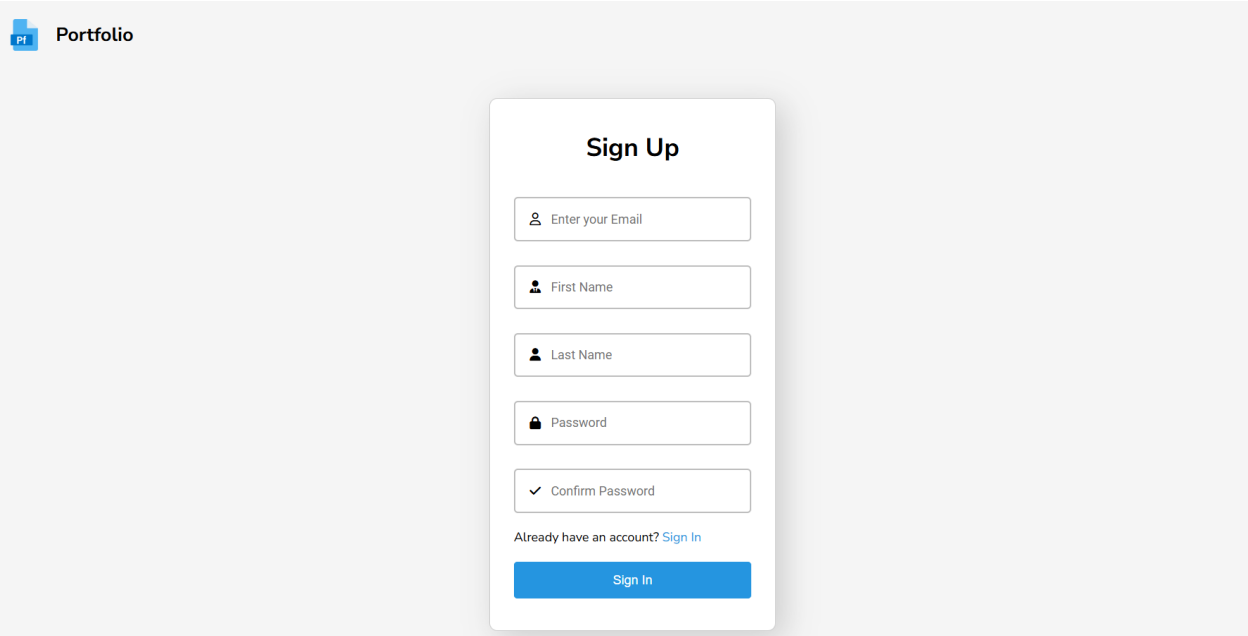

Figure A.2.2 Sign Up Page

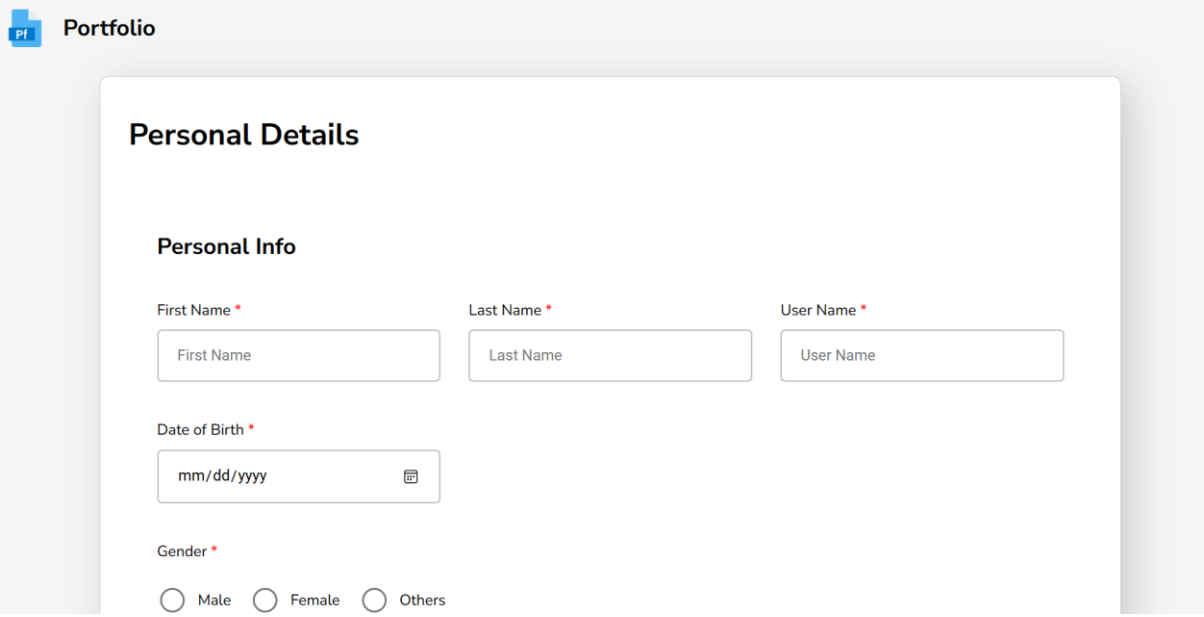

Figure A.2.3 Registration Form

Figure A.2.4 Profile with Edit Option

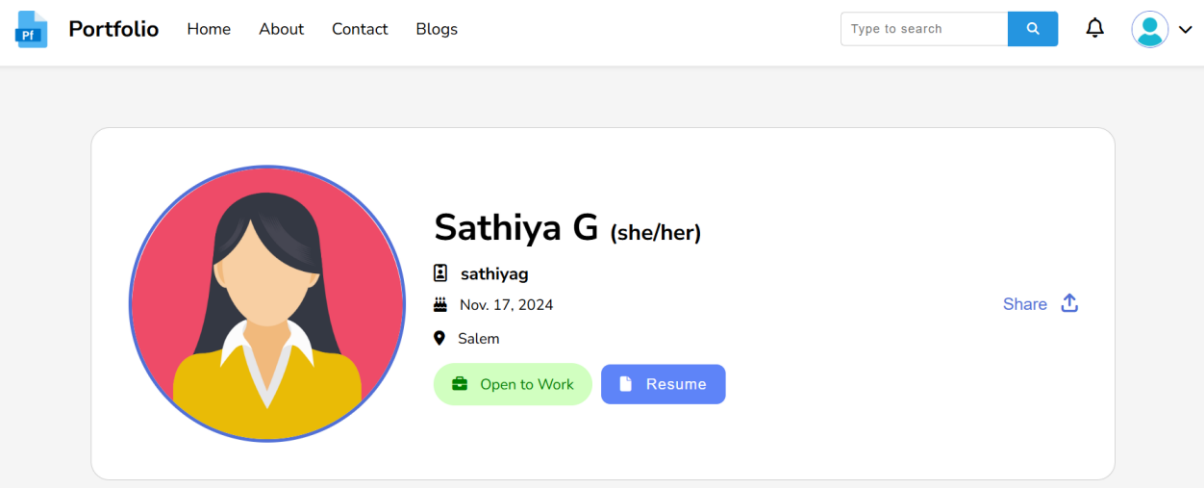

Figure A.2.5 Viewing Other Users Profile

REFERENCES

1. Ghasemzadeh, F., & Archer, N. P. (2000). Project portfolio selection through decision support. Decision Support Systems, 29(1), 73-88.
2. Django Software Foundation. (n.d.). Django documentation. Retrieved from https://docs.djangoproject.com
3. Augustin, P., & Constanta-Nicoleta, B. (2014). Project Prioritization and Portfolio Performance Measurement in Project Oriented Organizations. Procedia - Social and Behavioral Sciences 119(2014), 339-348. https://doi.org/10.1016/j.sbspro.2014.03.039
4. Englund, R. L., & Graham, R. J. (1999). From Experience: Linking Projects to Strategy. Journal of Production and Innovation Management, 16, 52-64.
5. Archer, N., & Ghasemzadeh, F. (2004). Project portfolio selection and management, In P. W. G. Morris & J. K. Pinto (Eds.). The Wiley guide to managing projects, 237-255. Hoboken, NJ: John Wiley & Sons.
6. Caballero, H. C., & Schmidt, E. K. (2014). Decision support system for portfolio components selection and prioritizing. paper presented at PMI® Global Congress 2014—North America, Phoenix, AZ. Newtown Square, PA: Project Management Institute
7. Voss, M. (2012). Impact of customer integration on project portfolio management and its success-developing a conceptual framework. International Journal of Project Management, 30, 567581. https://doi.org/10.1016/j.ijproman.2012.01.017
8. Yelin, K. C. (2005). Linking Strategy and Project Portfolio Management. In Levine, H. A. (eds.) (2005) Project Portfolio Management: A practical guide to selecting projects, managing portfolios and maximizing benefit, 137- 145. USA: Pfeiffer Wiley.
9. Project Management Institute. PMI (2013). The Standard of Portfolio Management. 3.ed. Project Management Institute, Inc. Newtown Square – PA.
10. Bible, M. J., Bivins, S., & Bivins, S. S. (2011). Mastering Project Portfolio Management: A Systems Approach to Achieving Strategic Objectives. J Ross Press Series, ISBN: 9781604270662, https://books.google.co.ma/books?id=8B-XmtUKOWMC