GCUF

Government College University Faisalabad

Final Documentation

AICA
(Artificial Intelligence Career Assistant)

By

Abdullah Arshad	**2021-GCUF-088742**
Hannan Ahmad	**2021-GCUF-088858**
M. Afaq Zafar	**2021-GCUF-073597**

Project submitted in partial fulfillment of the
requirement for the degree of

**BACHELORS OF SCIENCES
IN
COMPUTER SCIENCE**

DEPARTMENT OF COMPUTER SCIENCE

GOVERNMENT COLLEGE UNIVERSITY FAISALABAD

November (2021-2025)

ACKNOWLDGMENT

DEDICATION

This final year project report was submitted to **Government College University, Faisalabad**; in partial fulfillment of the requirements for the degree of **Bachelor of Science in Computer Science**. We are especially grateful to our loving parents for their endless encouragement and prayers. We also extend our sincere appreciation to our teachers, whose guidance and support have played a vital role in the successful completion of this project. Above all, we express our deepest gratitude to **Almighty Allah** for granting us the ability, patience, and determination to accomplish this work.

DECLARATION

We carried out the work presented in this project under the supervision of **Sir Tayyab Ali**, Department of Computer Science, **Government College University, Faisalabad**, Pakistan. We hereby declare that this project, in whole or in part, has not been copied from any source. The development of this project and the preparation of the accompanying report are entirely based on our own efforts. If any part of this project is found to be plagiarized, we accept full responsibility and are prepared to face the consequences. Furthermore, this work has not been submitted for the award of any other degree or qualification at this or any other university or institution.

Group Members:

Signature of student

Name: Abdullah Arshad

Registration No: 2021-GCUF-088742

Signature of student

Name: Hannan Ahmad

Registration No: 2021-GCUF-088858

Signature of student

Name: M. Afaq Zafar

Registration No: 2021-GCUF-073597

Certificate by the Supervisory Committee

This is to certify that the final year project of BS (Hons) in CS "**Artificial Intelligence Career Assistant**" project was developed by "**ABDULLAH ARSHAD**", "**HANNAN AHMAD**" AND "**M. AFAQ ZAFAR**" under registration number (2021-GCUF-088742, 2021-GCUF-088858, 2018-GCUF-075743) respectfully supervised by "**MR. TAYYAB ALI**" and that in his opinion it is fully adequate, for the degree of BS (Hons) in Computer Science.

Co-Supervisor (if any)

Signature of Supervisor ………………….

Name: …………………………………...

Designation with Stamp……………………..

Member of supervisory committee

Signature of Supervisor ………………….

Name: …………………………………...

Designation with Stamp……………………..

Member of supervisory committee

Signature of Supervisor ………………….

Name: …………………………………...

Designation with Stamp……………………..

<u>Chairperson</u>

Signature with Stamp……………………………

<u>Dean / Academic Coordinator</u>

Signature with Stamp……………………………

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1. Introduction

When we talk about the job market in the modern world, the role of artificial intelligence and career guidance will definitely come up. The recruitment process has evolved over time, but still, job seekers often struggle to match their skills with what companies require. In countries like Pakistan and across the globe, job seekers—especially fresh graduates—face difficulties in identifying their weaknesses and understanding what skills or experience they lack according to industry standards.

We have found that there is no centralized or smart platform that helps individuals analyze their resumes and compare them against real job descriptions posted by companies. Most candidates are unaware of the areas they need to improve to match specific job roles, such as domain knowledge, technical skills, soft skills, and experience.

To solve this issue, we are providing a smart and user-friendly platform called the **AI Career Assistant**. Through this platform, users can upload their CVs and job descriptions, and the system will automatically detect shortcomings and weaknesses by analyzing both documents using Artificial Intelligence techniques. The results will be shown graphically, divided into four key sections: **domain knowledge, technical skills, experience, and soft skills**. This platform also allows users to perform a **mock interview** with an AI model trained to simulate real interview scenarios. Users can interact with the AI to answer job-related questions and receive feedback, helping them prepare better for actual interviews. This feature is particularly helpful for those who do not have access to professional coaching or interview training sessions.

The **AI Career Assistant** saves all data securely so users can view their progress and feedback over time. The platform is designed to be responsive, informative, and fully aligned with modern recruitment and learning needs. It provides a new and intelligent way to prepare job seekers for the real world while also assisting HR professionals in evaluating candidate readiness more efficiently.
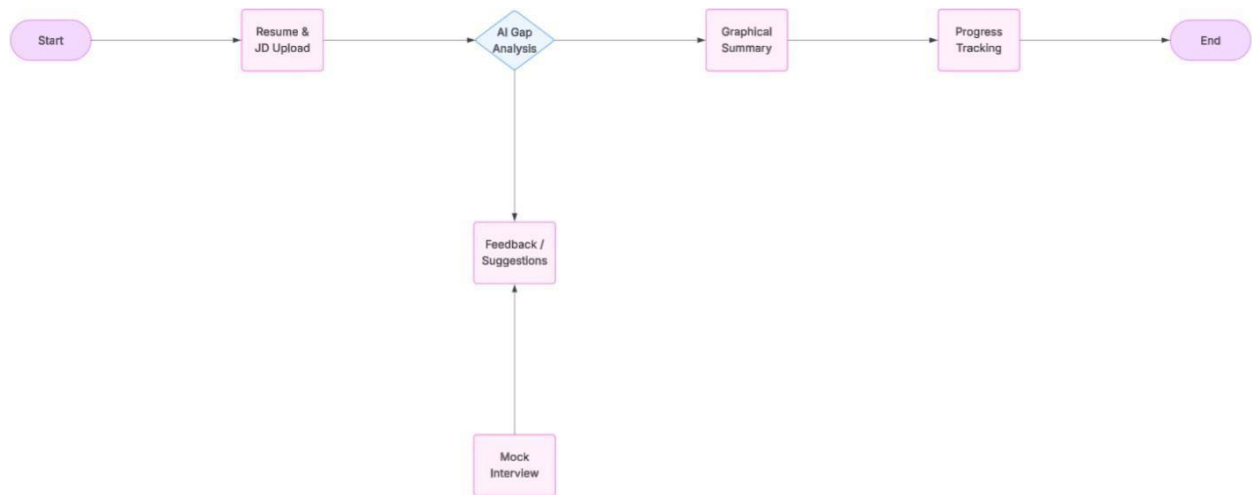
Figure: 1.1 Introduction

1.2. Background

In today's competitive job market, securing employment is not just about having a degree or basic qualifications. Employers now look for well-rounded candidates who possess the right blend of technical knowledge, domain expertise, experience, and soft skills. However, most job seekers lack a clear understanding of how their current profile compares with the expectations outlined in job descriptions. As a result, many talented individuals fail to secure opportunities due to a lack of personalized guidance.

Traditionally, candidates rely on general advice from seniors, friends, or online articles to improve their CVs and prepare for interviews. This often leads to a one-size-fits-all approach, which does not cater to individual shortcomings. Moreover, manual CV review by HR departments is time-consuming and inconsistent. There is a clear gap between what candidates believe they are capable of and what the job market actually demands.

On the other hand, many job seekers are not aware of how to prepare for technical interviews or how to highlight their strengths during job applications. They do not receive personalized feedback, and most mock interview sessions are limited to in-person workshops, which are not accessible to everyone due to financial, geographical, or time constraints.

With the rapid advancement of Artificial Intelligence (AI) and Prompt Engineering, it is now possible to build intelligent systems that can evaluate documents, extract meaningful patterns, and offer recommendations based on real-time data. Leveraging this technology can greatly benefit both job seekers and recruiters.

1.3. Purpose

The purpose of our project, AI Career Assistant, is to help job seekers understand where they stand in today's job market and what they need to improve in order to land their desired jobs. Many people, especially fresh graduates, don't know exactly what's missing in their CVs or how to prepare for job interviews. This platform is built to solve that problem in a simple and smart way.

With the help of AI, our system compares a user's CV with a job description and shows what areas need improvement. These areas are divided into four main parts: technical skills, domain knowledge, experience, and soft skills. The results are shown in a graph so users can easily see where they're strong and where they need to improve.

Another big part of our project is the mock interview feature. A lot of people get nervous during interviews or don't know how to answer questions properly. Our system lets users take practice interviews with an AI that acts like a real interviewer. It's a great way to build confidence and get better without needing to go anywhere.

Overall, our goal is to make career planning easier and more personalized. Whether someone is applying for their first job or switching careers, this platform will guide them step-by-step. It's simple, helpful, and saves both time and effort.

1.4. Scope

Our main purpose is to provide an intelligent and easy-to-use platform where job seekers can check the gaps in their resumes and also practice interviews online. The system will offer smart suggestions to help them improve their skills and prepare better for job opportunities. The platform is divided into different modules to increase user interest and make it more useful for everyone. Here are some key points which explain the scope, features, and limitations of our platform:

- Users must upload their **CV** and a **job description** to get the gap analysis.
- After registration, users will receive login credentials to access their dashboard.
- Users will be able to view graphical **summary** of their weak and strong areas in:
1. Domain Knowledge.
2. Technical Skills.
3. Experience.
4. Soft Skills.

- Users can participate in **AI-based mock interviews** and receive feedback.

- All records will be saved, so users can track their performance over time.

- A personal profile will be created for each registered user.

- Users will be able to download personalized reports after analysis.

Limitations

- Unregistered users won't be able to access the platform or use any services.

- The platform does **not apply for jobs** or generate full CVs.

- The system won't provide job listings or company contacts—its main focus is **skill gap analysis**

- Only the **admin** can manage backend data like user records or system content.

- The AI mock interview is limited to text-based interaction (no voice or video currently).

- The system won't give certified coaching or training—only guidance based on analysis.

- Data will only be available for users who are already registered; no public browsing allowed.
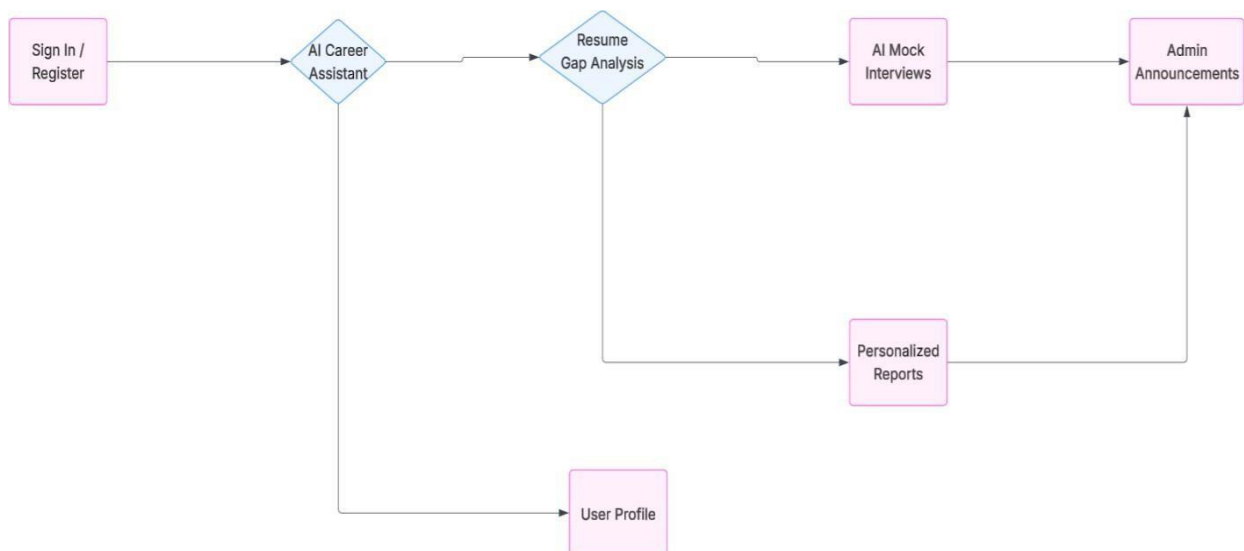
Figure: 1.4 Scope

1.5. Objective

Following are the main objectives of our **AI Career Assistant** system:

- Design and develop a user-friendly AI-powered career guidance platform.

- The main goal is to help users match their CV with real job descriptions and highlight the gaps in four areas: **technical skills**, **domain knowledge**, **experience**, and **soft skills**.

- Allow users to **upload their resume and a job description**, and get detailed feedback and recommendations instantly.

- Provide a clear **Gap Summary Report**, divided into:

1. Summary of Fit.
2. Gaps & Missing Requirements.
3. Key Gaps Identified.
4. Actionable Recommendations.
5. Rejection Risk.
6. Detailed Observations.

- Make it easy for users to **understand their weaknesses** and improve them before applying for jobs.

- Provide a **mock interview feature** where users can practice real-time job interviews with an AI model to boost their confidence and preparation.

- Save all user analysis and mock interviews, so they can view their **progress over time**.

- Make sure only **registered users** can access the full features of the system.

- Provide users with a **one-click dashboard** to access their skill gaps, past interview results, CV history, and suggestions.

- Allow users to **avoid multiple visits to career counselors or physical mock interview sessions** by using this platform from anywhere.

- Make the system **easy to use, responsive, reliable**, and simple enough for both fresh graduates and working professionals.

Platform Goals	System Objectives
Help users understand what's missing in their resumes.	Develop a system that shows missing skills and experience through a gap analysis.
Provide a smarter way to practice job interviews.	Offer AI-based mock interviews to simulate real-world interview scenarios.
Allow users to improve job readiness without physical coaching.	Give personalized suggestions, tips, and a clear visual overview of skill levels.
Provide a centralized place to track career progress.	Enable registered users to access all their reports and records from one platform.
Offer useful services in a user-friendlyand accessible format.	Design an intuitive and responsive interface for maximum user satisfaction.

Table 1: Objective

1.6.   Intended Audience and Reading Suggestions

Intended Audience

Our project is designed for the following audience:

- **ADMIN**

    Admin is the person responsible for managing the platform, handling user registrations, storing resume/job data, and maintaining system operations.

- **JOB SEEKERS / USERS**

    Users are individuals who upload their CVs and job descriptions to receive detailed feedback about their strengths, weaknesses, and career gaps.

- **DEVELOPERS / TECHNICAL TEAM**

    Developers may be interested in understanding how AI and Prompt Engineering are used in resume screening, gap analysis, and mock interview functionality.

- **CAREER COUNSELORS / HR PROFESSIONALS**

    Professionals who may refer this tool to candidates for improving job readiness or may use it to pre- screen applicants.

Reading Suggestions

Readers should have a basic understanding of how hiring works career development, how job descriptions and resumes are written, and a general interest in AI-based career guidance**.** Familiarity with**,** interviewing, or resume writing would be helpful to better understand the use and importance of this platform.
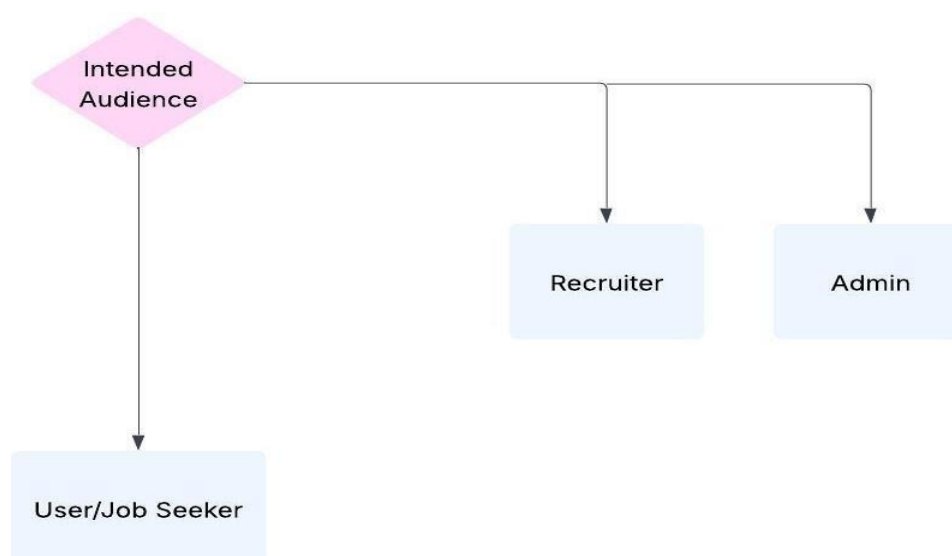
Figure: 1.6 Intended Audience

1.7.   Documentation Conventions

This document use following conventions:

- **Times New Roman** font style is used throughout the document.

- For the main heading, the **"Heading 1"** style is used with font size **14pt**, and for subheadings, **"Heading 2"** and **"Heading 3"** are used with font size **12pt**.

- **Chapter headings** are written in font size **16pt**.

- **Regular text** is in **12pt**, and text alignment is **justified**.

- **Line spacing** is set to **1.5**.

- **Margins** are set as: **Left 1.5 inch**, **Top 1 inch**, **Bottom 1 inch**, and **Right 1 inch**.

- Important points are presented in the form of **bullets**.

- Important or main words are enclosed in **double quotation marks**.

CHAPTER 2
SOFTWARE REQUIREMENT SPECIFACTIONS

2.1. Overall Description

This section provides a general overview of the **AI Career Assistant** system, including its purpose, major features, system constraints, and core assumptions. It describes how the product fits within the broader environment of career services and outlines external factors that may influence its development or performance. This documentation also defines the architectural landscape, user interaction patterns, and technical boundaries under which the system will operate.

2.1.1. Product Perspective

The **AI Career Assistant** is a self-contained, web-based platform developed to support job seekers in evaluating how effectively their resumes align with specific job descriptions. Unlike traditional job portals, this system functions as an **intelligent career advisory tool**. It is built using a modular architecture that encompasses the following key components:

• A responsive **frontend interfaces.**

• A secure and scalable **Flask-based backend server.**

• A **machine learning pipeline** for AI-powered analysis.

• An **interactive mock interview** module for practice.

This product is designed to **complement** popular job search platforms such as **LinkedIn**, **Rozee.pk**, and **Indeed**, by offering personalized insights and actionable feedback to users. It belongs to the growing domain of **AI-driven career development** and **job-readiness enhancement tools**.

2.1.2. Product Features

The system offers a wide range of features to enhance user experience, boost job readiness, and guide personal development:

• **About Agency** – Overview of platform mission and vision.

• **User Registration/ Sign Up.**

• **Log In / Log Out Functionality.**

• **Secure Login and User Profiles.**

- **Resume and Job Description Upload.**

- **AI-Powered Resume Analyzing Engine.**

- **Gap Summary Report.**

- **Graphical Representation of Resume Gaps** (via charts).

- **Mock Interview System** using AI-generated Q/A.

- **Progress Tracking Dashboard.**

- **User-Specific Announcements and Notifications.**

These features aim to guide job seekers step-by-step through resume refinement, skill identification, interview practice, and readiness evaluation.

2.1.3. Design and Implementation Constraints:

The design and implementation will follow a **Waterfall Model** — where each project phase is completed before moving to the next. This ensures structured development and thorough validation at each stage. The following constraints are acknowledged:

• The entire system is divided into separate **frontend**, **backend**, **database**, and **AI logic** modules.

• **User Journey** is optimized — from uploading files to receiving actionable reports and practicing interviews.

• The **UI/UX design** will prioritize accessibility, responsiveness, and clarity.

• Separate login flows for **users** and **admin** will be enforced.

Additional Constraints

• The admin has full authority to manage user records, content, and platform data.

• Registrations must be authentic — fake or invalid entries may distort AI recommendations.

• The database will be continuously maintained by the admin team for performance and reliability.

• Security protocols will be implemented to protect against fake IPs, injections, and unauthorized access.

2.1.4. Assumption and Dependencies:

The system assumes specific conditions and dependencies for its proper operation and accurate results:

Assumptions:

- Users have access to a **stable and high-speed internet connection**.

- Uploaded resumes and job descriptions are **authentic, structured, and text-based**.

- Users are expected to follow system prompts for better output and valid suggestions.

- AI assumes users seek employment in **well-defined roles or industries** (e.g., tech, business, marketing).

Technical Dependencies

- **Backend Framework**: Python with Flask.

- **Frontend**: HTML5, CSS3, Tailwind CSS, Bootstrap, JavaScript.

- **Database**: SQLite with SQLAlchemy ORM.

- **AI & ML**: LangChain for prompt processing, Scikit-Learn for generating gap score.

- **Charting & Visualization**: Chart.js for rendering gap graphs.

- **User Data Handling**: Secure session-based access and role management.

Mock interview limitations: Responses are limited to paragraph-based answers and may not handle irrelevant or off-topic questions.

The system assumes a standard operating environment: **modern browsers**, **updated system libraries**, and

desktop/mobile compatibility.

2.2. System Features

Every website or software application is primarily defined by its functionalities and how effectively those features serve the end users. The **AI Career Assistant** offers a suite of powerful, intelligent features designed to empower users in improving their career readiness. These features work cohesively to help users refine resumes, identify qualification gaps, and prepare for interviews.

2.2.1. Registration

2.2.1.1. Description and Priority

The system provides a secure and role-based **registration and login mechanism** for two types of users: **Job Seekers** and **Administrators**.

·   **Job Seekers** initiate the registration process via the "Sign Up / Login as Job Seeker" button.

·   **Administrators** log in through the "Login as Admin" interface using pre- assigned credentials.

Once registered and authenticated, users gain access to a customized **dashboard**, where they can upload their resumes, receive a comprehensive gap analysis, and interact with the AI-powered **mock interview module**.

Priority: This is a **critical feature** since secure access is foundational to all other platform functionalities.

2.2.1.2. Stimulus/Response Sequence

The following outlines typical user interactions and corresponding system responses:

• **Stimulus:** User clicks on **Sign up.**

Response: The sign-up form appears with required fields (name, email, password, user type).

• **Stimulus:** User fills the form and clicks **Submit.**

Response: The data is validated and saved in the system's database.

• **Stimulus:** Registered user clicks on **Login.**

Response: Login form appears with options: "Login as Job Seeker" or "Login as Admin".

• **Stimulus:** User selects role and enters login credentials.

Response: The system verifies input from the database and grants access if valid.

• **Stimulus:** If login details are incorrect.

Response: The system shows an error message and prompts for correction.

• **Stimulus:** If user **forgets the password**, they click on the **"Forgot Password"**

link.

Response: The system opens a recovery form and asks for the registered email address.

- **Stimulus:** User enters their email and clicks **Send OTP.**

Response: The system sends a **One-Time Password (OTP)** to the registered email.

- **Stimulus:** User enters the OTP and sets a **new password.**

Response: If the OTP is valid, the system updates the password and allows the user to log in.

2.2.1.3. Functional Requirements

- **Req-1:** Users must register by providing necessary information such as name, email, password, and user role (Job Seeker or Admin).

- **Req-2:** The system will verify login credentials from the database before allowing access to the dashboard.

- **Req-3:** Job Seekers must log in using the **"Login as Job Seeker"** option; otherwise, access will be denied.

- **Req-4:** Admin must use the **"Login as Admin"** path to access administrative functions.

- **Req-5:** After successful login, Job Seekers will be redirected to their personal dashboard where they can upload resumes, view reports, and use the mock interview feature.

2.2.2. Services

2.2.2.1. Description and Priority

This section highlights the core **career enhancement services** provided by the **AI Career Assistant** platform. These services are central to the system's value proposition, equipping job seekers with data-driven insights and practical guidance to improve their career prospects.

All features described under this section are fully accessible to registered users upon successful authentication. These services are tightly integrated within the user dashboard and designed to promote ease of use, productivity, and effective self-assessment.

The services include:

- Uploading resumes (CVs) and job descriptions

- Receiving AI-generated career gap analysis reports

- Viewing graphical breakdown of skill and experience deficiencies

- Conducting mock interviews with an AI interviewer

• Getting personalized, actionable recommendations for improvement

Priority: This feature holds **high priority**, as it represents the core functional suite of the platform that supports users throughout their career preparation journey.

2.2.2.2. Stimulus/Response Sequence

• **Stimulus:** After successful login, the user clicks the **"Services"** button from the main navigation bar

Response: The system opens the **Services Dashboard**, displaying all available tools and functionalities in a structured format.

• **Stimulus:** User selects a specific service, such as **"Gap Analysis"**, **"Mock Interview"**, or **"View Reports"**

Response: The system navigates the user to the corresponding tool or feature interface with a smooth transition.

• **Stimulus:** User clicks on the **"Upload Resume and Job Description"** button **Response:** A file upload dialog appears, enabling the user to submit a resume and job description in supported formats (.pdf, .docx, .txt).

• **Stimulus:** User browses previous reports or service history

Response: The system loads saved sessions and displays associated data, including gap summaries and mock interview performance.

2.2.2.3. Functional Requirements

• **Req-1:** All registered users can access the full set of career assistant features after logging in.

• **Req-2:** The system must clearly show all available services and guide users on how to use them.

• **Req-3:** Users should be able to move between features (e.g., upload, mock interview, gap summary) smoothly from the dashboard.

• **Req-4:** No additional payment or unlock process is required; all features are available to every authenticated user.

• **Req-5:** Each service section must respond quickly and load the respective tool/interface when accessed.

2.2.3. Announcement:

2.2.3.1. Description and Priority

The **Announcements** feature enables the **Admin** to broadcast important information to all registered users of the AI Career Assistant platform. This module plays a crucial role in maintaining transparency, improving user awareness, and ensuring timely communication of technical or functional updates.

Announcements are typically used to notify users about:

• Scheduled **system maintenance** or **downtime**

• Introduction of **new features** or platform enhancements

• **Improvements** to the AI models or analytical algorithms

• **Database updates**, resets, or user-impacting operations

• **General notices** or usage policies relevant to all users

All announcements created by the admin are visible to job seekers through a dedicated

"Announcements" section within their dashboard. Optionally, the Admin can choose to

push these updates via email notifications, ensuring that users stay informed even if they are not actively logged in.

Priority: High — this feature is essential for administrative communication and user engagement.

2.2.3.2. Stimulus/Response Sequence

Below is a typical flow of interactions involved in the announcement lifecycle:

• **Stimulus:** Admin drafts a new announcement via the Admin Dashboard

Response: The system saves the announcement to the database and displays it in the

Announcements section for all users.

• **Stimulus:** A registered user (job seeker) clicks on the **"Announcements"** button from the navigation bar or dashboard

Response: The platform opens the **Announcements Page**, listing the most recent updates in reverse chronological order.

- **Stimulus:** The user reads the announcement

Response: The user becomes informed of any ongoing or upcoming system changes and can take relevant action if required (e.g., downloading data before maintenance).

- **Stimulus:** Admin triggers the **"Send Announcement to All Users"** function
  Response: The system automatically delivers the announcement content to every user's registered email address via a mass-notification process.

2.2.3.3. Functional Requirements:

- **Req-1:** The system must allow Admin users to create, edit, and delete announcements from the Admin Dashboard.

- **Req-2:** All announcements must be stored in the system database and displayed in chronological order to users.

- **Req-3:** Registered users should be able to view all announcements from their dashboard via a clearly labeled **Announcements** section.

- **Req-4:** Admin must be able to optionally trigger **email notifications** containing announcement content to all users.

- **Req-5:** The announcement view page must be responsive and easily accessible across all devices.

2.2.4. Contact Us:

2.2.4.1. Description and Priority:

The **Contact Us** feature allows users to directly reach out to the platform's **Admin** or **Support Team** for communication, inquiries, or assistance. Whether users need technical support, want to offer suggestions, ask questions about features, or submit complaints, this feature serves as a formal channel for two-way communication.

It promotes transparency, builds trust, and ensures that user feedback is received and can be acted upon. This page typically provides users with:

- Admin/Support email address.

- Optional contact form to submit a message.

- Other communication channels (e.g., phone number, LinkedIn, or social handle if applicable)

Priority: Medium to High — Enhances user satisfaction and provides a communication bridge between users and administrators.

2.2.4.2. Stimulus/Response Sequence:

• **Stimulus**: User clicks on the **"Contact Us"** button from the navigation bar or footer

Response: The system opens the **Contact Us** page containing a form and relevant contact information.

• **Stimulus**: User fills out the contact form or copies the support email

Response: User can send their message; the system confirms the message has been received (if implemented via form), or the user may email directly.

• **Stimulus**: Admin receives the message and responds accordingly

Response: The user receives a follow-up via email or platform response within a designated time period.

2.2.4.3. Functional Requirements:

• **Req-1:** Users must be able to view admin contact information clearly (email, name, role).

• **Req-2:** The system should provide a secure contact form for submitting inquiries or feedback (optional).

• **Req-3:** The contact information should be consistently accessible via navigation or footer on every page.

• **Req-4:** Users should be able to submit complaints, ask questions, or provide feedback without logging in.

• **Req-5:** The contact form must validate user inputs (e.g., email format, message length).

2.2.5. About Us:

2.2.5.1. Description and Priority:

The **About Us** page introduces the **AI Career Assistant** platform to both new and returning users. It provides essential information about the system's **vision**, **origin**, and **value proposition**, helping users understand:

• The **purpose** behind the creation of the platform

• A brief **background** of its development journey

- An overview of the **team or individual(s)** managing the system

- The **mission**, **values**, and long-term **goals**

- What **differentiates** it from other career advisory tools

This section builds **trust and credibility**, making users more comfortable engaging with the platform.

Priority: High — establishes identity and helps users build a connection with the platform.

2.2.5.2. Stimulus/Response Sequence:

- **Stimulus:** User clicks on the **"About Us"** link in the navigation bar or footer

Response: The system loads and displays the About Us page.

- **Stimulus**: User scrolls through the information

Response: The system presents structured sections outlining the platform's background, values, and mission.

2.2.5.3. Functional Requirements:

- **Req-1:** The system must include a dedicated **About Us** page accessible from the main navigation or footer.

- **Req-2:** The page must display clear and concise background information about the platform's goals and history.

- **Req-3:** Information should be well-structured with headers, bullet points, or paragraphs for readability.

- **Req-4:** The page must be responsive and accessible across all screen sizes (desktop and mobile).

- **Req-5:** Content should be easy to update or expand from the admin interface (if CMS is integrated).

2.3.   External Interface Requirements

2.3.1. User Interface:

The **AI Career Assistant** offers a modern, responsive, and intuitive web-based interface designed with usability and accessibility in mind. The interface supports seamless interaction between users and the system's core functionalities. The layout ensures consistent navigation, a clean visual hierarchy, and responsive behavior across all screen sizes (desktop,

tablet, and mobile).

Once a user successfully registers and logs in, the dashboard becomes the central hub for interaction. From here, users can:

• Upload their resumes and corresponding job descriptions

• Receive AI-generated gap analysis reports

• View graphical summaries of identified gaps

• Participate in role-specific AI-powered mock interviews

• Access personalized recommendations and feedback

• Check announcements or system notifications

• Edit personal details (e.g., name, password)
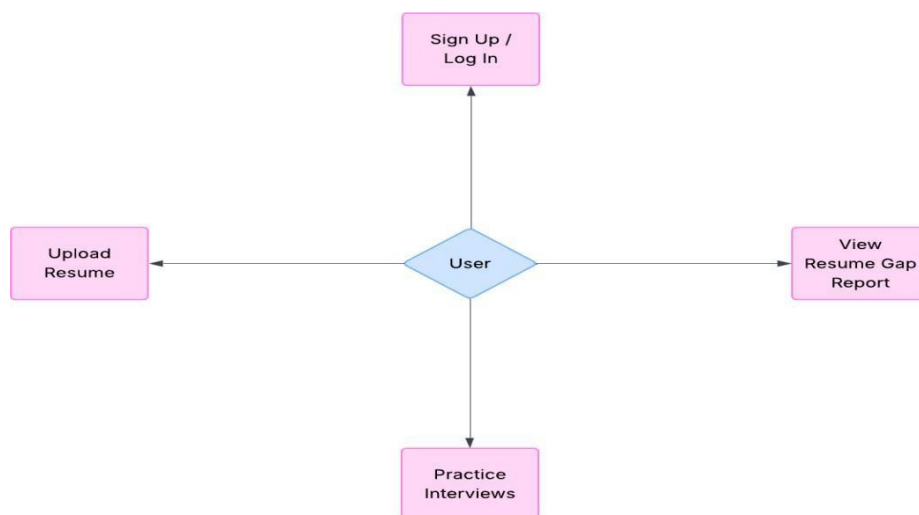
Figure: 2.3.1 User Interface

2.3.2. Admin Interface:

The admin interface is designed for **complete control and oversight** of the platform's operations. Accessible only to authorized personnel, the interface offers an intuitive dashboard that facilitates administration, monitoring, and maintenance of user activity and platform integrity.

Key admin responsibilities include:

• Reviewing and managing user registration and login activity

• Accessing and moderating uploaded user documents

• Posting system-wide announcements and feature updates

• Monitoring AI usage metrics and system logs

• Performing database management (insert/update/delete)

• Managing platform performance, uptime, and content security

The admin has **exclusive access rights** for sensitive actions such as data updates, mass communication, or user moderation to preserve platform integrity and protect user data.
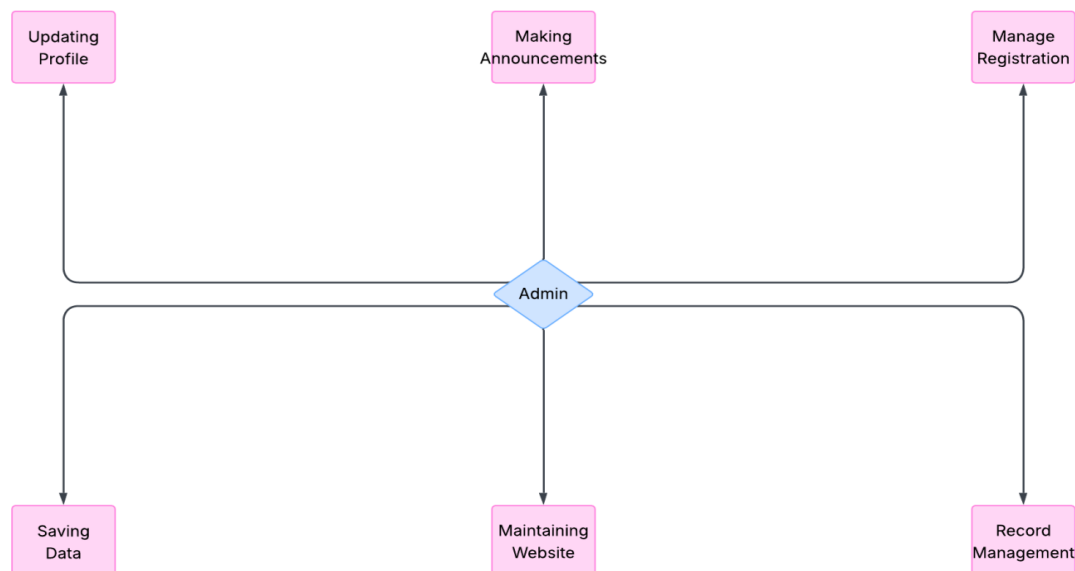
Figure: 2.3.2 Admin Interface

2.3.3. Hardware Interface:

The AI Career Assistant is designed to operate across a variety of internet-enabled devices, including **desktop PCs, laptops, tablets, and smartphones**. It supports adaptive rendering for different screen resolutions and device orientations.

Minimum hardware recommendations for optimal performance:

Component Recommended Specification RAM   4 GB or more.

GPU   2 GB or integrated graphics support.

CPU   Dual-core 2.0 GHz or higher.

Internet   Stable broadband; minimum 50 kbps (400 kbps recommended).

Latency   Preferably under 150 ms for real-time interaction.

The system is compatible with standard input devices (keyboard, mouse, touchscreen) and supports browser-based file uploads.

2.3.4. Software Interface:

The platform relies on modern technologies, development environments, and browsers to ensure compatibility and performance. It is primarily developed and tested on **Windows and macOS** systems using cross-browser compliant code.

Development Stack & Environment:

• **IDE:** PyCharm (Community/Professional Edition) on Windows 8.1, 10, and 11.

• **Frontend:** HTML5, CSS3, Bootstrap 5, Tailwind CSS, JavaScript (ES6+).

• **Backend:** Python 3.11.9 with Flask micro-framework.

• **Database:** SQLAlchemy ORM with SQLite or any supported RDBMS.

Supported Web Browsers:

Browser	Version / Compatibility
Google Chrome	v97 and above (Windows/macOS)
Mozilla Firefox	v95 and above (Windows/macOS)
Safari	v15.0 and above (macOS)
Microsoft Edge	Chromium-based Edge

The platform uses **RESTful routing**, AJAX calls for dynamic content loading, and complies with standard web accessibility practices (ARIA labels, keyboard navigation, and color contrast).

2.3.5. Communication Interface:

The system enables direct and efficient communication between users and platform administrators through the following integrated channels:

• **In-app Announcements Panel:** Visible from user dashboards with real-time updates.

- **Email Notifications:** Sent via SMTP to inform users of changes, confirmations, and updates.

- **Contact Us Page:** Enables feedback, complaints, and general inquiries.

- **OTP-based Password Reset System:** For secure password recovery.

Security and Privacy Protocols:

- Passwords are encrypted using Flask-Bcrypt before being stored

- Email recovery is secured using OTP verification sent to the registered email address

- HTTPS is enforced to secure all client-server communication

- Minimal personal data is stored (only necessary fields for operation)

- Access to data is permission-based, with separate privileges for users and admins.

Figure: 2.3.5 Communication Interface

2.4.   Other Non-Functional Requirements:

2.4.1. Performance Requirements:

• **PR-1:** The system is engineered to deliver **fast, reliable AI-powered services** such as resume gap analysis, mock interviews, and recommendation generation. Low latency in response time is critical to delivering a smooth user experience and increasing user engagement.

• **PR-2:** The **Graphical User Interface (GUI)** is developed with responsiveness in mind. It supports all modern browsers and screen resolutions, ensuring seamless accessibility across desktops, laptops, tablets, and mobile phones.

• **PR-3:** The platform is **user-centered**, offering intuitive workflows with clearly labeled buttons, tooltips, real-time feedback, and help text that assist users in navigating the platform— even if they are interacting with it for the first time.

- **PR-4:** Uploading documents, viewing analysis, and accessing history are optimized to **process less than 2 seconds** in average load conditions, ensuring uninterrupted task flows.

2.4.2. Safety Requirements:

The platform follows multiple safety protocols to ensure that users' personal and career- related data remains secure.

- **SR-1:** The platform uses **SSL (Secure Socket Layer) certificates** to ensure encrypted, end-to-end data transmission.

- **SR-2:** All APIs, including those handling resumes, job descriptions, and account credentials, are served over **HTTPS using secure transport protocols**.

- **SR-3:** The platform continuously **monitors data storage and server logs** to detect anomalies, corruption, or potential data loss.

- **SR-4:** A **multi-level access control** system is enforced. Users and admins are separated by permissions, and admin-level controls are isolated and protected.

- **SR-5: Scheduled maintenance** windows are implemented for system updates and patching. Users are notified in advance to reduce disruption.

- **SR-6: User data is protected by session-based login access**, ensuring only authenticated users can view or manipulate their data.

- **SR-7:** Only authorized administrators have access to the **backend and database systems**, which are protected by password policies, 2FA (optional), and audit logging.

2.4.3. Security Requirements:

- **SEC-1:** The system employs **multi-layered security protocols** including authentication, authorization, and input validation to prevent common attack vectors such as SQL injection, XSS, and brute-force attempts.

- **SEC-2:** The server is **SSL-certified**, ensuring that all communications between clients and the server are encrypted and immune to packet sniffing and man-in-the-middle attacks.

- **SEC-3:** Passwords are stored securely using **hashed and salted encryption** methods via Flask-Bcrypt.

- **SEC-4:** A **secure OTP-based password reset mechanism** is implemented for account recovery.

2.4.4. Software Quality Attributes:

A good system's software should possess good qualities of software attributes. These attributes talk about how well the system is complies with and conforms with the designed functional requirements. These attributes lead the software to meet the markets and the user demands.

To meet professional standards and gain user confidence, the AI Career Assistant system is built to uphold essential software quality attributes:

Following is the Software Quality Attributes:

2.4.4.1 Availability:

- The platform is designed to stay **available and responsive at all times**.

- In case of downtime or a technical issue, the system will notify users and the **maintenance team will act immediately** to restore access.

- Reliability, uptime, and data confidentiality are prioritized.

2.4.4.2 Performance:

- The system is developed using **asynchronous programming techniques**, allowing users to perform multiple actions simultaneously — like uploading documents while reviewing reports.

- Performance testing ensures the system responds swiftly even under high load.

2.4.4.3 Testability:

All core features go through **unit, integration, and user testing** before deployment.

Testing ensures:

- High software quality

- Functional and non-functional requirement verification

- Reliable and secure performance

- Accurate gap analysis and AI interview logic

2.4.4.4 Security

Security is a top priority for user data such as CVs, job descriptions, and personal info.

The system uses:

- **SSL certification**

- **User-level authentication**

- **Admin-only access** to sensitive operations

- **Routine monitoring** for threats or anomalies

2.4.4.5 Usability:

- The user interface is built with **simplicity and clarity** in mind.

- All pages and components are designed to be **intuitive**, so that users can access features without needing prior technical knowledge.

- **Tooltips, progress indicators, and form guidance** are used to improve ease of use and reduce confusion.

CHAPTER 3
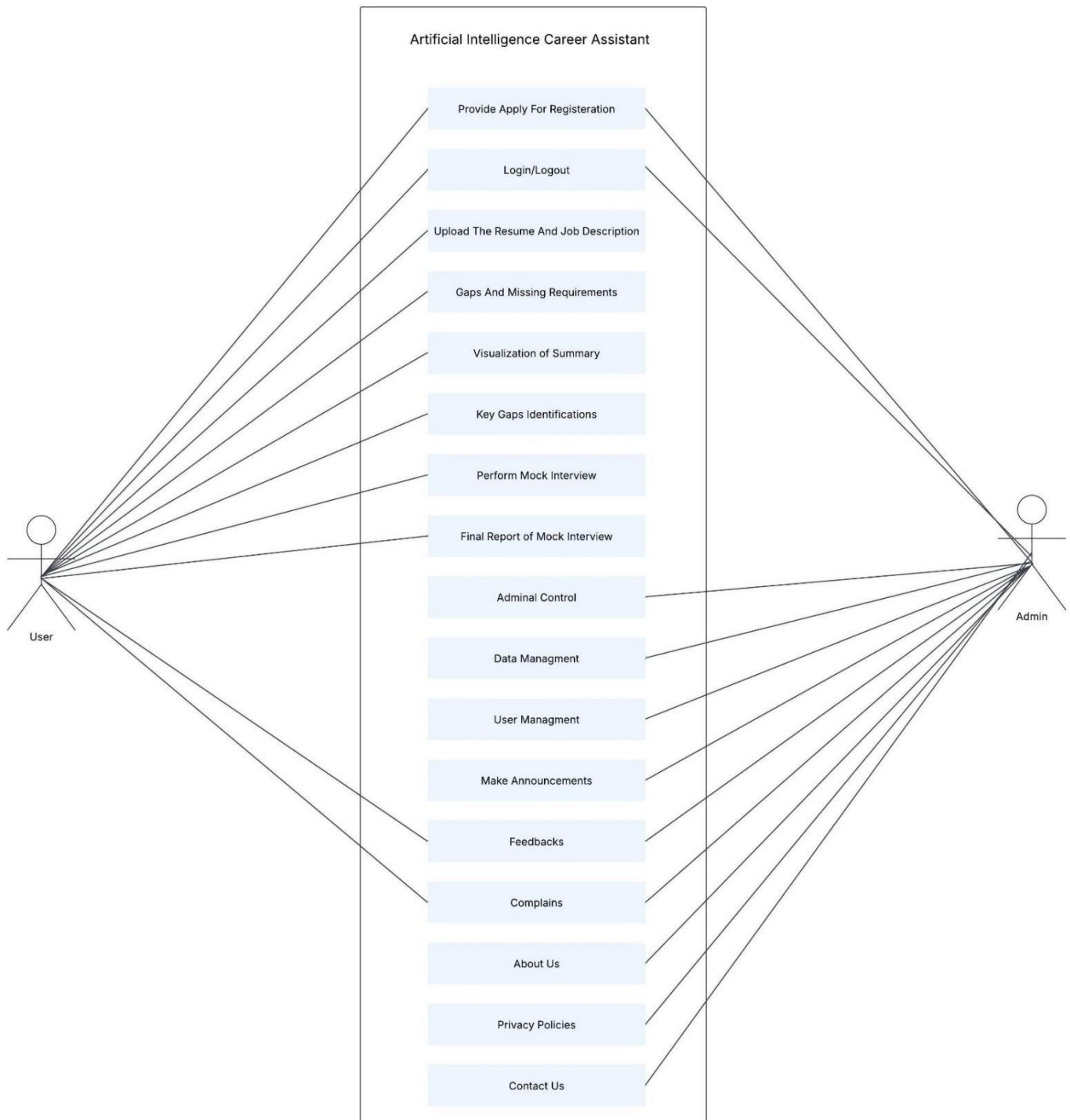# USE CASE ANALYSIS
3.1. Use Case Model:

Figure 3.1: Use cases

3.2. Fully Dressed Use Cases

Use case: 1

Number	001
Name	User Registration.
Summary	A job seeker signs up on the platform to access career-enhancing tools like resume analysis and mock interviews.
Priority	High.
Pre-condition	The user must not already have an existing account and must use a valid, unique email address.
Post-condition	A new user account is created, and the user is redirected to their dashboard. Confirmation is displayed.
Primary	Job Seeker (User)
Secondary	SQLAlchemy Database, Flask Backend
Trigger	The user clicks the "Sign Up / Register" button from the home or login page.
Main scenario	Step action: 1. The system displays the homepage with login and registration options. 2. The user clicks the **Signup** button. 3. The registration form appears requiring fields: First Name, Last Name, Email, Password, User Role. 4. The user fills in all the required fields and clicks **Submit**. 5. The system validates the data (e.g., password strength, email uniqueness). 6. If valid, the password is hashed and stored, and the account is created in the database. 7. The user is redirected to the login page or dashboard with a success message: *"Registration successful. You can now log in."*
Extension	Steps: 1. Missing or Invalid Input. 2. Email Already Registered. 3. Password Too Weak.

Table 3.1: Registration

Use case: 2

Number	002
Name	User/Admin Login
Summary	Authenticates registered users (Job Seekers or Admin) to access personalized dashboard admin panel.
Priority	High.
Pre-condition	User must be registered with a valid email and password.
Post-condition	If login is valid, the user is redirected to their respective dashboard.
Primary	User / Admin
Secondary	SQLAlchemy Database, Flask Auth
Trigger	The user clicks on "Login".
Main scenario	Step action: 1. The system displays the login screen with two role options. 2. The user selects their role and fills in their email and password. 3. The system verifies credentials by querying the database. 4. User are redirected to their dashboard. 5. A welcome message or dashboard is shown.
Extension	Steps: 1. Incorrect Credentials. 2. Missing Fields. 3. Forgotten Password.

Table 3.2: Login

Use case: 3

Number	003
Name	Resume and Job Description Upload.
Summary	Allows users to upload their resume and paste a job description for analysis.
Priority	High.
Pre-condition	User must be logged in.
Post-condition	Files and job descriptions are saved and passed to AI analysis pipeline.
Primary	Job Seeker.
Secondary	Server, LLM AI Model
Trigger	The user clicks "Match Resume" and uploads required data.
Main scenario	Step action: 1. User logs in and lands on dashboard. 2. Clicks **Upload Resume**. 3. Uploads a supported document (.pdf, .docx, .txt). 4. Pastes the job description in a provided text box. 5. Clicks Submit. 6. The system saves the files and sends them for AI processing. 7. A loading screen is shown, followed by the results page.
Extension	Step action: 1. Unsupported File Format: • The system alerts: "Only PDF, DOCX or TXT files are supported." 2. Empty Job Description: • User is prompted to enter a valid description.

Table 3.2: Resume and JD Uploading

Use case: 4

Number	004
Name	Resume Gap Analysis.
Summary	Compares user resume with job description to identify skill/experience gaps.
Priority	High.
Pre-condition	Resume and Job Description must be uploaded.
Post-condition	A detailed gap summary and visual representation is shown.
Primary	User.
Secondary	AI Pipeline (LangChain, LLM), Database.
Trigger	Resume + JD submitted from dashboard.

Main scenario	Step action:
	1. User uploads resume and JD.
	2. The AI analyzes and returns:
	    • Fit Summary.
	    • Gaps in technical, experience, domain, soft skills.

Table 3.4: Resume Gap Analysis

Use case: 5

Number	005
Name	Mock Interview Simulation.
Summary	Users practice mock interviews via AI with contextual and behavioral questions.
Priority	Medium-High
Pre-condition	Resume must be uploaded to tailor interview questions.
Post-condition	Feedback and suggestions are shown after each interaction.
Primary	User.
Secondary	AI Model Pipeline, Flask Server
Trigger	User clicks **Mock Interview** from dashboard.
Main scenario	Step action: 1. The system loads a chat-like interface. 2. AI begins with general questions and escalates based on responses. 3. User answers via text box. 4. AI Model provides feedback after each question. 5. At end, user gets a summary of performance.

Table 3.5: Mock Interview

Use case: 6

Number	006
Name	Admin Announcement Posting.
Summary	Admin can create, update, or delete announcements visible to all users.
Priority	Medium.
Pre-condition	Admin must be authenticated.
Post-condition	Announcements saved in DB and visible to users.
Primary	Admin.
Secondary	Database.
Trigger	Admin navigates to **Announcements** section.
Main scenario	Step action: 1. Admin logs in and goes to **Announcements Panel**. 2. Clicks **Create New** and types the message. 3. Clicks Submit → announcement is saved in DB. 4. Users see the message on their dashboards.

Table 3.6: Admin Panel

CHAPTER 4

SYSTEM DESIGN

Design

In this section, the overall architecture of the **AI Career Assistant** system is described in detail. It outlines how the system was designed to fulfill all functional and non-functional requirements efficiently and reliably. The design includes the system's architecture, user flow, backend logic, data management, and UML-based diagrams to visually represent each component and process.

System design is the process of defining the architecture, modules, components, interfaces, and data that make up the complete software system. It begins once the requirement analysis is complete and aims to convert high-level specifications into detailed, implementable blueprints.

The goal is to build a robust, scalable, secure, and user-friendly system for intelligent career guidance and resume enhancement powered by AI.

The AI Career Assistant follows a modular design with clean separation of frontend, backend, AI engine, and database layers. The system is built to support intuitive user interactions, high responsiveness, strong security practices, and maintainability.

Before implementing any module, the entire system is visualized using diagrams to simplify development, identify relationships between components, and ensure clarity among developers.

This chapter includes:

- Work Breakdown Structure (WBS).

- Entity Relationship Diagram (ERD).

- Class Diagram.

- Object Diagram.

- Sequence Diagram.

- Activity Diagram.

- State Transition Diagram.

- Collaboration Diagram.

4.1. Work Break Down System

WBS stands for Work Breakdown Structure. Work breakdown structure (WBS) is the hierarchal decomposition of the total scope of work for the project. It enables the development

team to manage and deliver module in a step-by-step manner.

- User Authentication Module.

- Resume Upload and Parsing Module.

- Job Description Handling Module.

- AI Resume Gap Analysis Engine.

- Mock Interview Chat System.

- Announcement System (Admin).

- Admin Panel for User and Data Management.

- Database Integration and Security Layer.

- Frontend UI Design.

Each team member is assigned specific components based on their expertise. The structure ensures all aspects of the platform are completed with coordination and within project timelines.
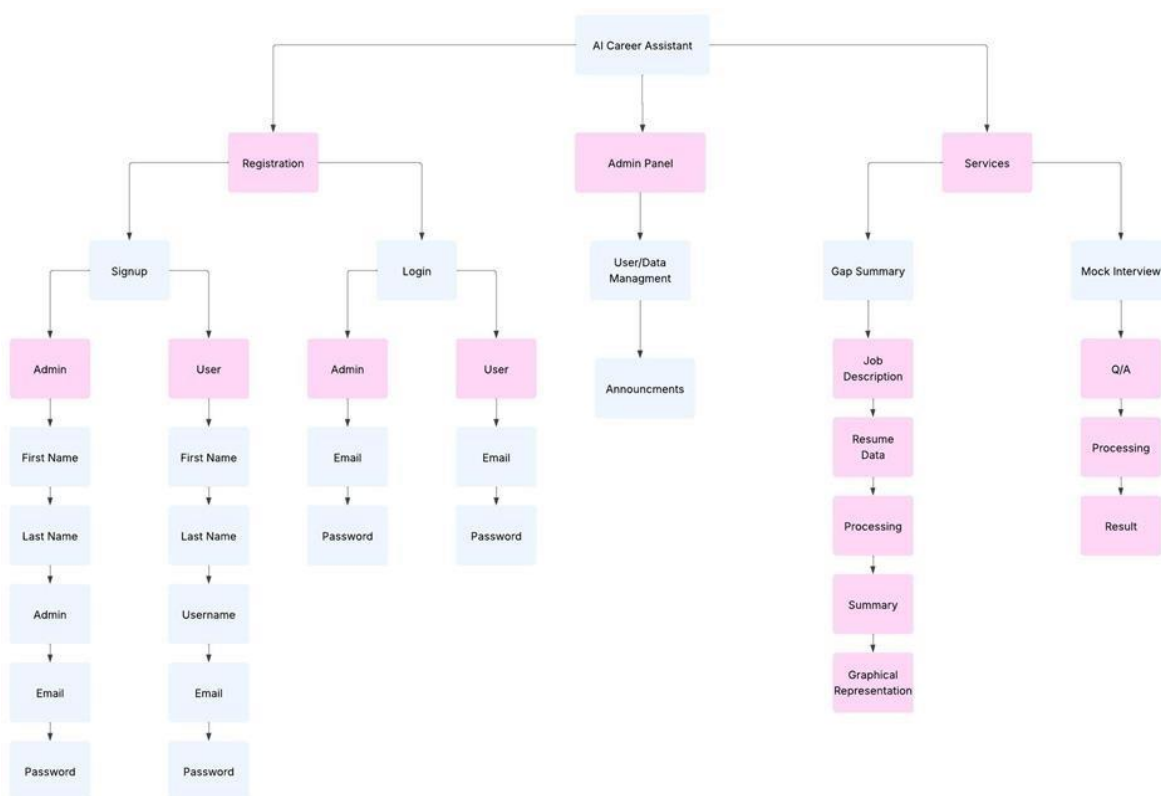
Figure 4.1: Work Breakdown System

4.2. ERD

ERD stands for Entity Relationship Diagram. An entity relationship diagram describes interrelated things of interest in a specific domain of knowledge and they have important role in database. A basic ER model is composed of entity types and specifies relationships that can exist between entities.

Following ERD shows the working of our project "AI Career Assistant". We have multiple entities such as:

· **User** (with attributes like ID, Name, Email, Password).

· **Resume Data** (uploaded document content).

· **Job Description** (entered by user).

· **Gap Report** (generated by AI).

· **Announcements** (created by Admin).

The relationship between entities such as a **User having multiple Data entries** (one- to-many) or **Announcements being globally accessible** are modeled clearly using cardinality and modality.

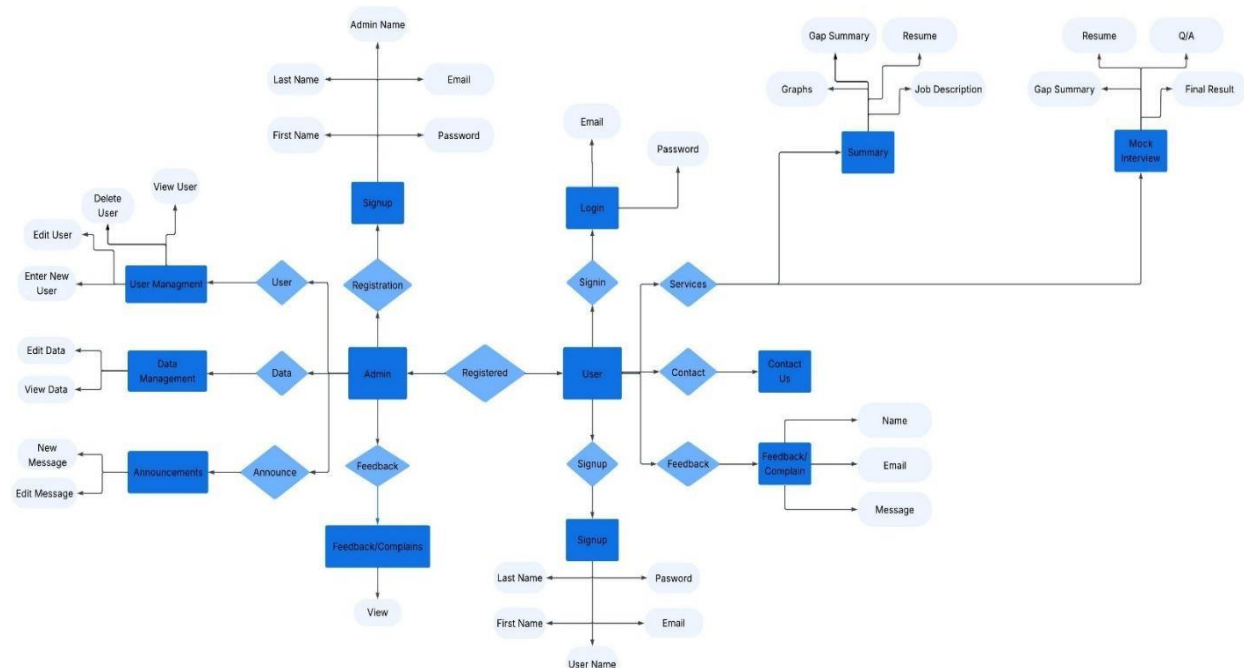The ERD ensures structured data storage and retrieval through SQLAlchemy ORM.

Figure 4.2: Entity Relation Diagram

4.3. Class Diagram

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity.

- **User class** includes methods like set_password() and check_password().

- **Data class** includes attributes for storing resumes, job descriptions, and summary reports.

- **Admin class** is implicitly part of the user model with elevated privileges.

- **LLM/AI module** interacts with Data class for analysis.

Following figure illustrates our project working with classes that have their data members and member functions. Every class performs its working correctly and according to entities of ERD diagram. Every arrow has unique purpose such as it shows strong and weak relationship between classes etc.
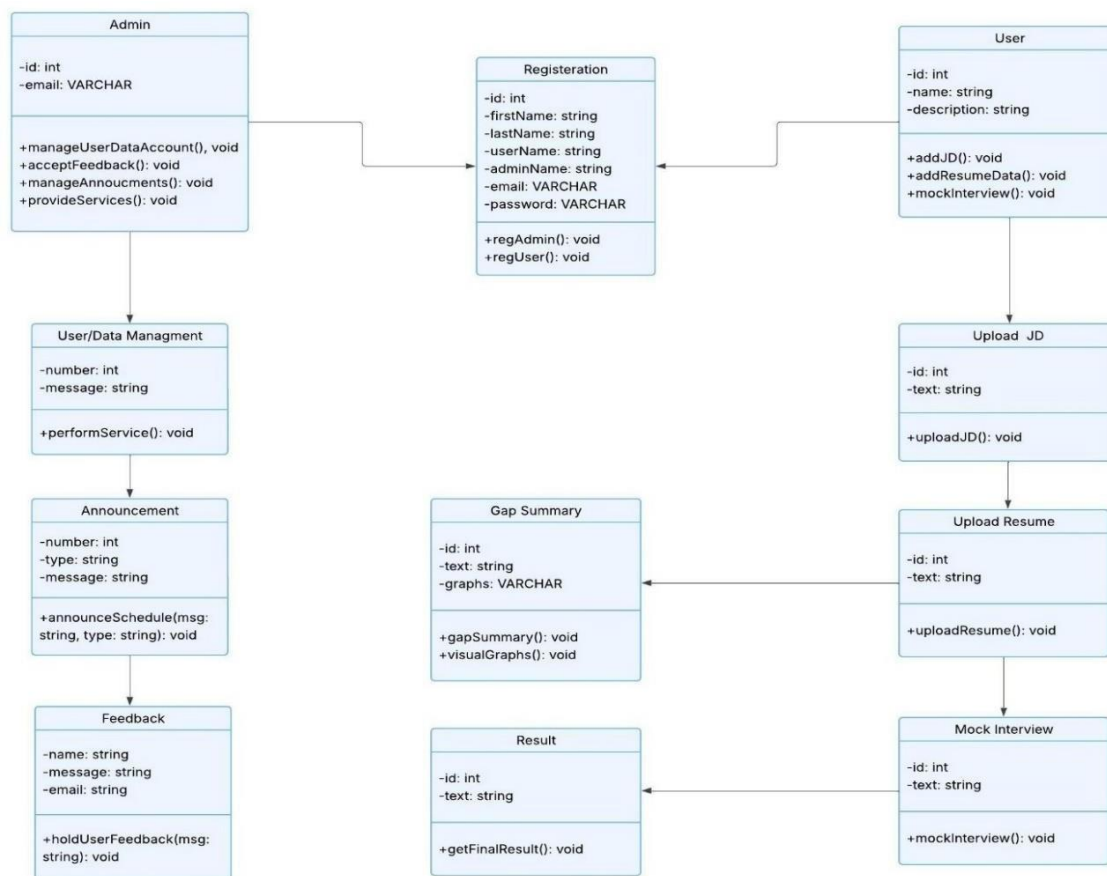

Figure 4.3: Class Diagram

4.4. Object Diagram

A UML object diagram represents a specific instance of a class diagram at a certain moment in time. When represented visually, you'll see many similarities to the class diagram. An object diagram focuses on the attributes of a set of objects and how those objects relate to each other. In other words, object is the real time representation of class diagram.

- An instance of User(id=1) may have multiple Data objects linked.

- An instance of Data(id=12) may store resume text, job description, and AI feedback.

- Announcements are linked to the admin context but accessible globally.

Following figure illustrates our project working with object that represents the real-time data of classes and how they actually used.
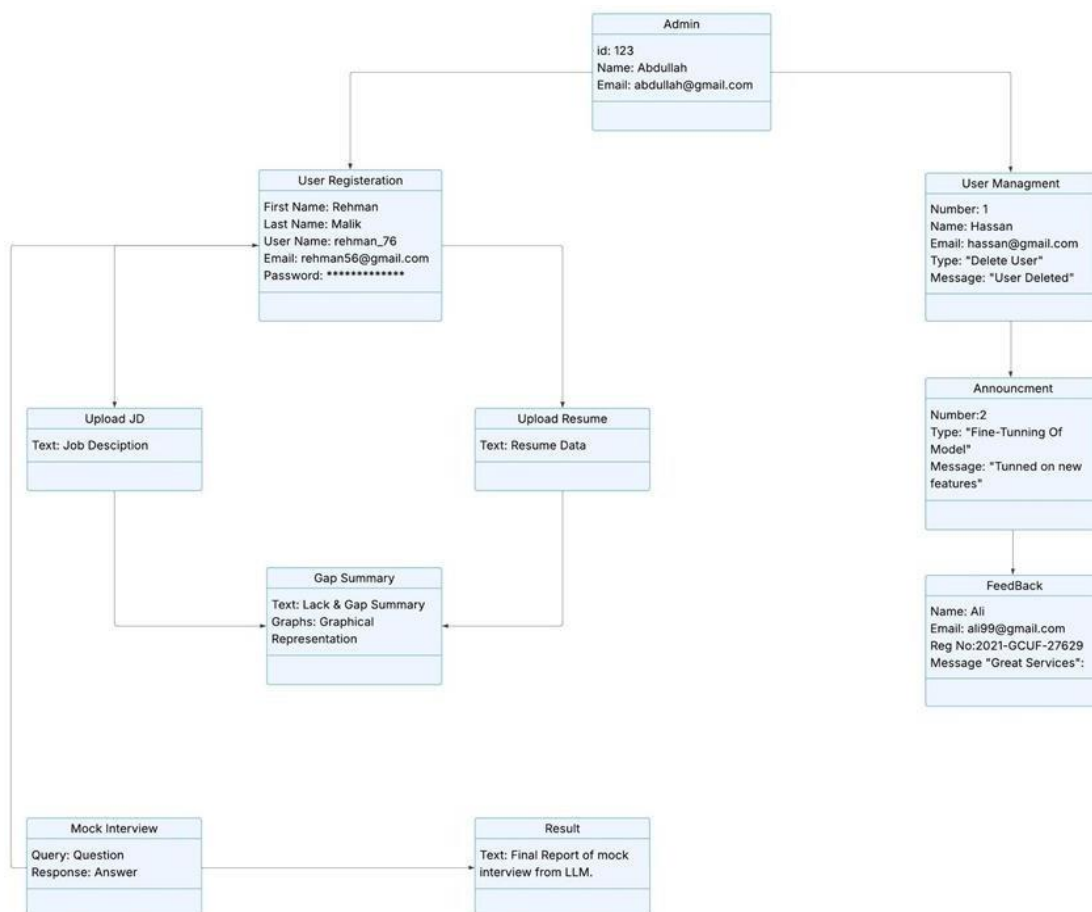
Figure 4.4: Object Diagram

4.5. Sequence Diagram

A sequence diagram is a Unified Modeling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction.

• User → Uploads Resume + Job Description.

• System → Extracts text.

• System → Sends data to AI Engine (LLM).

• AI → Returns Gap Analysis.

• System → Stores result in database.

• System → Displays result + chart to user.

A sequence diagram is structured in such a way that it represents a timeline that begins at the top and descends gradually to mark the sequence of interactions. Each object has a column and the messages exchanged between them are represented by arrows.

The following figure of sequence diagram according to our project illustrates the working criteria and timelines of actor. It shows which actor respond which object and how they interact during their lifeline.
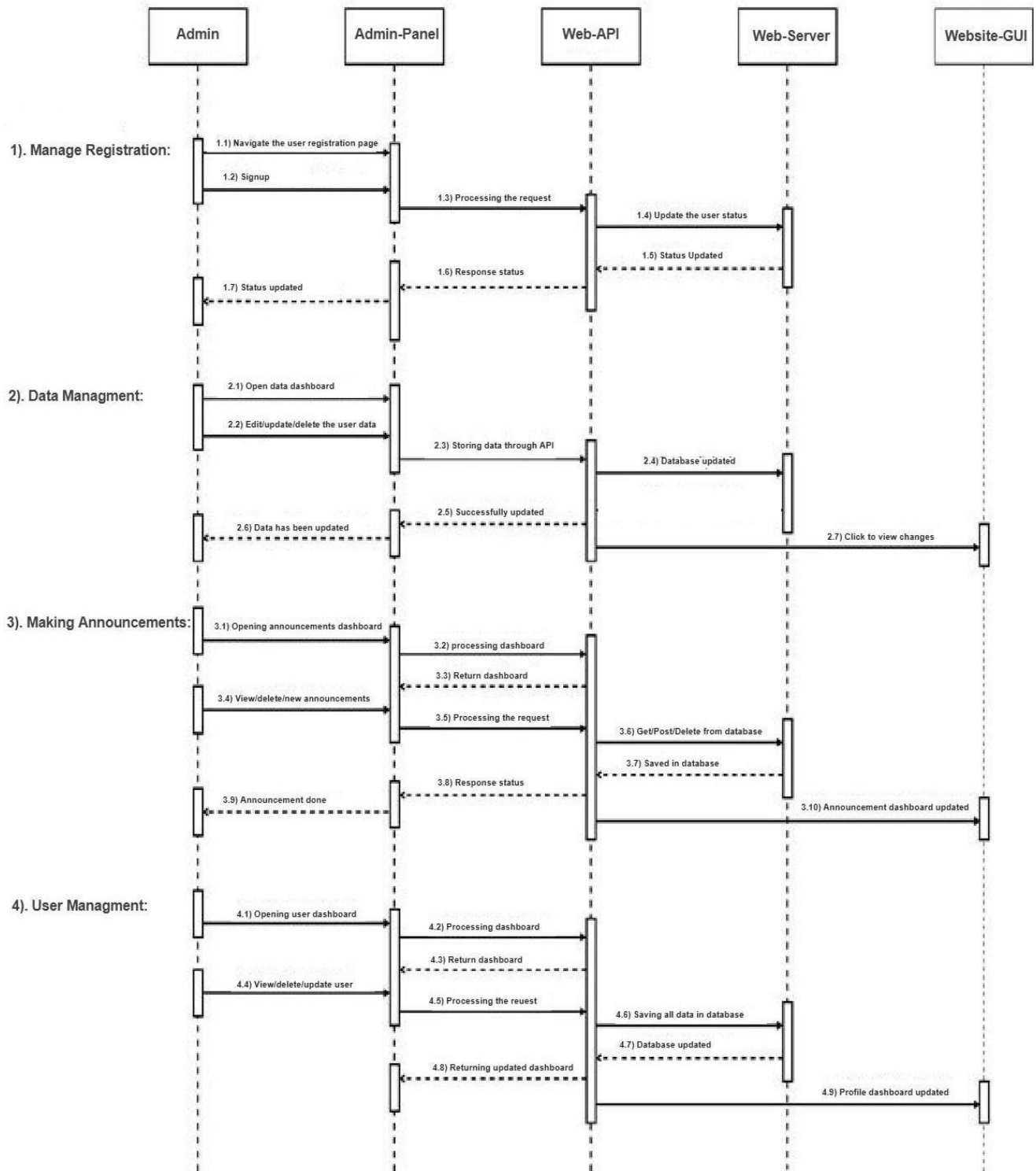
Sequence Diagram of Admin

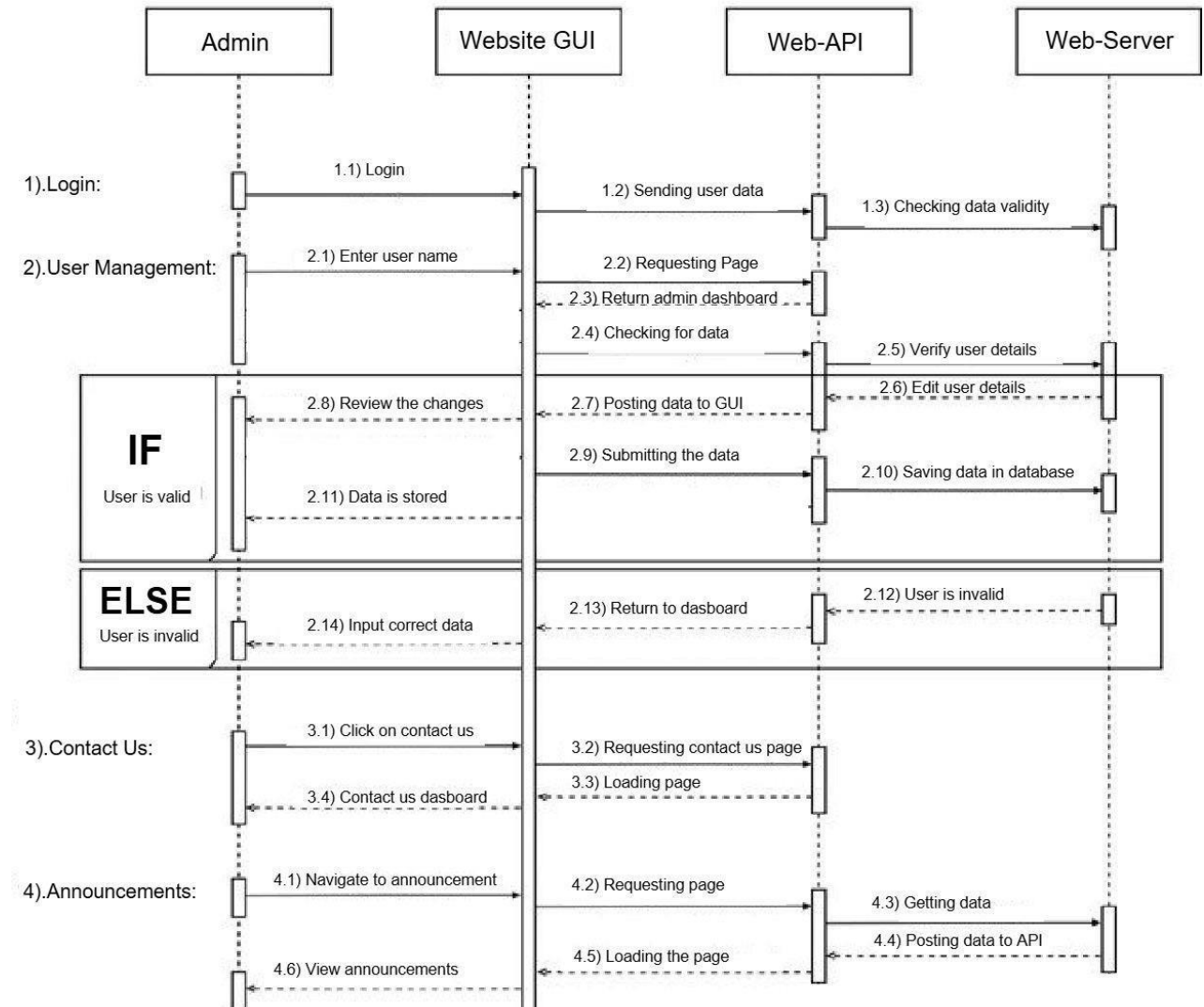**Figure 4.5.1.1 Sequence Diagram of Admin**

Sequence Diagram of Admin

Figure 4.5.1.2 Sequence Diagram of Admin

Sequence Diagram of User
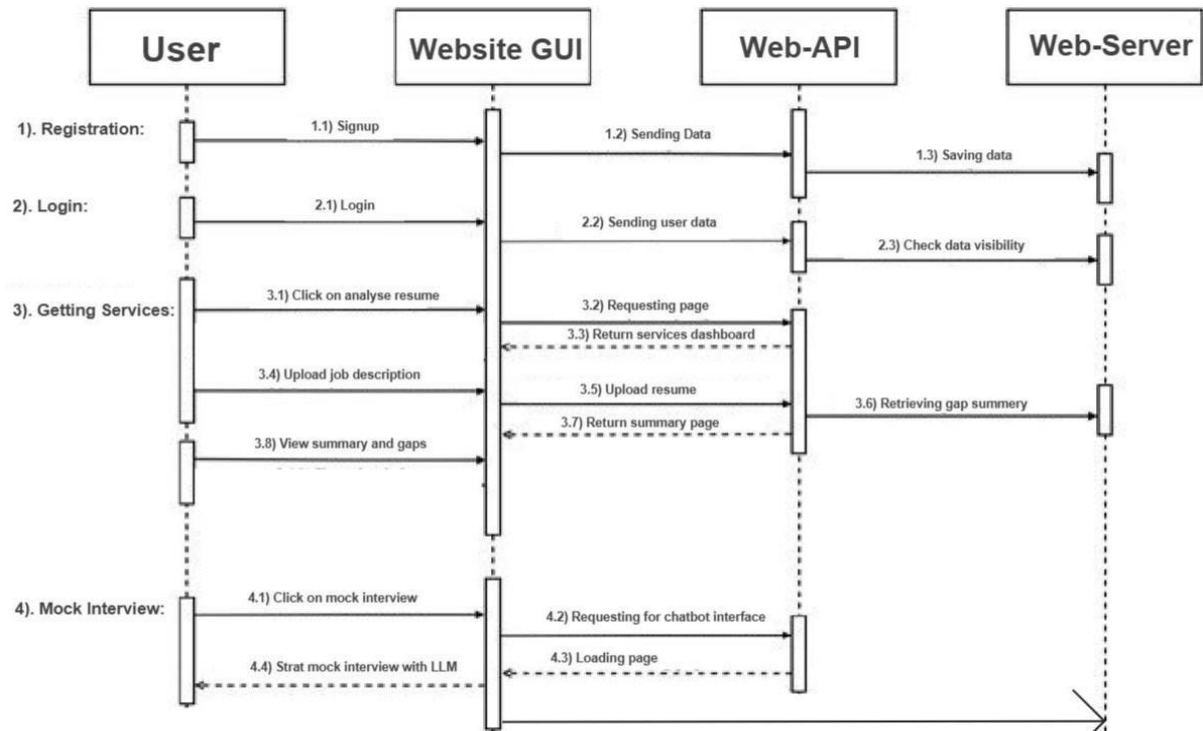
Figure 4.5.2 Sequence Diagram of User

4.6. Activity Diagram

Activity diagram is another *important behavioral diagram in UML diagram to describe dynamic aspects of the system*. Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram is used by developers to understand the flow of programs on a high level. It also enables them to figure out constraints and conditions that cause particular events.

•   **User Activity Diagram**: Register → Login → Upload Resume → View Report → Logout.

•   **Admin Activity Diagram**: Login → View Users → Manage Announcements → View Logs.

•   **System Activity Diagram**: Validate Inputs → Parse Files → Run AI Analysis → Return Results.

We have to make three activity diagrams for showing the dynamic behavior of our project i.e.; activity diagram of coach, player and admin. The following diagram shows the behavior of the actors and how they interact with entities

Activity Diagram of User
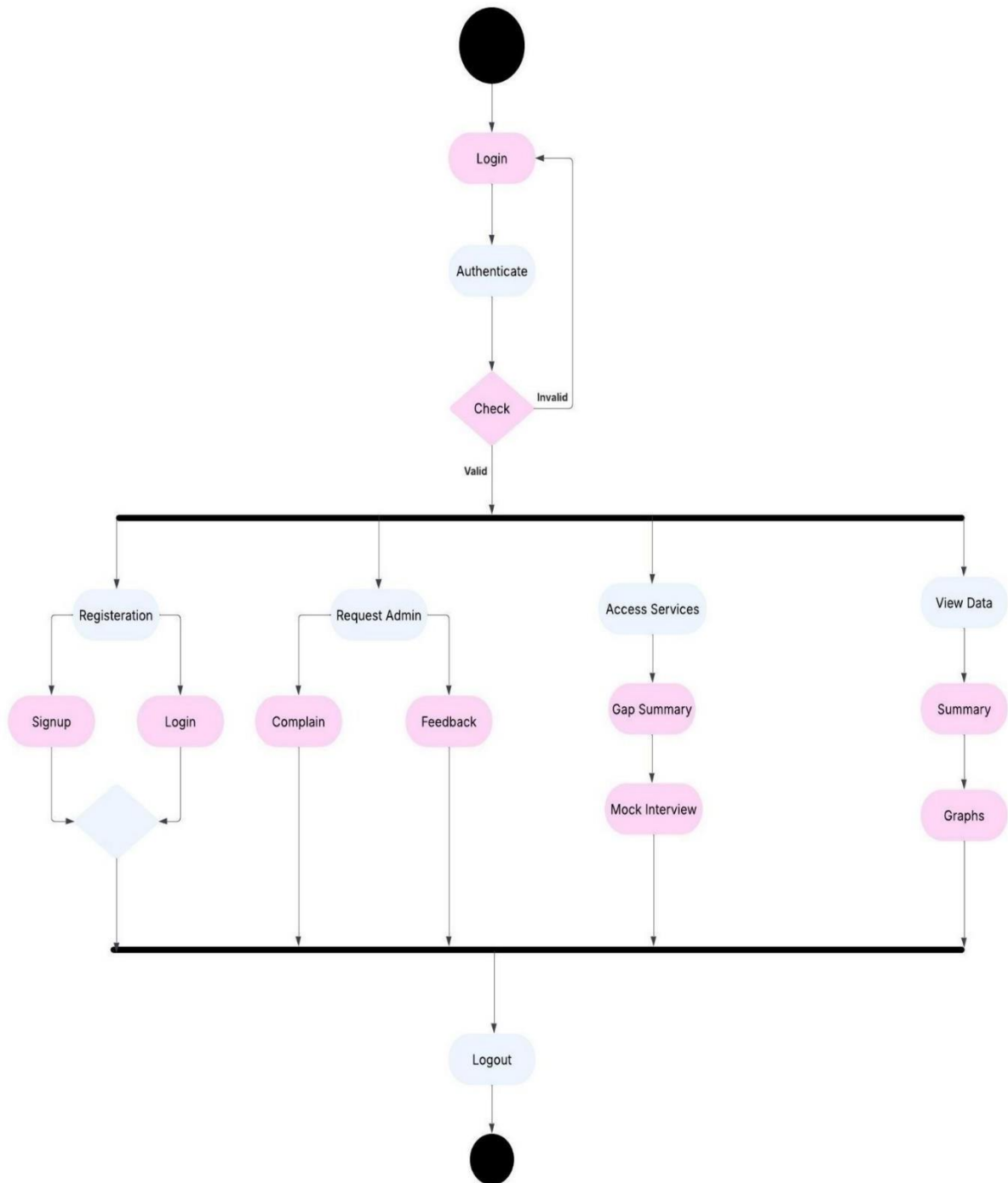
Figure 4.6.1 Activity Diagram of User

Activity Diagram of Admin
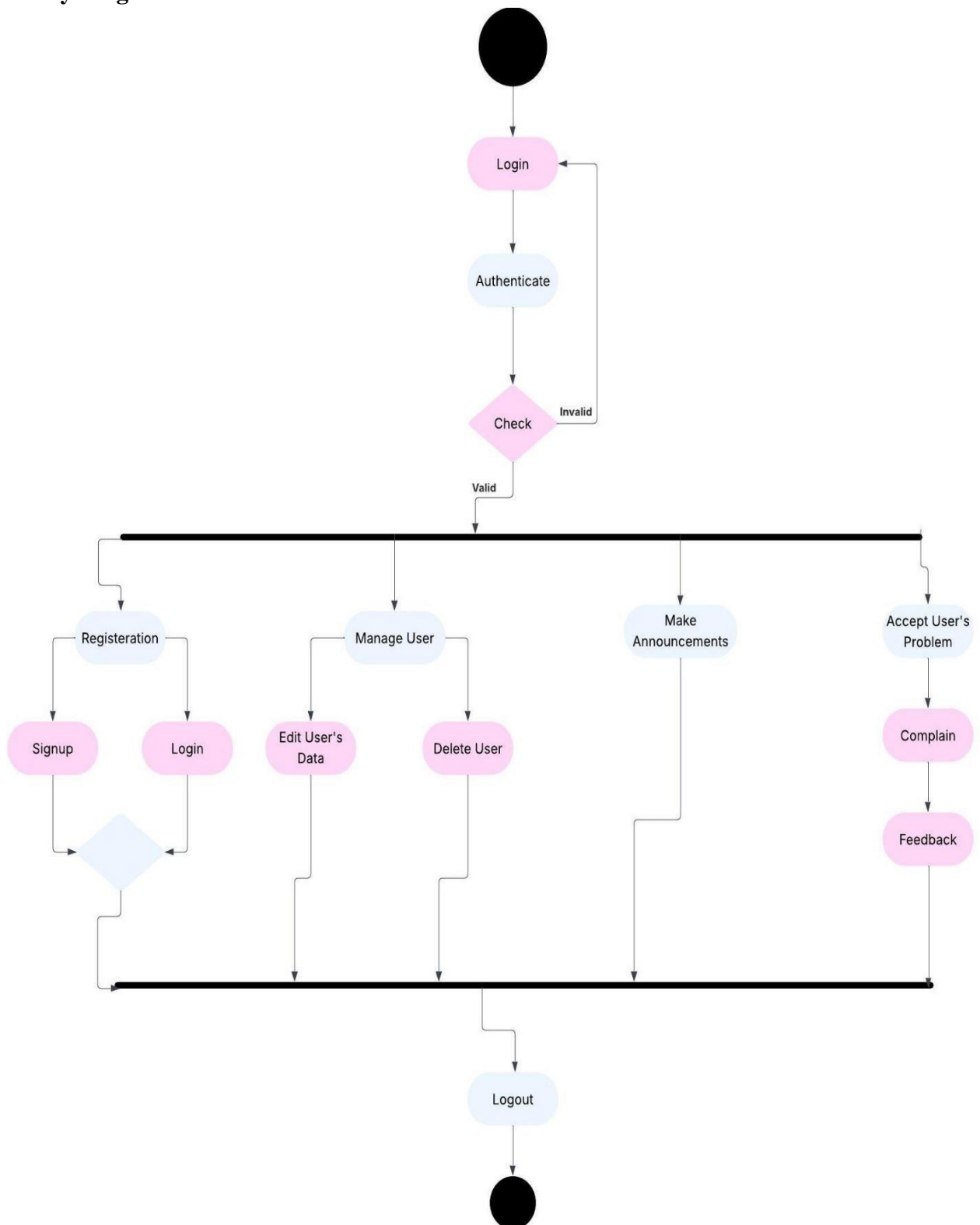
Figure 4.6.2 Activity Diagram of User

4.7. State Transition Diagram

A state-transition diagram is a type of diagram used in computer science and related fields to describe the behavior of systems.

A state-transition diagram, also known as a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language (UML).

State-transition diagrams describe all of the states that an object can have, the events under which an object changes state (transitions), the conditions that must be fulfilled before the transition will occur (guards), and the activities undertaken during the life of an object (actions).

The following figure of state-transition diagram according to our project illustrates the working criteria of our actors such as player, admin and coach; its shows how they behave a role in a system.

Example states:

- Unauthenticated → [Login] → Authenticated.

- Authenticated → [Upload Complete] → Analyzing.

- Analyzing → [Results Ready] → Result Displayed.

- Admin → [Edit Announcement] → Publishing.

State Transition Diagram of Admin
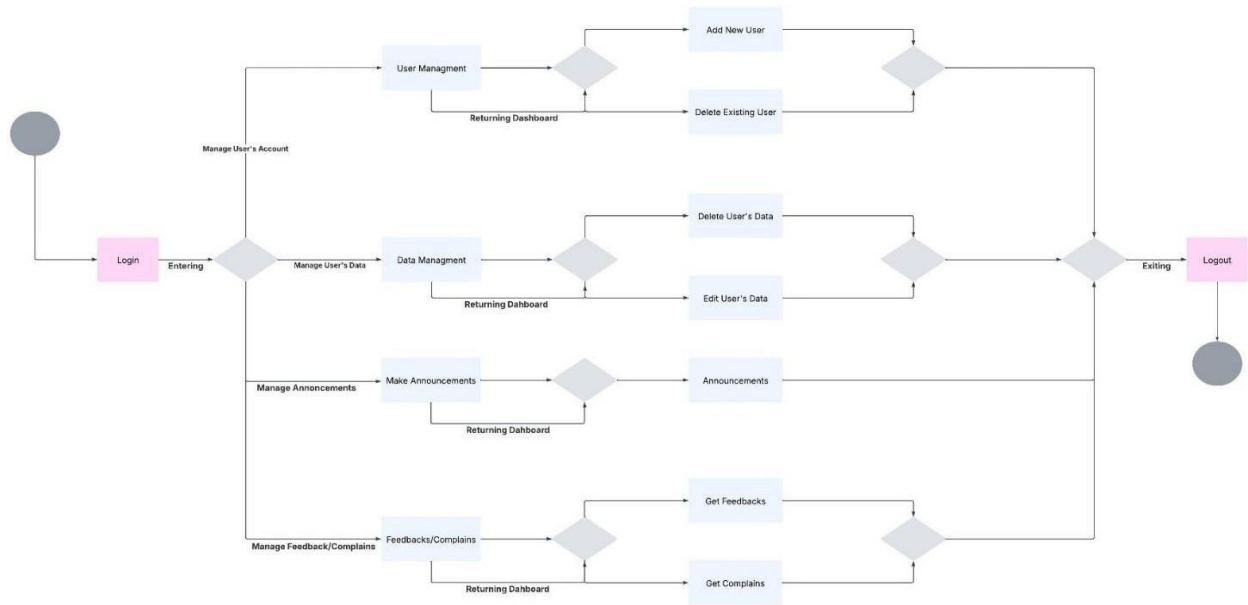
Figure 4.7.1 State Transition Diagram of Admin

State Transition Diagram of User:
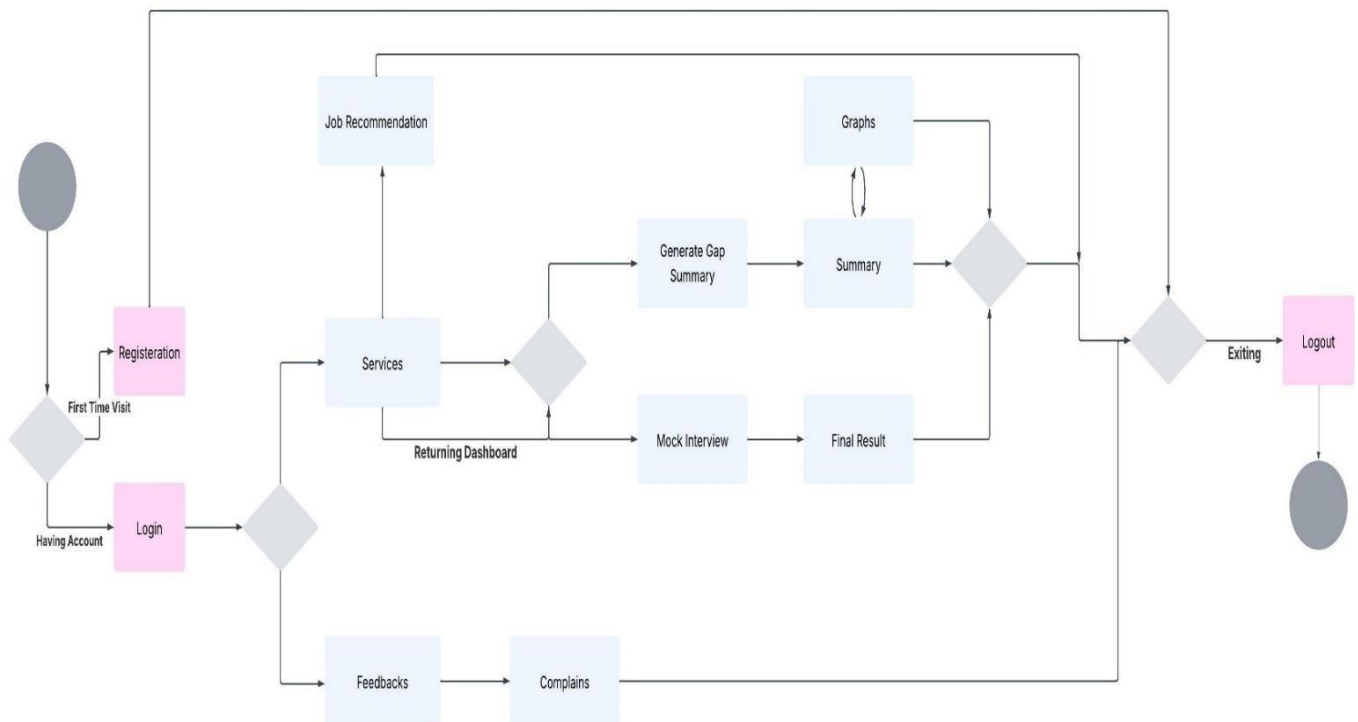
Figure 4.7.2 State Transition Diagram of User

4.8. Collaboration Diagram

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object. It contains objects, associations and message.

Following figure shows the working criteria of our project, how this diagram used with our actors (actors means users).

It determines the behavior for which the realization and implementation are specified. It also discovers the structural elements that are class roles, objects, and subsystems for performing the functionality of collaboration and think through alternative situations that may be involved.

Actors: User, Admin, AI Engine, Resume Parser, Database

- User ↔ Resume Uploader → Parser → AI Analyzer → DB Saver → Result Displayer.
- Admin ↔ Announcement Manager → DB → Email Service.

Collaboration Diagram of Admin

Figure 4.8.1: Collaboration diagram of Admin

Collaboration Diagram of User
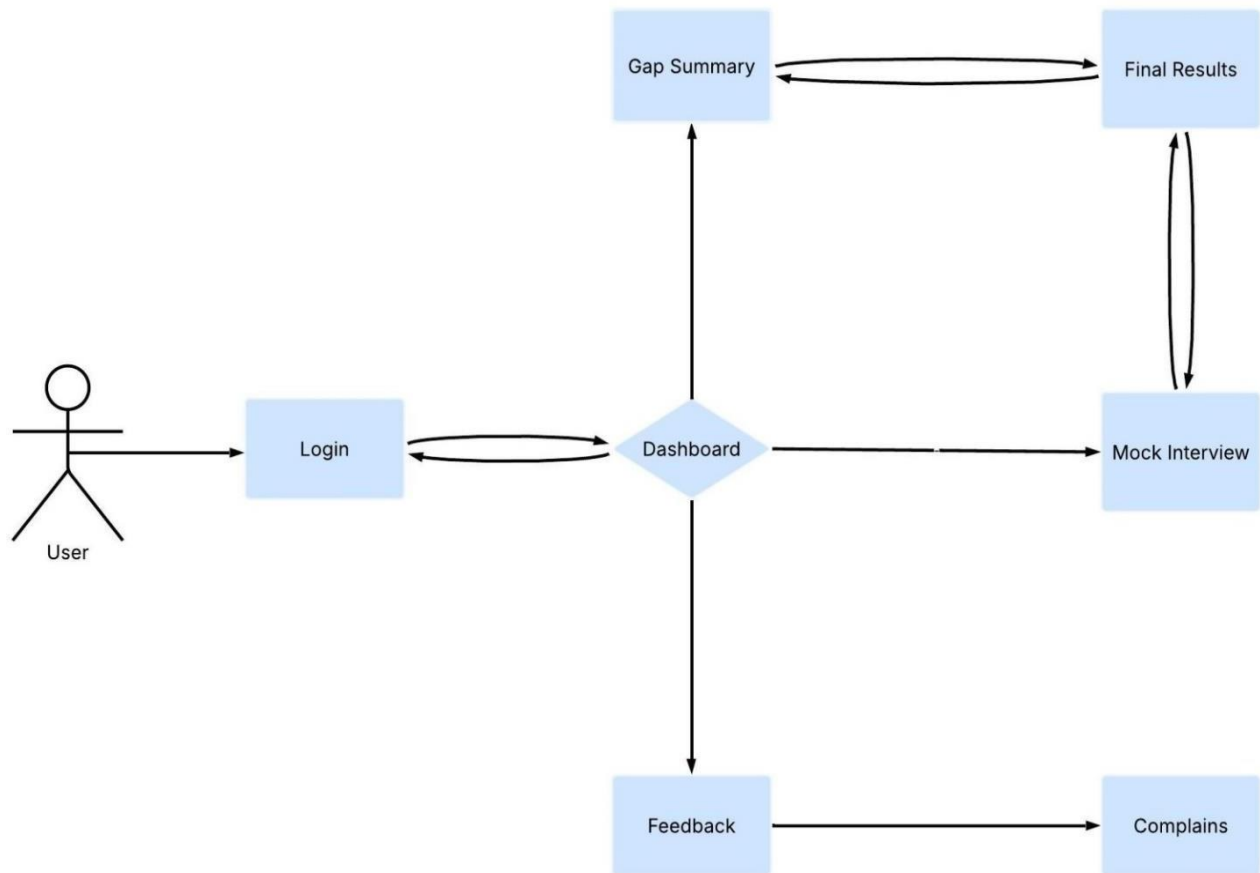
Figure 4.8.2 Collaboration diagram of User

CHAPTER 5

IMPLEMENTATION

5.1. Component Diagram

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned development. A component in UML represents a modular part of a system.
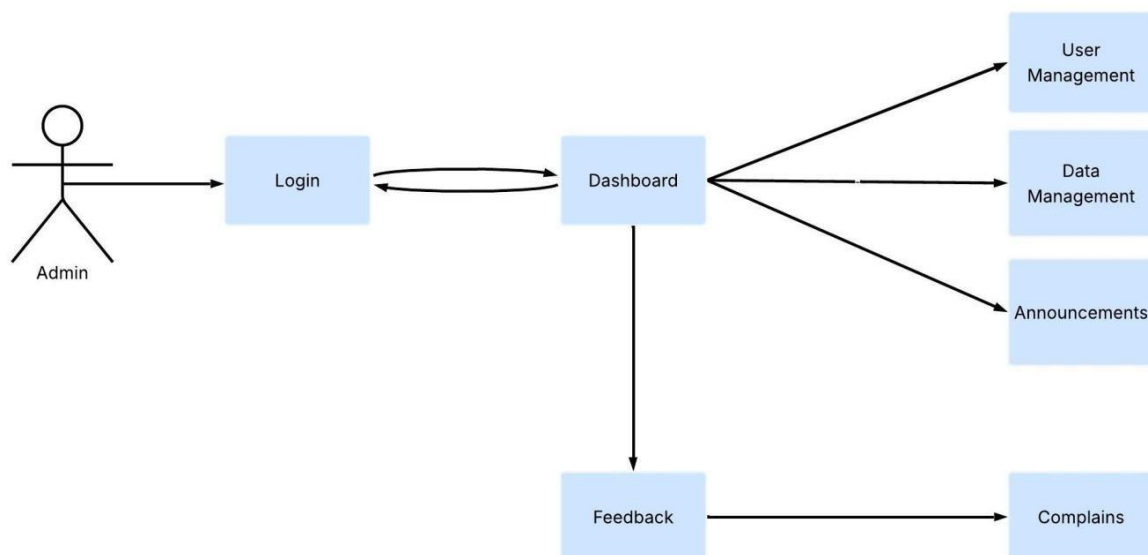
The behavior is defined in terms of required and provided interfaces. A component has an external view with public properties and operations, and it has an internal view with private properties and realizing classifiers.

Following figure shows the working of component diagram according to our project; it shows the physical aspect of our project such as database, libraries. It has 2 interface that are widely used:

Required interface and provided interface. We use dependency arrows to show the relationship between components.

In the **AI Career Assistant**, the platform is developed using a modular and scalable architecture. Each component handles a specific responsibility, ensuring separation of concerns and ease of maintenance. These components include:

• User Authentication & Session Management.

• Job Description Input Handler.

• Gap Analysis Engine (AI-Powered).

• Mock Interview Chat Module.

• Admin Control Panel.

• Email Notification Service.

• Resume and Result Storage (Database).

• Frontend UI/UX Interface.

The behavior of each component is described by its interfaces:

• **Provided Interface**: Services or APIs that the component exposes to other components (e.g., the AI Gap Analyzer provides resume feedback to the frontend).

• **Required Interface**: Services or APIs the component depends on (e.g., the resume parser requires file input and invokes AI for semantic analysis).
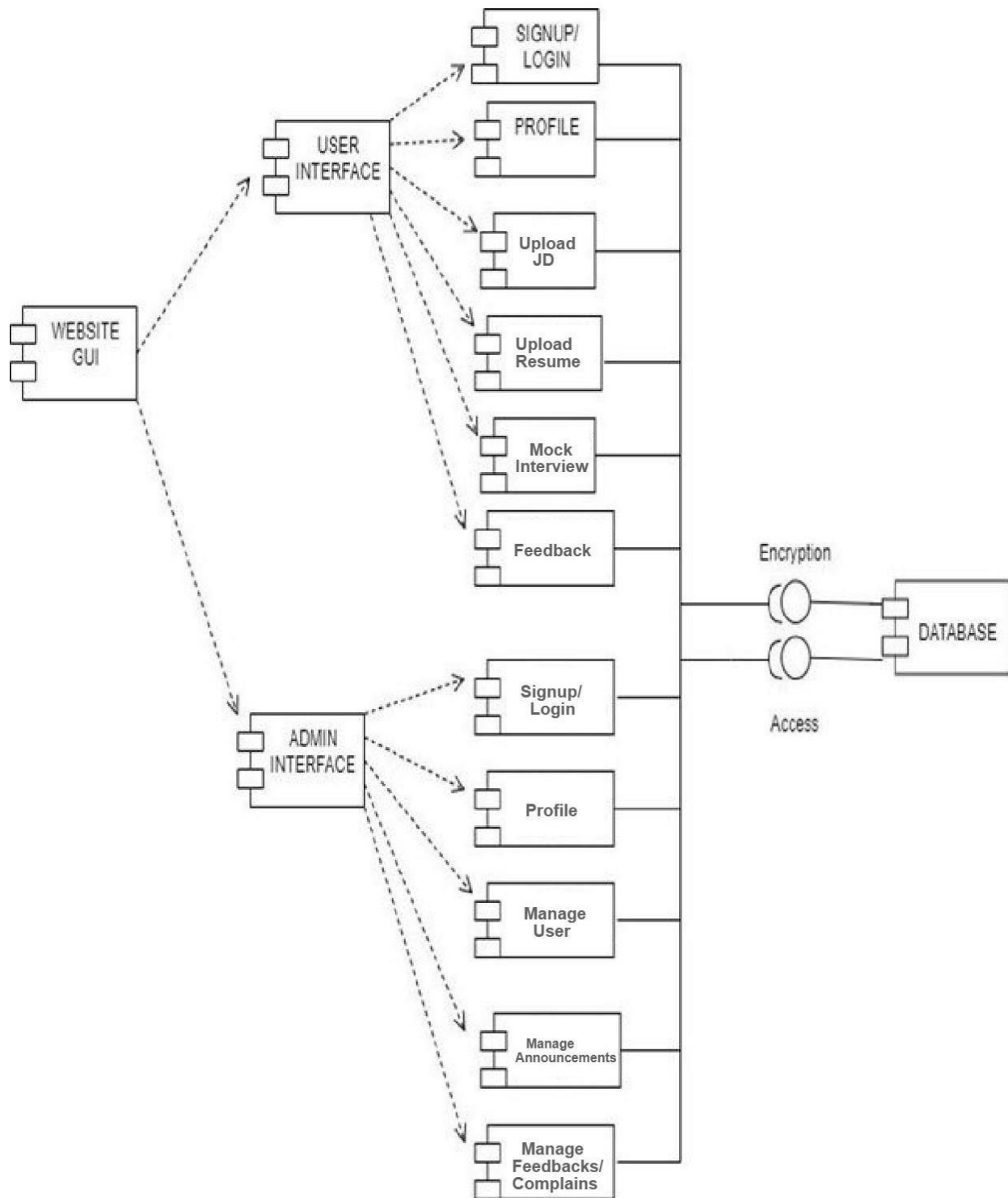
Figure 5.1: Component Diagram

5.2. Deployment Diagram

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed. Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them.

It visualizes how software interacts with the hardware to execute the complete functionality. It is used to describe software to hardware interaction and vice versa. Following figure shows the working criteria of deployment diagram according to our project; it shows static deploy view of our system.

➢ It is used to represent how a component is deployed over a node.

➢ It has 2 building block Node Artifacts and We use artifacts to deploy over nodes.

In our case, the platform is deployed as a **web-based application**, typically hosted on a cloud server with secure communication protocols. The architecture involves:

Deployment Nodes

• **Client Device (Browser):**

o User interface rendered using HTML, Tailwind CSS, Bootstrap Icons.

o Executes JavaScript for interactivity and API requests.

• **Web Server (Flask with Unicorn/UWSGI):**

o Hosts the Python backend.

o Routes HTTP requests and communicates with AI modules and DB.

• **AI Service Container:**

o Manages LangChain, Scikit-learn models for resume analysis and mock interviews.

• **Database Server (SQLite / SQLAlchemy ORM):**

o Stores user data, resumes, job descriptions, and analysis summaries.

• **Mail Server / Notification API:**

o Sends OTPs, updates, and announcements.

Artifacts:

- resume_parser.py – Extracts text from .docx, .pdf.

- ai_analysis.py – Invokes LLM for scoring and recommendations.

- user_routes.py – Manages login/signup and user sessions.

- admin_routes.py – Controls announcements, data management.

- templates/*.html – Rendered frontend pages.

- static/css, js – Styling and client-side logic.
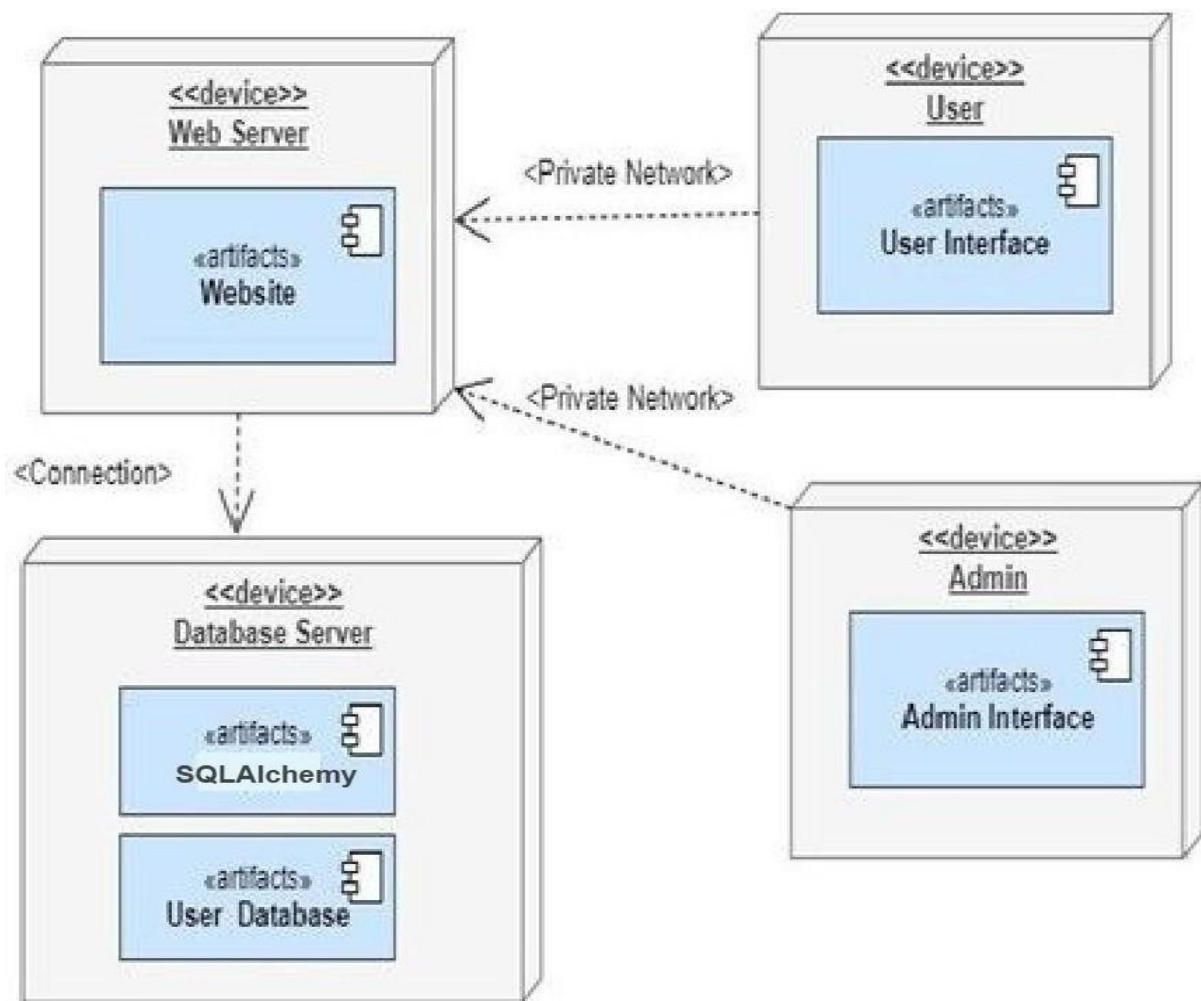
Figure 5.2: Deployment Diagram

5.3. Screenshots

5.3.1. Home Screen

Explanation
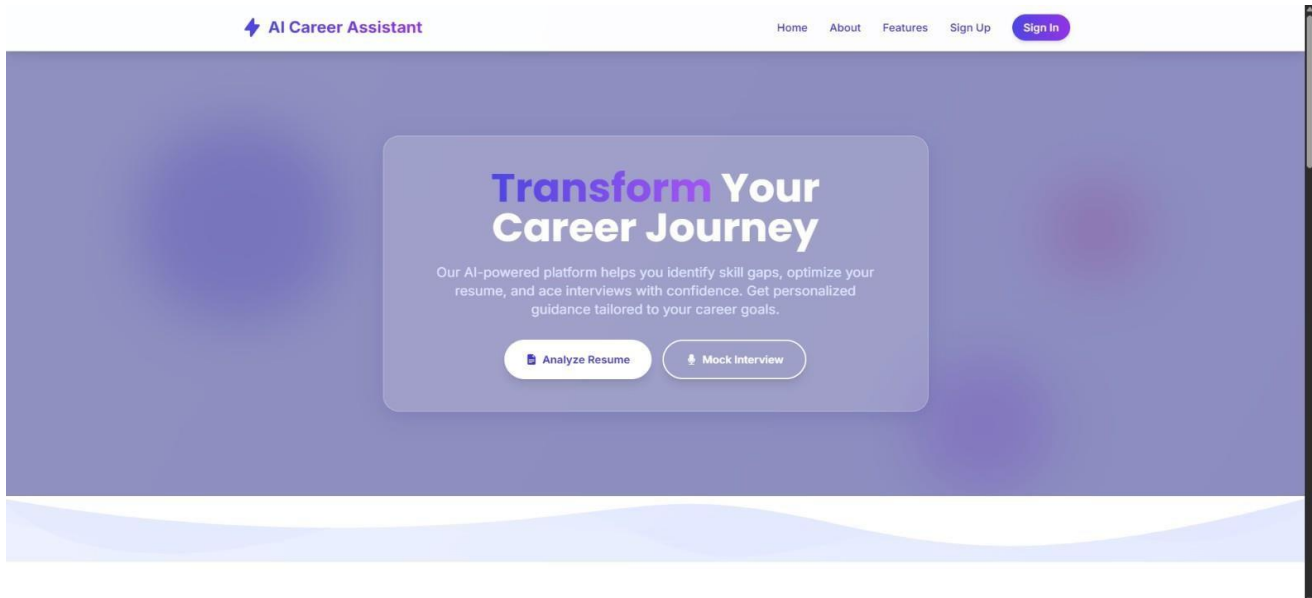
Figure 5.3 Home Page

Every Website must contain home page that chive navigation bar, footer and other

buttons etc. on it. Our Home page has an animated background and has two buttons (Analyze

Resume and Mock Interview):

It also has navigation bar on the top of screen and the nav bar has following buttons:

· Home

· About Us

· Features

· Sign Up

· Sign In

The main section includes two key buttons:
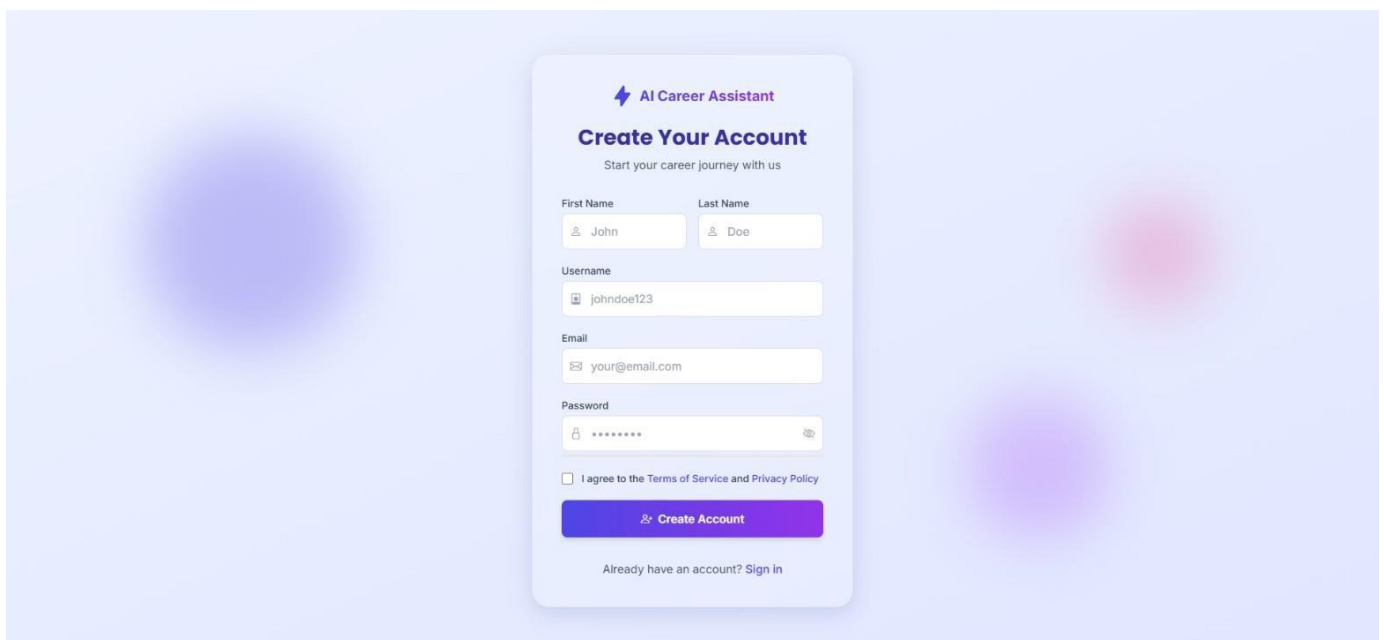
· Signup/Login as User.

· Login as Admin.

Visitors are directed to sign up if they're new, or log in to access personalized services. The

homepage showcases the core value of the platform—providing smart, AI-powered career

guidance including resume analysis and mock interviews.

This page will show to every user, a user clicks on his desired buttons and view every page. If he wants to get registered, he must be click on signup and if he wants to login, he must click on login button.

In this part user can perceive what we are serving. The landing page fundamentally is our alternate way page of diverting to any of the ideal page of the site. It shows every class of the Service there could be by which it makes simple to decide for the user as indicated by their ideal. Furthermore, could book and benefit the administrations from us.

5.3.2. Sign Up

Figure 5.4 Signup

Explanation

- For registration into the website, we need a full name, user name, email and password of the user then click on create account button.

- After this process user will be able to enter in the site (By login).

- In this part the user can sign up by putting the right credential according to the requirement of the account by the AICA.

- This is the registration page of our site for user. Registration page includes:

- First Name.

- Last Name.

- User Name.

- Email.

- Password.

5.3.3. Log In
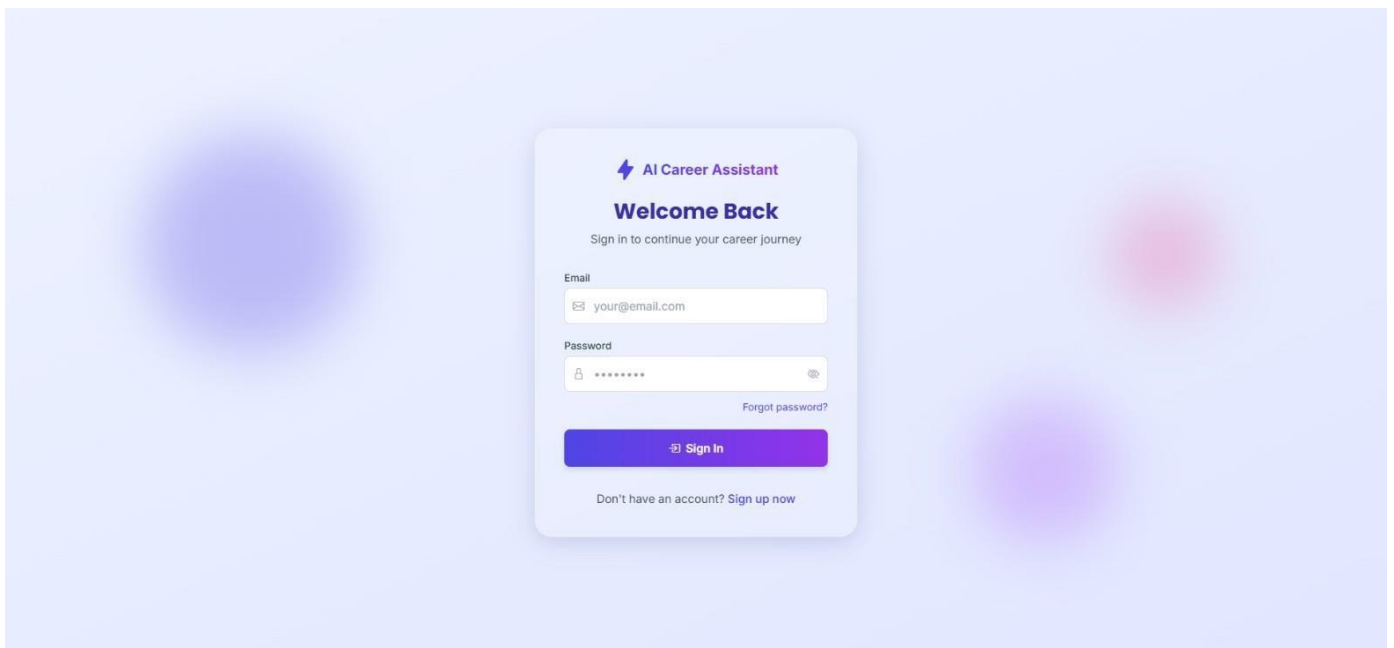
Figure: 5.5 Login

Explanation

- For login into the website, we need an email no and password of the user then click on login button. After this process user will be able to enter in the Website.

- After login, the dashboard will appear in front of user screen where he has allowed to enter resume and job description.

- This is the login page of our website. Login page includes:

- Email.

- Password.

- Login.

5.3.4. Services (Dashboard)
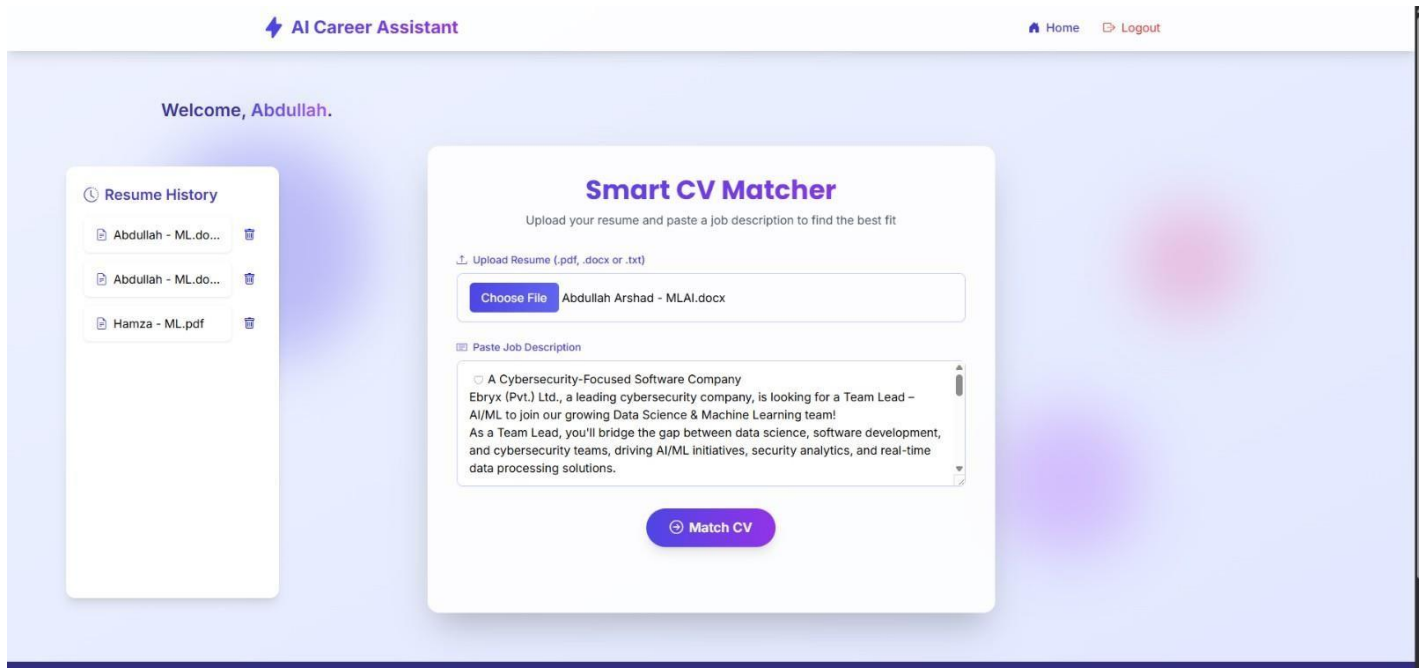
Figure 5.6.1 Dashboard

Explanation

The user dashboard is a central control panel that allows job seekers to:

• Upload resumes and job descriptions

• Access AI-powered gap analysis

• Practice interviews using the mock interview bot

• View system announcements

• Track progress and previous results

This page provides seamless navigation and shows system updates, profile data, and AI-generated insights.

5.3.5. Gap Summary and Visuals
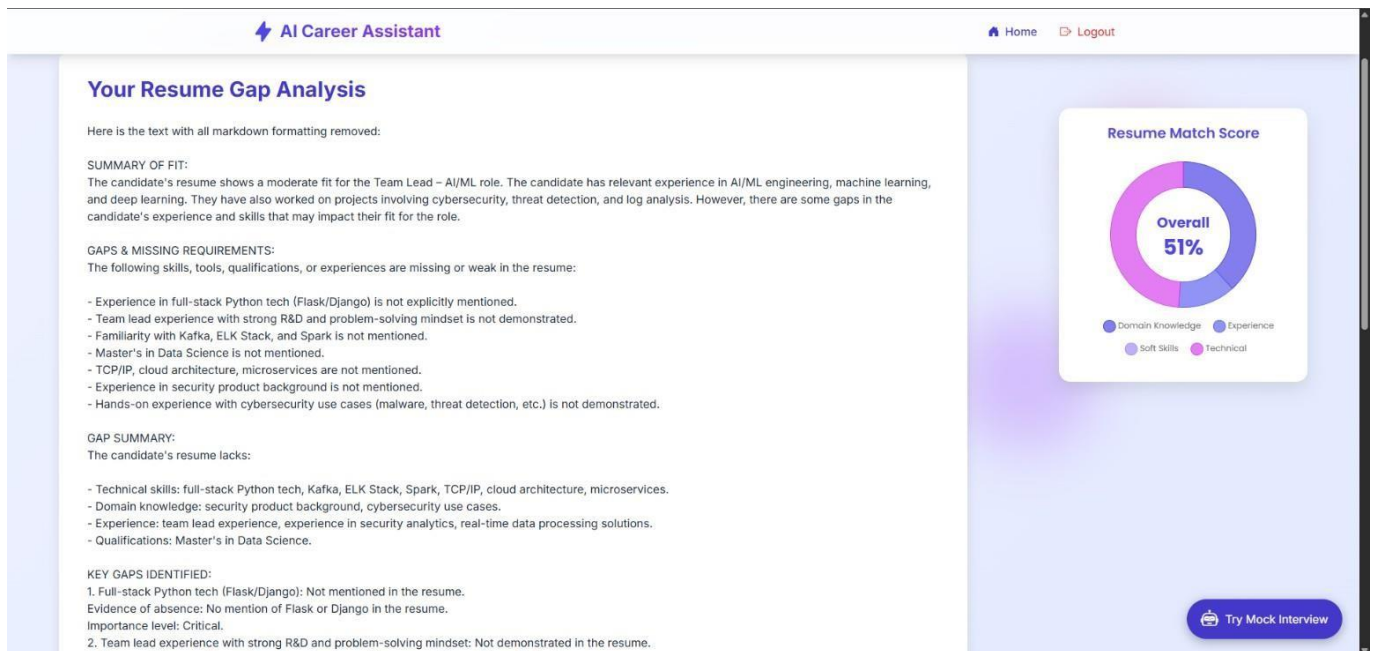
Figure 5.6.2 Gap Summary

Explanation

- User receive an AI-generated **Gap Summary Report** after submitting their resume and job description.

- Textual summary of strengths and weaknesses.

- Visual pie chart breakdown of four key area:

➢Technical Skills

➢Domain Knowledge

➢Soft Skills

➢Work Experience

5.3.6.   Mock Interview
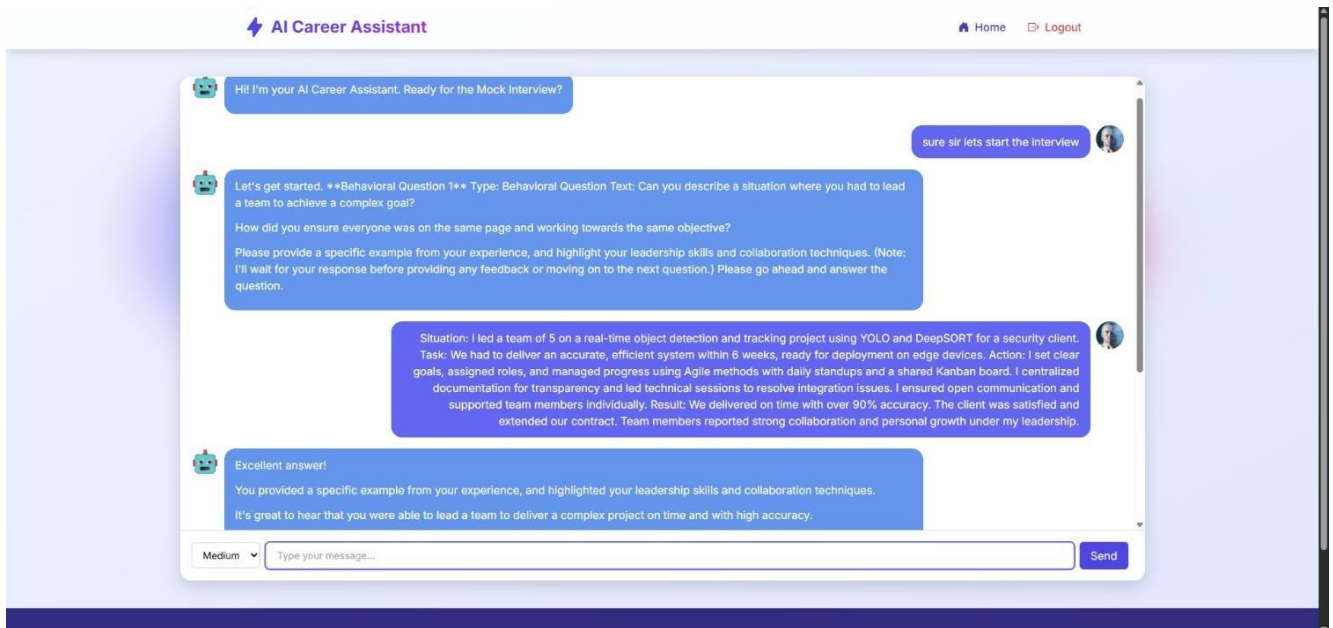
Figure 5.6.3 Mock Interview

Explanation

- This module simulates a real-world job interview using AI. Users can engage in a chat-like interface with the mock interviewer. The system asks role-specific questions and evaluates responses for:

- Communication clarity.

- Relevance of answer.

- Confidence and reasoning.

5.3.7. Admin Panel
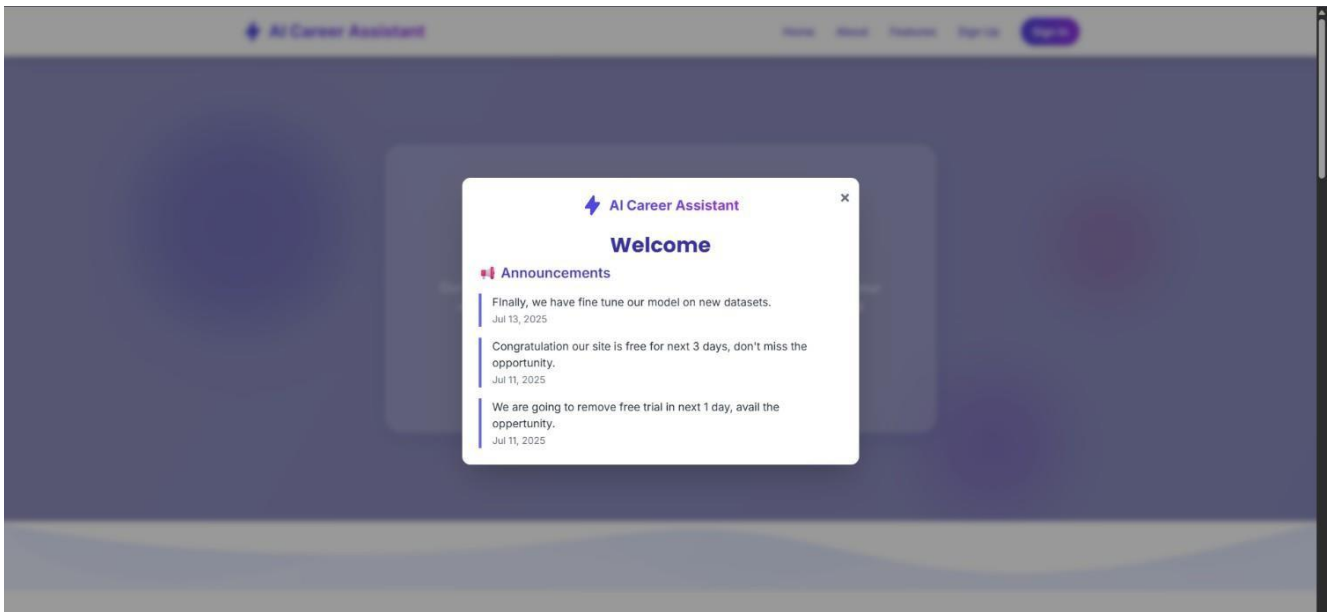
Figure 5.7.1 Announcements
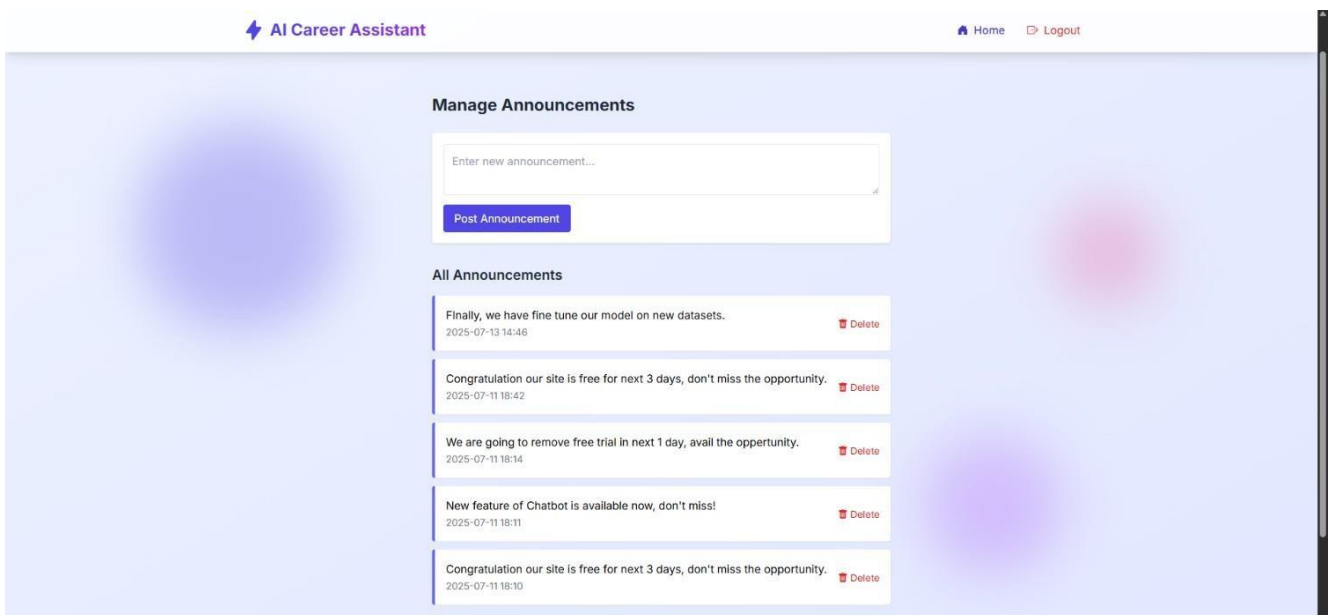
Figure 5.7.2 Admin Panel

Explanation

• This section of the admin panel allows administrators to manage platform-wide announcements.

• Admins can:

• Post new announcements via a secure form.

• View a list of all previously posted announcements, ordered by most recent.

• Delete any announcement with confirmation to ensure safe moderation.

• Announcements posted by the admin are dynamically displayed on the user-facing homepage. This section ensures:

• Users are always informed of important updates, changes, or opportunities.

• Only the latest announcements are shown in a clean, readable format.

• The messages are synchronized with the admin panel so any new or deleted announcement is reflected here in real time.

5.3.8. Generate OTP
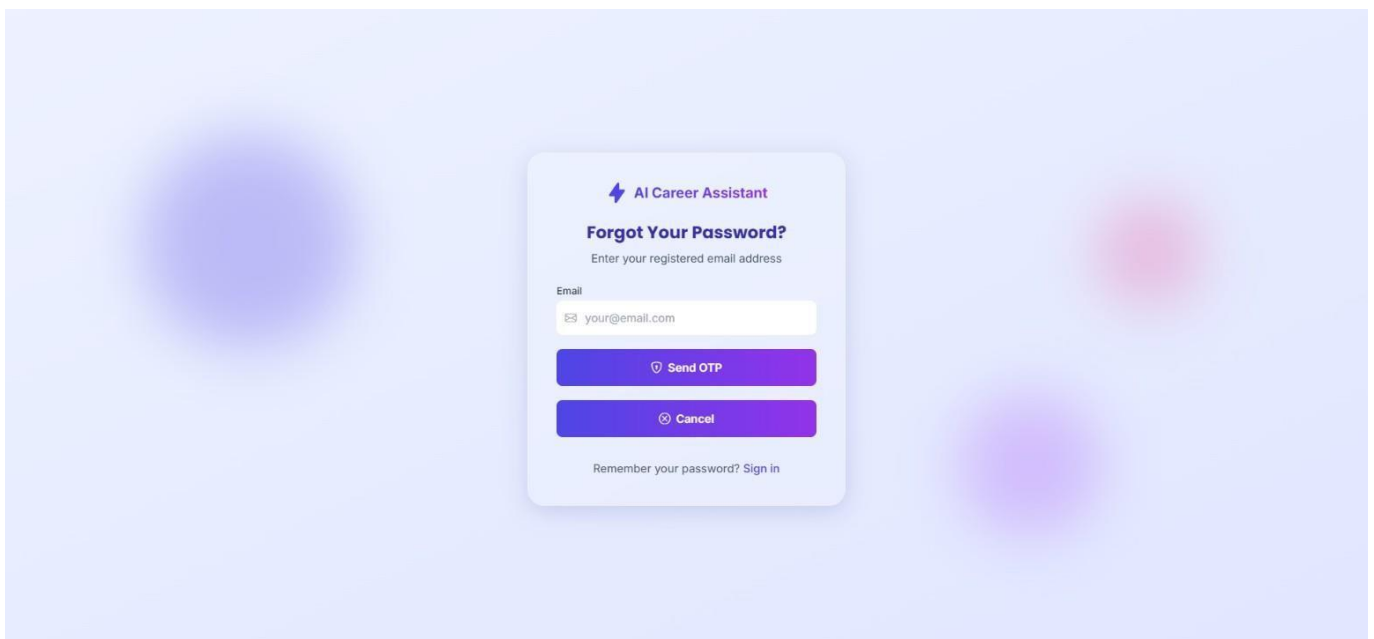
Figure 5.8 Generate OTP

Explanation

- When users request to reset their password, the system generates a **One-Time Password (OTP)** sent via email. This page confirms the request and facilitates secure password updates. Only registered emails are allowed.

5.3.9. Update Password
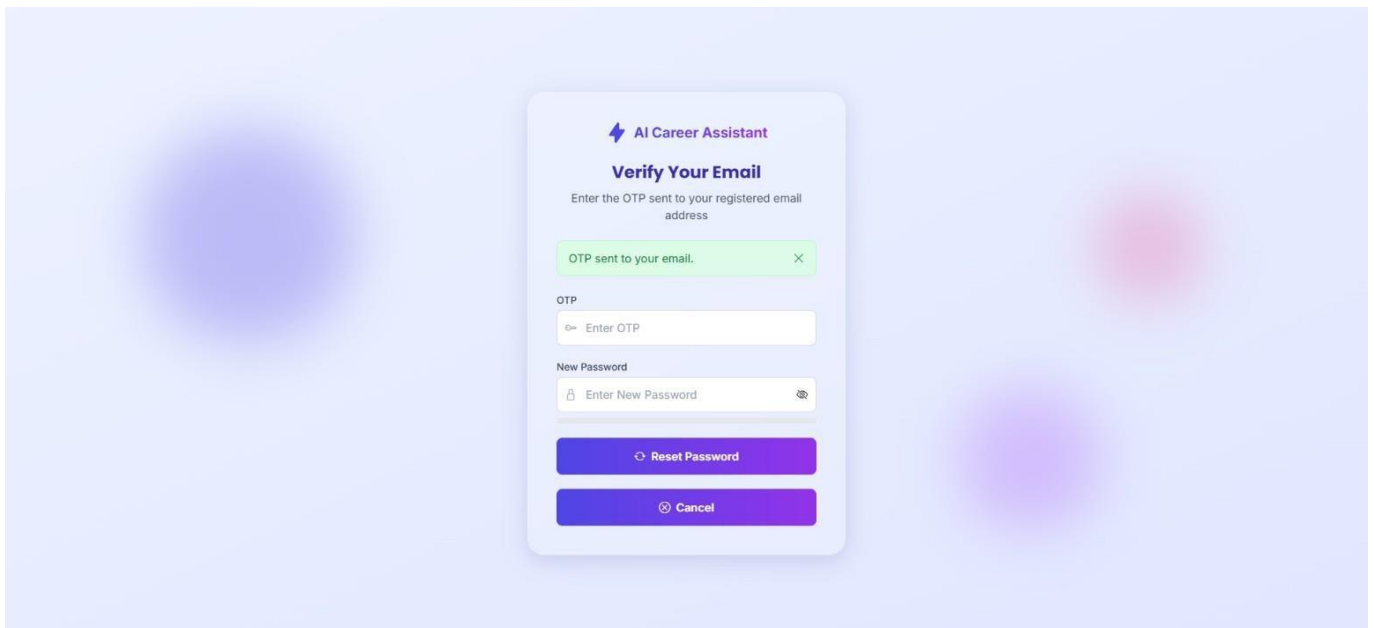
Figure 5.9 Update Password

Explanation

- After OTP verification, users are taken to this page to enter a new password. Passwords must follow platform security rules. Once submitted, the new password is stored securely using hashing and the user can log in immediately.

5.3.10. Email of OTP
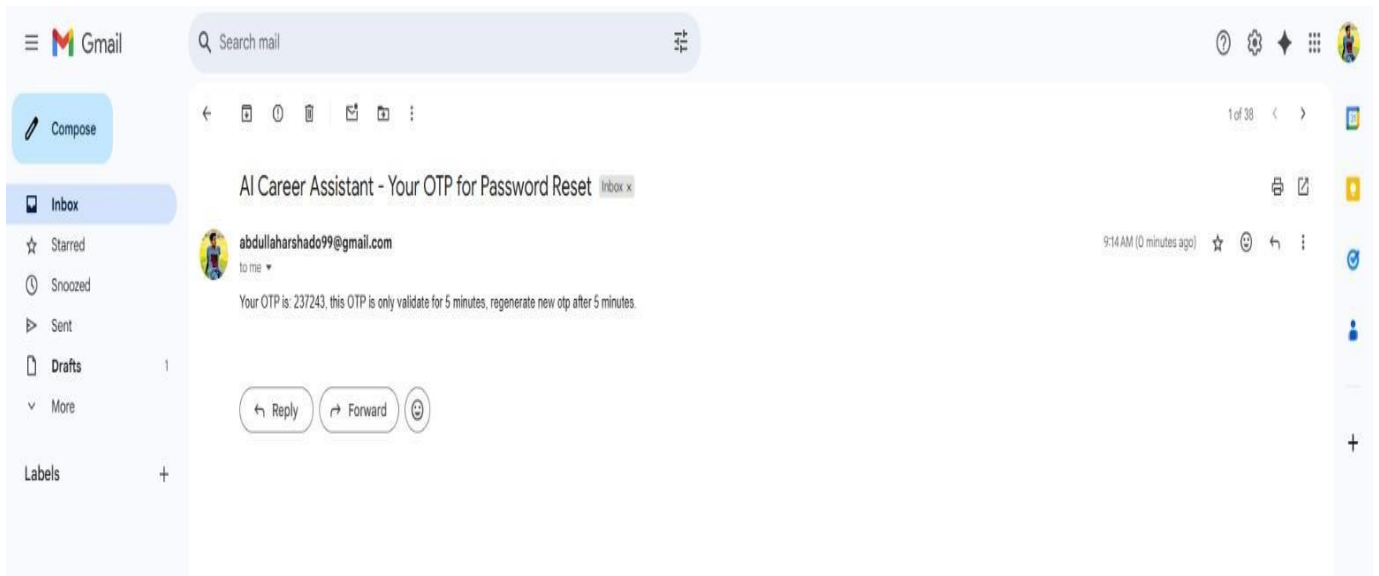
Figure 5.10 Email of OTP

Explanation

- This screenshot shows how the system sends a **password reset OTP** to the user's registered email. It confirms the email address, includes the OTP, and gives instructions on resetting the password securely. This ensures secure user identity verification.

CHAPTER 6

TESTING AND IMPLEMENTATION

In this chapter, we can test and evaluate our application and write the results which are taken during the testing. Through this, we can know where our system lacks and how does it behave when modules interact with each other.

6.1. Use Case Testing

Use case testing is a procedure that assists with distinguishing experiments that cover the whole framework, on an exchange-by-exchange premise, beginning to end. It is a portrayal of a specific utilization of the framework by a client. It is utilized generally in creating tests or frameworks for satisfactory levels.

In this technique, we will perform testing on each use case of the system so we can see if their workflow is normal and we will write them in form of a use case.

Use case testing is a methodology that helps with recognizing tests that cover the entire structure, on a trade-by-trade premise, start to finish. It is a depiction of a particular usage of the system by a customer. It is used commonly in making tests or systems for palatable levels.

In this strategy, we will perform testing on each utilization instance of the framework so we can check whether their work process is ordinary and we will think of them in type of a utilization case.

6.1.1. Registration

Test Case: Registration		
	Steps	**Description**
	1	A: User opens the AI Career Assistant Platform.
Main success scenario A: Actor **S: System**	2	S: System displays homepage with sign-up options.
	3	A: Clicks "Sign Up as Job Seeker".
	4	S: Displays registration form.
	5	A: Fills in details (name, email, password, etc.).
	6	A: Clicks Submit.
	7	S: Validates inputs and stores user data in database.

	8	S: Shows success message and redirects to login.
Extensions:	2a	Slow/ no internet connection. S: The home page will not be opened.
	6a	Weak/invalid password. S: Error message shown.
	6a	The registration number should be valid. S: The message will be displayed as an invalid email.

Table 6.1: Registration

6.1.2. Log In

Test Case: Login		
	Steps	**Description**
	1	A: Opens website and selects "Login".
Main success scenario A: Actor	2	S: Displays login page with two options: Job Seeker / Admin
S: System	3	A: Enter the email address and password of the account.
	4	S: Authenticates user against DB
	5	S: Allow access to the account and redirect to the dashboard.
Extensions:	2a	Slow/ no internet connection. S: The dashboard page will not be opened.
	4a	Email address and password are invalid. S: The message will be displayed of invalid email and enter password again.

Table 6.2: Log-In

6.1.3. Contact Us

	Steps	Description
Test case: Contact Us:		
	Steps	**Description**
	1	A: Open the website.
Main success scenario A:	2	S: Display the main page.
Actor	3	A: Select the contact us option from navigation bar.
S: System	4	S: Display the contact us page on user screen.
Extensions:	1a	Slow/ no internet connection. S: the main page and contact us page will not be opened.

Table 6.3: Contact Us

6.1.4. Dashboard

	Steps	Description
Test case: Uploading resume and job description:		
	Steps	**Description**
	1	A: Logs in as job seeker.
Main success scenario A:	2	S: Displays dashboard with options.
Actor	3	A: Navigates between Resume Upload, Gap Summary, Mock Interview.
S: System	4	S: Loads respective content without reload or delay.
Extensions:	1a	Slow/ no internet connection. S: the dashboard page will not be opened.
	4a	Backend unresponsive. S: Error message or retry logic.

Table 6.4: Dash Board

6.1.5. Gap Summary

Test case: Summary:

	Steps	Description
	1	A: Uploads resume and job description.
Main success scenario A: **Actor** **S: System**	2	S: Processes the input and sends it to LLM.
	3	S: Displays a textual summary and pie chart with gap percentages
Extensions:	1a	Slow/ no internet connection. S: the main page will not be opened.
	1a	File format unsupported. S: Error displayed
	2a	Server timeout. S: Retry prompt

Table 6.5: Gap Summary

6.1.6. Mock Interview

Test case: Chatbot:

	Steps	Description
	1	A: Click on Mock Interview button allocated on gap summary page.
Main success scenario A: **Actor** **S: System**	2	S: Displays chat interface.
	3	A: Responds to AI prompts.
	4	S: Returns new questions or feedback based on response.
	5	A: Ends session and receives final score or summary.
Extensions:	2a	Slow/ no internet connection. S: Delayed Response.
	4a	Inappropriate input. S: System gives warning or fallback response.

Table 6.6: Mock Interview

6.1.7. User Name

On our website, the username is used. If the username contains alphabets and numbers, it is considered valid otherwise it is invalid.

Input	Result
Abdullah@287	**Invalid** (special characters not allowed).
123	**Invalid** (must start with a letter**).
Username: abdullah9o	**Valid.**

Table 6.7: User Name

6.1.8. Password

On our website, the password is used to login into a system. The password should have a length of letter and alphabets so, it is considered valid otherwise it is invalid.

Input	Result
abcde	**Invalid** (only alphabets).
abc123	**Invalid** (too short).
12345678	**Invalid** (only numbers).
adnan123456	**Valid.**

Table 6.8: Password

6.2. Black Box Test Case

A simple login screen of software or a web application will be tested for seamless user login. The login screen has two fields, email and password as an input and the output will be to enable access to the system.

A black box testing will not consider the specifications of the code, and it will test the valid email and password to login to the right account. This form of testing technique will check the input and output.

- A user logged in when inputs a present email and correct password.
- A user receives an error message when enters email and incorrect password.

Black Box Testing is also called behavioral/ Specification-Based/ input-output testing.

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure and implementation details. This test case is done be mainly focusing on the input and output of AMS. Following is the prominent Test Strategy amongst the many used in Black box testing.

- Boundary Value Analysis

- Equivalence Class Partitioning

- State Transition Testing

- Decision Table Testing
- Graph Base Testing

6.2.1. BVA or Boundary Value Analysis

Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values is acceptable by the system or not. Using this technique, we have taken the test conditions as partitions and designed the test cases by getting the boundary values of the partition. We have gotten both valid boundaries (from the valid partitions) and invalid boundaries (from the invalid partitions).

On our "Signup" page, there is a password field, which will accept an alphanumeric more than 8 digits, no less. It will also not accept the only alphabets/ only numbers in that field.

6.2.2. Equivalence Class Partitioning

Using this technique, we have divided the test conditions of our project into groups, and from each group, we have tested only one condition by keeping in mind that if a condition from a group works then all the conditions from the same group will work.

By using this method, we have saved a lot of time by reducing the total number of test cases.

On our "Signup" page, there are some fields that will collect data from the user. "First Name" and "Last Name" fields will only accept the numeric value.

6.2.3. Decision Table Testing

Decision Table Testing is a software testing methodology used to test system behavior for various input combinations. In this systematic approach, the several input combinations and their corresponding system behavior are represented in tabular form.

Decision Table testing is used for testing two or more inputs that have a logical relationship. A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

The decision table for the login page is given below. A user with a valid email and password can log in. If the email or password is not given correctly, it will generate an error message.

o **Conditions**

Enter Valid Email Address, Enter Valid Password

o **Actions:**

Show User Portal, Show Error Message.

ID	Condition/Action	Test case 1	Test case 2	Test case 3	Test case 4
Condition 1	Valid User Email	T	T	F	F
Condition 2	Valid Password	T	F	T	F
Action 1	User Portal	Execute			
Action 2	Show Error Message		Execute	Execute	Execute

6.2.4. Graph Base Testing

This technique of Black box testing involves a graph drawing that depicts the link between the causes (inputs) and the effects (output), which trigger the effects. This testing utilizes different combinations of output and inputs. Let us explain graph box testing and how we have implemented it on our login module.

Scenario: Resume Upload and Gap Analysis Flow

Graph Nodes (States)

• Start.

• Login Successful.

• Resume + JD Uploaded.

• AI Gap Analysis Triggered.

• Gap Summary + Visual Displayed.

• End.

Valid Transitions

• Start → Login Successful.

• Login Successful → Resume + JD Uploaded.

• Resume + JD Uploaded → AI Gap Analysis Triggered.

• AI Gap Analysis Triggered → Gap Summary Displayed.

• Gap Summary Displayed → End.
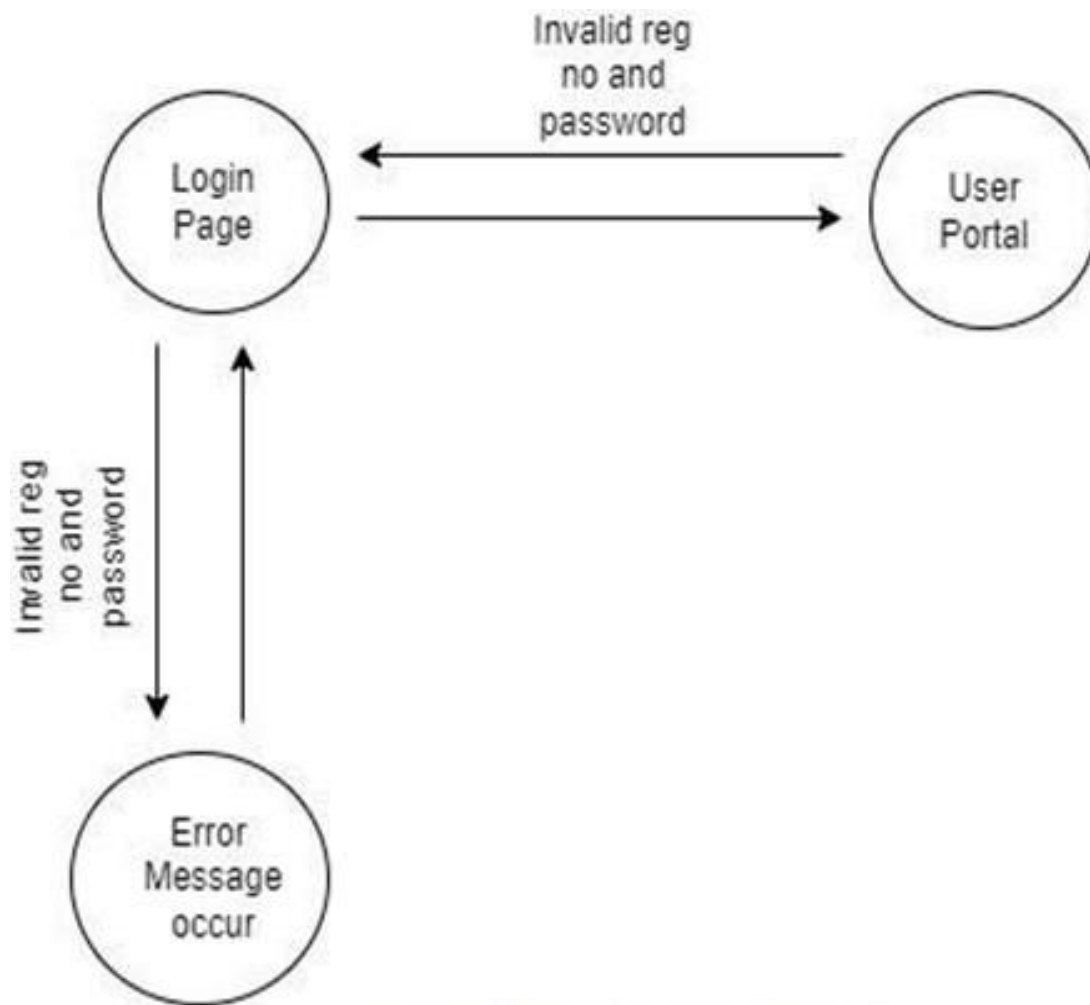
Figure: 6.2.4 Graph Base Testing

6.3. Equivalence Class Partitioning

The equivalence Partitioning Method is otherwise called Proportionality class dividing (ECP). It is a product testing method or discovery testing that partitions input area into classes of information, and with the assistance of these classes of information, experiments can be determined. An ideal experiment recognizes a class of blunder that may require numerous subjective experiments to be executed before the broad mistake is noticed.

In identicalness dividing, equality classes are assessed for given information conditions. At whatever point any information is given, then, at that point kind of info condition is checked, then, at that point for this information conditions, Equality class addresses or portrays set of legitimate or invalid states. Equivalence partitioning is also a testing technique to check a site. It is used when random values are inserted in a system to show how it behaves during wrong inputs.

Data Flow Testing is a particular technique of programming testing that spotlight information factors and their qualities. It utilizes the control stream diagram. With regards to classification Information, stream testing will be considered as a sort of white-box testing and primary kinds of testing. It keeps a check at the information getting focuses by the factors and its utilization focuses. It is done to cover the way testing and branch testing hole.

The cycle is directed to recognize the bugs as a result of the erroneous use of information factors or information esteems. For example, Introduction of information factors in programming code, and so on.

In the context of the **AI Career Assistant**, ECP is applied to input fields like email, password, resume upload, and job description submission, ensuring they only accept valid data types and formats.

6.3.1. Registration

Module: Registration				
Description: This test validates the flow of user registration data through the form fields and into the backend database. The form must only accept valid email format and passwords (alphanumeric with minimum 8 characters).				
Step No.	**Step Description**	**Expected Result**	**Actual Result**	**Status**
1	Click on Sign up form	Data inserted	Data inserted	Pass
	{email=abdullaharshado99@gmail.com, password=abdullah123456}			

Table 6.9: Registration

6.3.2. Log In

Module: Login				
Description: This test verifies that login credentials match database entries and access is granted if valid.				
Step No.	**Step Description**	**Expected Result**	**Actual Result**	**Status**

1	Login {email=abdullaharshado99@gmail.com ,password=abdullah123456}	Check database	Login	Pass

Table 6.10: Login

6.3.3. Feedback

Module: Feedback				
Description: This test ensures that when a user submits a query or feedback through the contact form, the message and user data are correctly stored in the database.				
Step no	**Step Description**	**Expected Result**	**Actual Result**	**Status**
1	Add contact data {name=Abdullah, email=abdullaharshado99@gmail.com, message= hi I visit your site I want to get information about gap scoring services.}	Data inserted	Data inserted	Pass

Table 6.11: Feedback

6.4. Unit Testing

Unit testing is a software testing method that tests individual units/components individually. A unit can be described as the smallest piece of code that can be tested in an application. Unit tests are typically performed by developers during the application development phase.

Python/Flask Unit testing is the method of testing small pieces of code/components in isolation in your Flask application. This helps in improving the code quality and helps in finding bugs early on in the development life cycle.

Flask is a popular Python library works on interpreter. It is used to develop server-side components of web applications to do unit testing in Flask.

6.4.1. Registration

Test type	Unit testing
Test case number	TC-UT-001
Test case name	User Registration
Test case description	Validates that all required fields are filled properly and data
Items to be tested	First Name, Last Name, Email, Username, Password form validations, unique email constraint.
Specifications : All field must fill Can't fill all field	Successful registration. Failure message.

Table 6.12: Unit Testing of Registration

6.4.2. Log In

Test type	Unit testing
Test case number	TC-UT-002
Test case name	User Login
Test case description	Verifies that only registered users with valid credentials can access the platform.
Items to be tested	Email and password fields, bcrypt hash verification, login session logic
Specifications: All field must fill Can't fill all field	Successful login. Failure message.

Table 6.13: Unit Testing of Log-In

6.4.3. Log Out

Test type	Unit testing
Test case number	TC-UT-003
Test case name	User Logout
Test case description	The user should press the logout button.
Items to be tested	The account running on-site should be logged out.
Specifications: Press the logout button	Logout successfully and the main page is opened.

Table 6.14: Unit Testing of Log-Out

6.5. White Box Testing

White Box Testing is a software testing technique in which internal structure, design, and coding of software are tested to verify the flow of input-output and to improve design, usability, and security. In white-box testing, code is visible to testers so it is also called Clear box testing, Open box testing, transparent box testing, Code-based testing, and Glass box testing.

For **AI Career Assistant**, white box testing was used to validate:

• Resume parsing logic.

• Gap score distribution (ensuring total = 100%).

• Role-based access in login logic (admin/user).

• Secure password handling via bcrypt.

• File upload validation paths.

6.5.1. Statement Coverage

• Statement Coverage is a white box testing technique in which all the executable statements in the source code are executed at least once. It is used for the calculation of the number of statements in the source code that have been executed.

• The main purpose of Statement Coverage is to cover all the possible paths, lines, and statements in the source code. Statement coverage is used to derive scenarios based upon the structure of the code under test.

• **Statement Coverage** ensures that every line of code in the system is executed at least once. In our testing:

- ➢ All conditional statements for login (email, password match).

- ➢ File upload & validation paths for resume/job description.

- ➢ Email OTP sending flow for password reset.

- ➢ AI pipeline output parsing logic.

- ➢ Role-based redirection after login.

$$Statement\ Coverage = \frac{Number\ of\ executed\ statements}{Total\ number\ of\ statements} \times 100$$

6.5.2. Path Coverage

- Path coverage testing is a specific kind of methodical, sequential testing in which each individual line of code is assessed.

- As a type of software testing, path coverage testing is in the category of technical test methods, rather than being part of an overarching strategy or "philosophy" of code. The way that path coverage testing works are that the testers must look at each individual line of code that plays a role in a module and, for complete coverage, the testers must look at each possible scenario so that all lines of code are covered.

- Integration testing (in some cases called mix and testing, contracted I&T) is the stage in programming testing wherein singular programming modules are joined and tried collectively. It happens after unit testing and before approval testing. Joining testing takes as its information modules that have been unit tried, bunches them in bigger totals, applies tests characterized in a reconciliation test plan to those totals, and conveys as its yield the incorporated framework prepared for framework testing.

- **Path Coverage** validates all possible execution paths in a function or module. For example:
- Login module was tested for:
- Valid login (user/admin).
- Invalid password.
- Empty fields.
- Upload flow:
- Resume only upload.
- Resume + Job Description.
- Unsupported file formats.

6.6. Stress Testing

- Stress Testing is a kind of programming testing that checks the solidness and unwavering quality of programming applications. The objective of Stress testing is estimating programming on its vigor and mistake dealing with abilities under amazingly substantial burden conditions and guaranteeing that

- Product doesn't crash under crunch circumstances. It even tests past typical working focuses and assesses how the product functions under outrageous conditions.

- In Programming, Stress Testing is otherwise called Perseverance Testing. Under Pressure Testing, AUT is be pushed for a brief timeframe to know its withstanding limit. The most conspicuous utilization of stress testing is to decide the cutoff, at which the framework or programming, or equipment breaks. It additionally checks whether the framework exhibits viable blunder the board under outrageous conditions.

- In Programming, Stress Testing is generally called Perseverance Testing. Under Tension Testing, AUT is be pushed for a short period of time to realize its withstanding limit. The most obvious use of pressure testing is to choose the cutoff, at which the system or programming, or hardware breaks. It moreover checks whether the structure shows reasonable botch the board under unbelievable conditions.

6.7. Performance Testing

- Performance Testing is a product testing measure utilized for testing the speed, reaction time, steadiness, dependability, adaptability, and asset utilization of a product application under a specific responsibility. The fundamental motivation behind execution testing is to recognize and wipe out the exhibition bottlenecks in the product application. It is a subset of operational efficiency and furthermore known as "Perf Testing".

- Execution Testing is an item testing measure used for testing the speed, response time, unfaltering quality, reliability, versatility, and resource use of an item application under a particular duty. The crucial inspiration driving execution testing is to perceive and clear out the presentation bottlenecks in the item application. It is a subset of operational productivity and moreover known as "Perf Testing".

- **Speed** - Decides if the application reacts rapidly

- **Adaptability -** Decides the most extreme client load the product application can deal with.

- **Dependability -** Decides whether the application is steady under fluctuating burdens.

6.6.1. Sign Up

Test Title	Registration/
Description	This test case simulates the actions that user would use to perform registration.
Pre-Condition	User must be connected to the internet.
Test Case Steps	1- Open Website. 2- Click on signup. 4- Fill Registration Form. 5- Click on Submit Button.
Expected Result	4.0 sec
Response Time	3.0 sec
Average Load Time	2.0sec
Concurrent User	200 users at a time
Transaction Result	Passed

Table 6.17: Signup of Performance Testing

6.6.2. Log In

Test Title	Login
Description	This test case simulates the actions that user would use to perform Login.
Pre-Condition	User must be connected to the internet and have a valid account.
Test Case Steps	1- Open Website. 2- Click on login. 3- Fill Login Form. 4- Click on Login Button.
Expected Result	4.0 sec
Response Time	3.0 sec
Average Load Time	2.0sec
Concurrent User	200 users at a time
Transaction Result	Passed

Table 6.18: Login of Performance Testing

6.6.3. Feedback

Test Title	Feedback
Description	This test case simulates the actions that user would write feedback.
Pre-Condition	User must be connected to the internet.
Test Case Steps	1- Open Website. 2- Fill Feedback Form. 3- Click Submit Button.
Expected Result	4.0 sec
Response Time	3.0 sec
Average Load Time	2.0sec
Concurrent User	200 users at a time
Transaction Result	Passed

Table 6.19: Feedback of Performance Testing

CHAPTER 7
TOOLS AND TECHNOLOGIES

7.1. Languages/Tools & Frameworks

* HTML-5

* CSS-3

* JavaScript.

* Bootstrap-4

* Python

* Flask

* LangChain

* Scikit-Learn

7.1.1. HTML-5

* **HTML (Hypertext Markup Language)** is the main language used to create and structure content on websites. **HTML5** is the latest version of HTML and is widely used to build modern web pages.

* Unlike older versions, HTML5 comes with new features and improvements. One big update is that it now supports audio and video directly—meaning you no longer need extra plugins to play media on a web page. HTML5 also works better with today's internet technologies and is designed to make websites more interactive, fast, and user-friendly.

7.1.2. CSS-3

* **CSS3** stands for *Cascading Style Sheets Level 3*. It's the latest version of CSS, which is used to style and format the look of web pages. With CSS3, you can change things like colors, fonts, layouts, spacing, and even add animations to make a website more visually appealing.

* It's also important to know that **HTML and CSS aren't programming languages**— they're used for building and styling the structure of web pages, not for writing logic or functionality.

7.1.3. JavaScript

* JavaScript is the world's most popular programming language. JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate

images, and pretty much everything else.

• With JavaScript, you can add things like animations, real-time updates, pop-ups, sliders, form validations, and even control videos or images on a web page. It plays a huge role in how websites behave when users interact with them. As of now, **almost all websites (about 98%) use JavaScript** on the client side, meaning it runs directly in the user's browser. It's also used in other areas outside of browsers, like building mobile apps, servers, or even games.

• JavaScript is known for being **lightweight, flexible, and object-oriented**, and it supports things like reusable functions and modern web features.

7.1.4. Bootstrap-4

• **Bootstrap** is a free and open-source front-end framework used to build websites and web applications. It helps developers create **responsive, mobile-friendly** sites quickly and easily.

• Bootstrap comes with ready-to-use code for things like buttons, forms, navigation bars, and layouts, so you don't have to build everything from scratch. It uses **HTML, CSS, and JavaScript** to make your website look clean and work well on all screen sizes—from phones to desktops.

• **Bootstrap 4** is one of the popular versions that improved layout flexibility and added new features. It's completely free to use and saves developers a lot of time by offering pre-designed templates and components.

7.1.5. Python

• **Python** is a popular, high-level programming language known for being **simple, clear, and easy to learn**. It's widely used in many fields including **web development, data science, artificial intelligence (AI), machine learning (ML), automation, and more**.

• What makes Python so powerful is its **clean syntax**—which looks almost like regular English—making it beginner-friendly but also strong enough for advanced projects. Python supports both small and large-scale applications.

• Thanks to its huge community and a vast collection of libraries like **Flask** for web apps, **Scikit-learn** for machine learning, and **Pandas/NumPy** for data analysis, Python has become one of the most in-demand and flexible languages today.

• Whether you're building a simple website or training an AI model, Python is often the go-to choose because of its **versatility, readability**, and strong support.

7.1.6. Flask

• **Flask** is a lightweight and easy-to-use web framework written in **Python**. It helps developers build **web applications and APIs** quickly and with minimal setup.

• Flask is often called a **"micro-framework"** because it gives you just the tools you need to get started, without forcing you to use a lot of extra features unless you want to. This makes it perfect for both **small projects and larger, scalable applications**.

• It's great for building things like **login systems, dashboards, form handling, REST APIs**, and more. Flask is also known for being **flexible, beginner- friendly**, and easy to integrate with databases, machine learning models, and frontend tools like HTML and CSS.

7.1.7. SQLAlchemy

• **SQLAlchemy** is a powerful Python library that makes it easier to **work with databases**. Instead of writing long SQL queries by hand, developers can use SQLAlchemy to interact with the database using **Python code**.

• It works as an **ORM (Object-Relational Mapper)**, which means it lets you connect your Python objects (like classes) to database tables. This makes your code more organized, cleaner, and easier to maintain.

• SQLAlchemy supports **many different databases** like MySQL, PostgreSQL, SQLite, and more. It also gives you full control—so if you ever want to write raw SQL queries, you still can.

• Overall, SQLAlchemy helps developers **create, read, update, and delete data (CRUD)** in a more Pythonic way, and it's a great tool for managing databases in **Flask, Django**, or any other Python-based project.

7.1.8. LangChain

• LangChain is an advanced Python framework designed to integrate and manage large language models (LLMs) within applications. It simplifies the orchestration of prompts, memory management, chaining tasks, and connecting with various LLM providers like OpenAI, Anthropic, or Cohere. LangChain enables developers to create AI-driven experiences that are dynamic, context-aware, and modular.

• **In our use case (AI Career Assistant),** LangChain plays a crucial role in powering the intelligent analysis pipeline. It helps structure prompts for resume- to-job description comparisons, dynamically generates personalized gap feedback, and manages communication between the backend and the LLM. LangChain allows us to break down the analysis tasks into modular components such as resume evaluation, scoring, and mock interview generation— ensuring that AI responses are coherent, task-specific, and contextually relevant to each user.

7.1.9. Scikit-Learn

• Scikit-learn is a widely-used open-source machine learning library for Python. It provides efficient tools for data analysis and modeling, including algorithms for classification, regression, clustering, dimensionality reduction, and model evaluation. Built on top of NumPy, SciPy, and matplotlib, it's known for its ease of use, flexibility, and integration with the broader Python data science ecosystem.

• **In our AI Career Assistant platform,** Scikit-learn is used to perform lightweight machine learning operations, such as similarity matching between resumes and job descriptions, keyword extraction, and scoring models. These models assist in quantifying how well a resume aligns with the required skills and experiences of a given job post. By leveraging Scikit-learn, we are able to generate consistent, data-driven gap scores that complement the qualitative insights produced by LangChain and the LLM, resulting in a more well-rounded evaluation system.

7.2. Operating System

The development and testing environment used multiple operating systems for compatibility:

- Supported OS for Development & Deployment:

- **Windows 10 & 11** (Main dev environment).

- **Mac OS** (Cross-platform testing).

- **Ubuntu Linux** (Optional for production deployment).

- User Compatibility:

- Compatible with all modern browsers:

- Google Chrome (97+).

- Safari (15+).

- Microsoft Edge (Latest).

- Responsive layout supports:

- Desktop.

- Tablet.

- Mobile browsers (iOS & Android).

References

1. Brownlee, J. (2016). *Machine Learning Mastery With Scikit-Learn*. Machine Learning Mastery.

2. Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media, Inc.

3. LangChain Documentation. (2023). *LangChain: Building Applications with LLMs*. Retrieved from https://docs.langchain.com

4. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc.

5. Le, Q. V., & Mikolov, T. (2014). *Distributed Representations of Sentences and Documents*. ICML.

6. Jurafsky, D., & Martin, J. H. (2020). *Speech and Language Processing (3rd ed.)*. Draft retrieved from https://web.stanford.edu/~jurafsky/slp3/

7. Ng, A. (2019). *CS229: Machine Learning*, Stanford University. https://cs229.stanford.edu/

8. OpenAI. (2024). *GPT-4 Technical Report*. Retrieved from https://openai.com/research

Bibliography

1. Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). *Scikit-learn: Machine Learning in Python*. JMLR.

2. Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

3. Dean, J., & Ghemawat, S. (2004). *MapReduce: Simplified Data Processing on Large Clusters*. Google Research.

4. FreeCodeCamp. (2023). *Build a Flask App – Tutorial for Beginners*. https://www.freecodecamp.org/

5. IBM Developer. (2021). *Build AI-Powered Resume Matching with Watson NLP and Flask*. Retrieved from https://developer.ibm.com/

6. CareerBuilder AI Blog. (2022). *How AI is Transforming Recruitment and Resume Screening*. https://www.careerbuilder.com

7. UX Planet. (2021). *Best Practices for User-Centered Design in Career Portals.*

https://uxplanet.org

8. Real Python. (2022). *Using Flask for Web Applications*. https://realpython.com

9. Toptal Engineering Blog. (2022). *NLP Techniques for Resume Parsing and Matching*. https://www.toptal.com/

10. GitHub Repository – LangChain. https://github.com/hwchase17/langchain

11. Flask Mega-Tutorial by Miguel Grinberg. https://blog.miguelgrinberg.com

12. Stanford NLP Group. (2020). https://nlp.stanford.edu

13. Hugging Face Documentation. (2023). *Transformers and NLP Pipelines*. https://huggingface.co/docs.

GitHub Repository

To explore the complete source code, review implementation details, or contribute to the development of the **AI Career Assistant – Resume Gap Analyzer**, please visit the official GitHub repository:

https://github.com/abdullaharshado99/Resume-Gap-Analyzer

The repository includes:

• Full frontend and backend code.

• Flask-based server implementation.

• Resume gap analysis logic.

• AI-powered mock interview module.

• User authentication system.

• Database schema and sample data.

Feel free to fork, clone, or raise issues for collaboration or enhancement.