1. Introduction

Today rapidly evolving of digital era, job market traditional recruitment processes are increasingly being replaced by intelligence automated solution. Finding the right job time consuming task as it is often difficult to match job opportunities with a candidate's qualifications and interests.

Even after identifying a suitable role, the next challenge is building a high-quality resume and guidance to build it and it is time consuming task. Another significant issue is the lack of transparency in job application tracking because many websites provides only job apply link but not provide any proper tracking information about the job application.

Many candidates especially fresher graduates struggle with these issues. They often waste time for searching job create resume and preparing for job interview without guidance to solve these problems we develop end to end AI Platform that solve these problems.

1.1 Our Project Include

Job Recommendation System: using fine-tuned BERT model, our system intelligently understands both user profiles and job descriptions to suggest the most relevant job opportunities.

AI based resume builder: An intelligent tool that helps users create high quality resumes quickly with AI guidance.

AI Job Interview Assistant: built from scratch using a compact GPT-style model trained on a custom dataset to help of preparing of job interviews.

Interview Application: A scheduling and video interview tool inspired by platforms like Google Mee, this tool allows recruiters to easily schedule and conduct interviews.

1.2 Project Objective

The **primary goal** of this platform is to minimize manual effort in the hiring process and to create a seamless, intelligent recruitment experience. By streamlining every step of the hiring process from finding the right job and building a strong resume to preparing for interviews and scheduling them this system tackles many of the common challenges in today's recruitment landscape.

It helps job seekers stand out with better tools and preparation, while giving recruiters smarter ways to connect with the best candidates quickly and efficiently.

2. Related Work

The field of smart recruitment has experienced significant advancements thanks to the integration of artificial intelligence (AI) and natural language processing (NLP). These technologies help speed up hiring processes and improve the overall performance of recruitment platforms. More and more websites are now incorporating AI to boost efficiency, enhance user experience, and make recruitment smarter and faster.

In recent years, AI-powered recruitment tools have transformed traditional hiring by automating routine tasks and personalizing job recommendations. A major breakthrough is in job recommendation systems, which use advanced NLP models like BERT (Bidirectional Encoder Representations from Transformers) [1] to better understand job descriptions and candidate profiles.

By combining content-based filtering (which looks at skills, qualifications, and job details) with collaborative filtering (which considers past user behavior), these hybrid models provide more accurate and personalized job suggestions [2]. This helps users quickly find jobs that match their skills, cutting down the time spent on manual searching and making the application process much smoother.

Another exciting development is AI-based resume builders that leverage prompt-driven tools, such as OpenAI's GPT [3] or Google's Gemini API [4]. These tools assist users in creating well-structured and tailored resumes faster, adjusting content and format according to specific company requirements.

When it comes to interview preparation, AI-powered chatbots and virtual assistants are becoming widely used. They mimic real interview situations by asking relevant questions, offering immediate feedback, and even assessing communication skills. These chatbots can generate helpful responses based on user input, enabling users to prepare effectively and save time.

Advanced models like GPT-3 and GPT-4 can handle various interview styles—technical, behavioral, and more—and are increasingly popular in both academic and professional environments [3], [6].

Finally, interview scheduling has been made much easier through automation using Google Meet and Zoom APIs [7]. Platforms can now automatically create meeting links, send calendar invitations, and manage interview timings, which helps candidates attend interviews promptly and reduces coordination hassles for recruiters.

3. System Overview

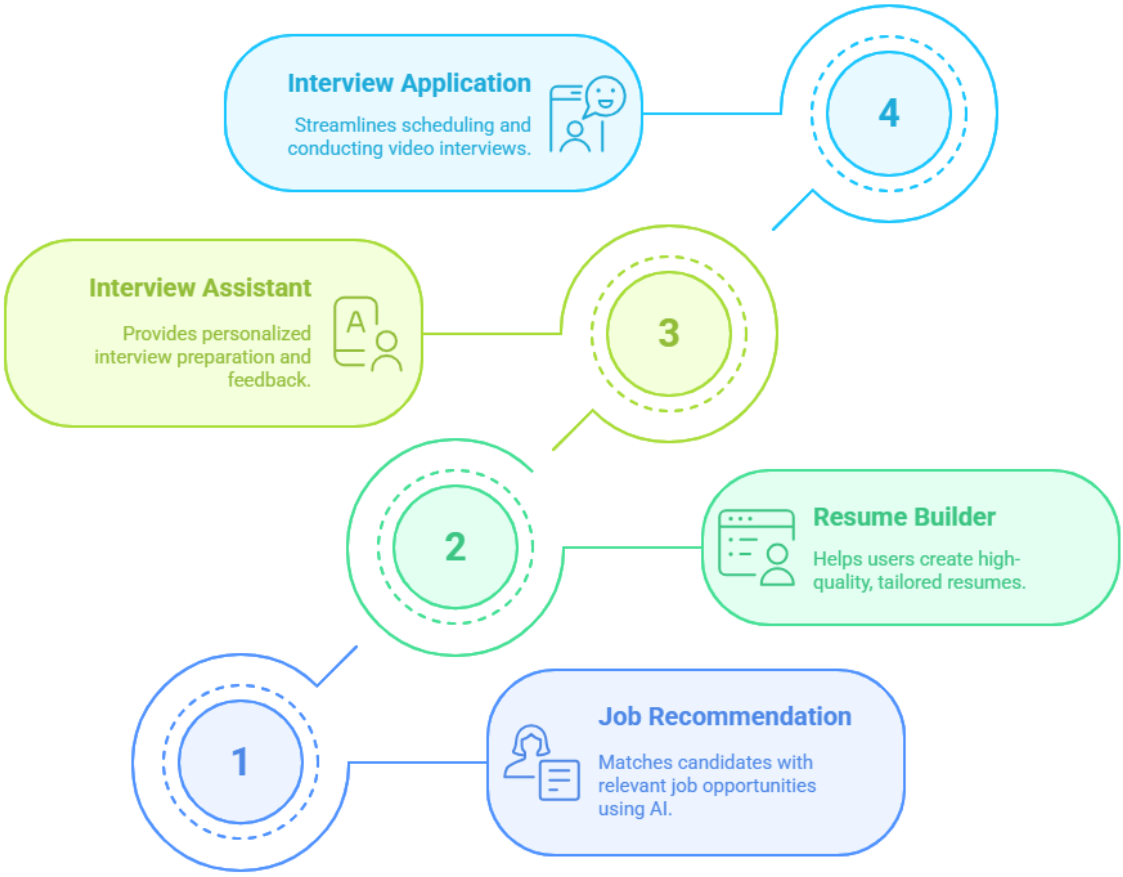

Fig. 1. Structured Workflow for Project

This is the full project workflow of the system, like one website, and these are four components integrated with the website. First, the **job recommendation system**, where users see job recommendations, and if a user applies for a job, then they move to the next step. If the user doesn't have a resume, our platform provides the facility to create a **resume** with the guidance of AI. Then they move to the next step, which is the **AI Interview Assistant**. Here, the user can ask interview-related questions to boost their confidence. The next step is the **interview process**. If the user is selected, the company recruiter helps schedule the job meeting/interview. All these facilities are provided on a single platform.

- Technologies used (Machine learning, Deep learning , Python, Django, React, HTML ,CSS, JavaScript, Node js, React.js , SQLite, Express.js ,MongoDB, Redux)
- Library used (Pandas, NumPy, Keras , Plotly, scikit-learn, transformers, scikit-learn metrics , faiss, pickle, nltk)

4. Working of the System

4.1 User Profile Management

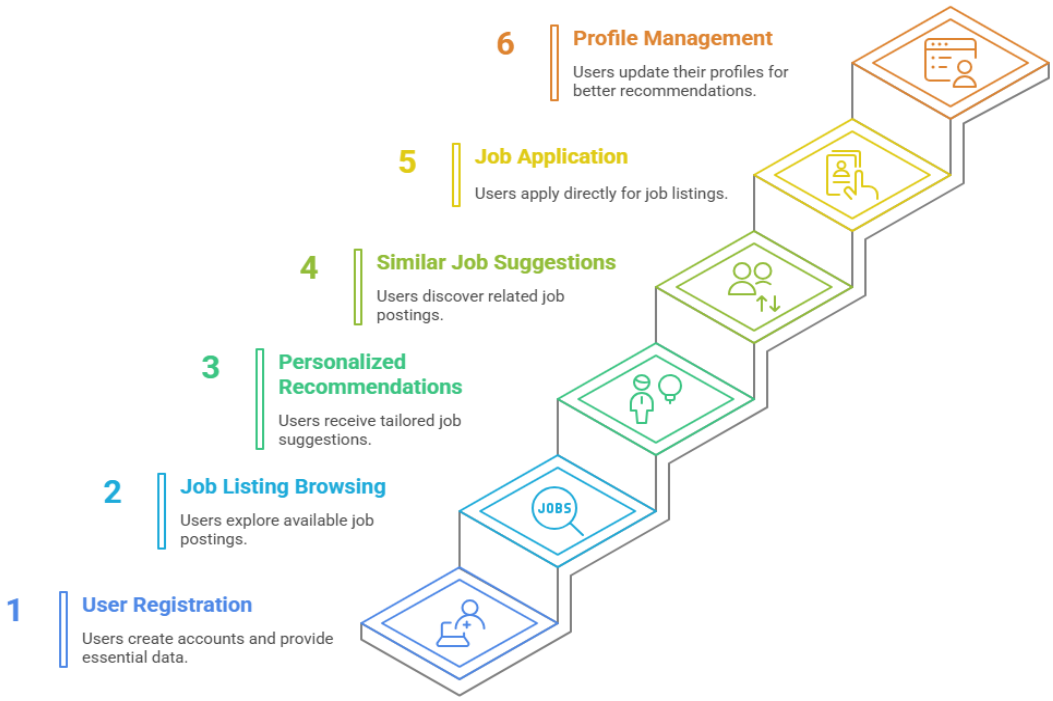

Fig. 2. User Profile Management

1. User Registration and Login

- Purpose: Allows users to create an account and securely log into the system.
- Functionality: Registration forms collect essential data (name, skills, education, etc.) Once logged in, the system begins tracking user behavior to offer personalized job services.
- The Facility also users to sign with their google account and login with it

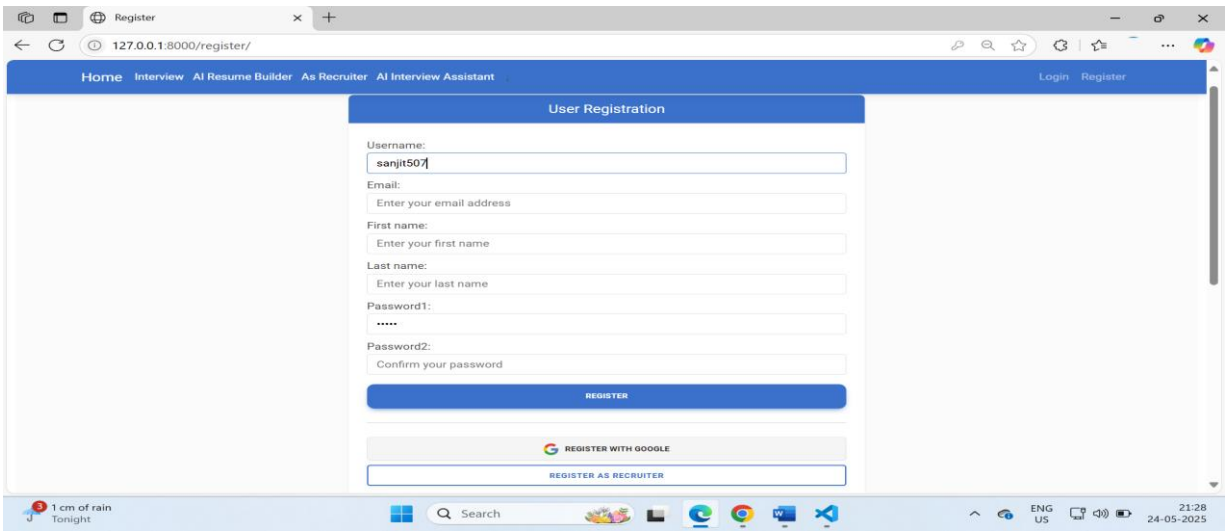

Fig.3. User Registration

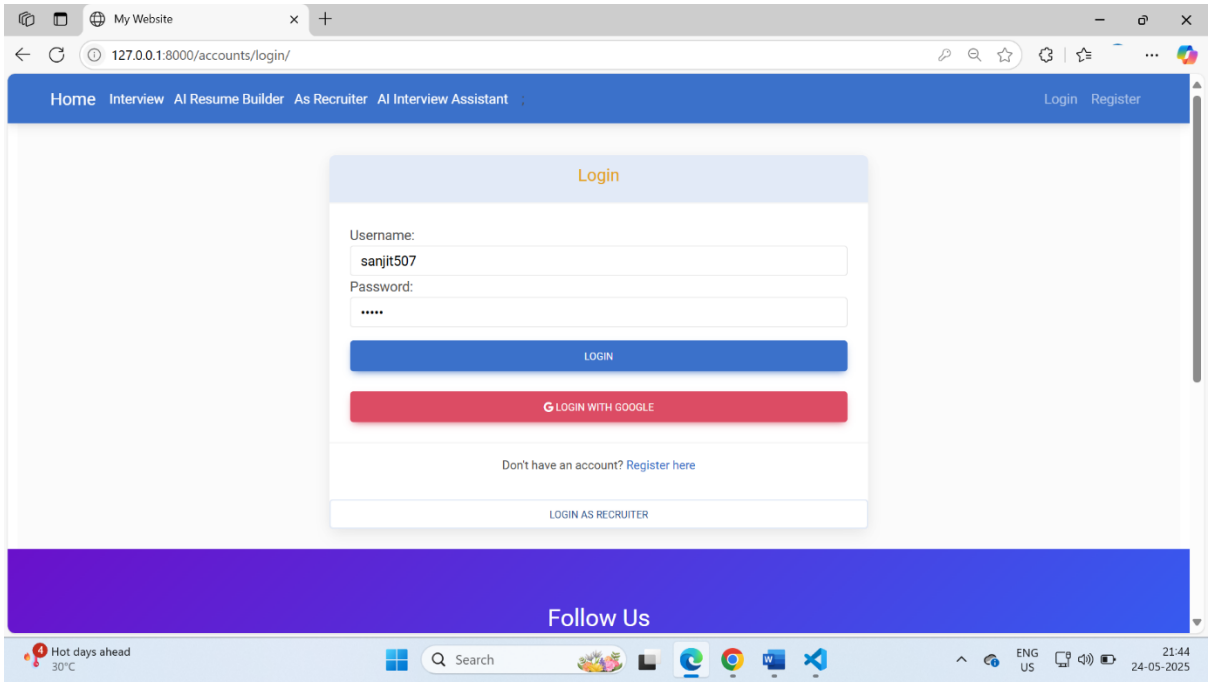

Fig.4. User Login

2. Job Listing and Browsing

- Purpose: Enables users to explore all available job postings.
- Functionality: Jobs are displayed based on category, location, or relevance.

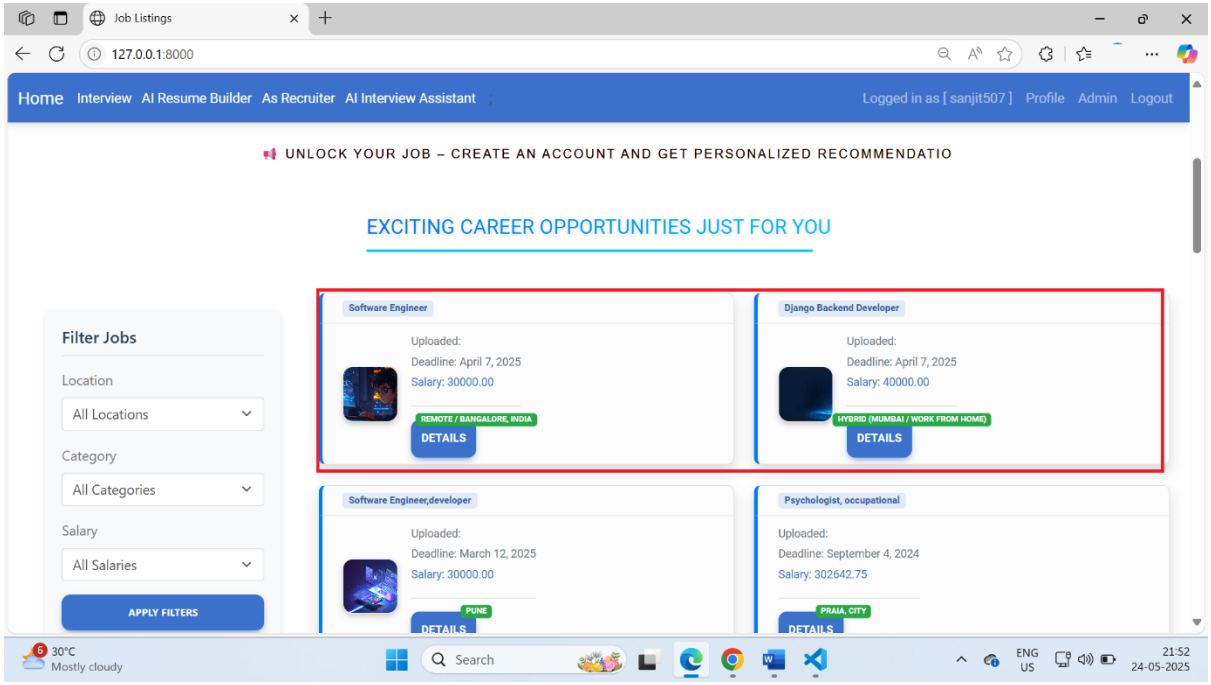

Fig.5. Job Listing and Browsing

3. Personalized Job Recommendations

- Purpose: Deliver tailored job opportunities to users.
- Functionality: The system uses a deep learning model **(Job-Rec-v1)** to encode user profiles and job descriptions into semantic embeddings, enabling accurate content-based job recommendations

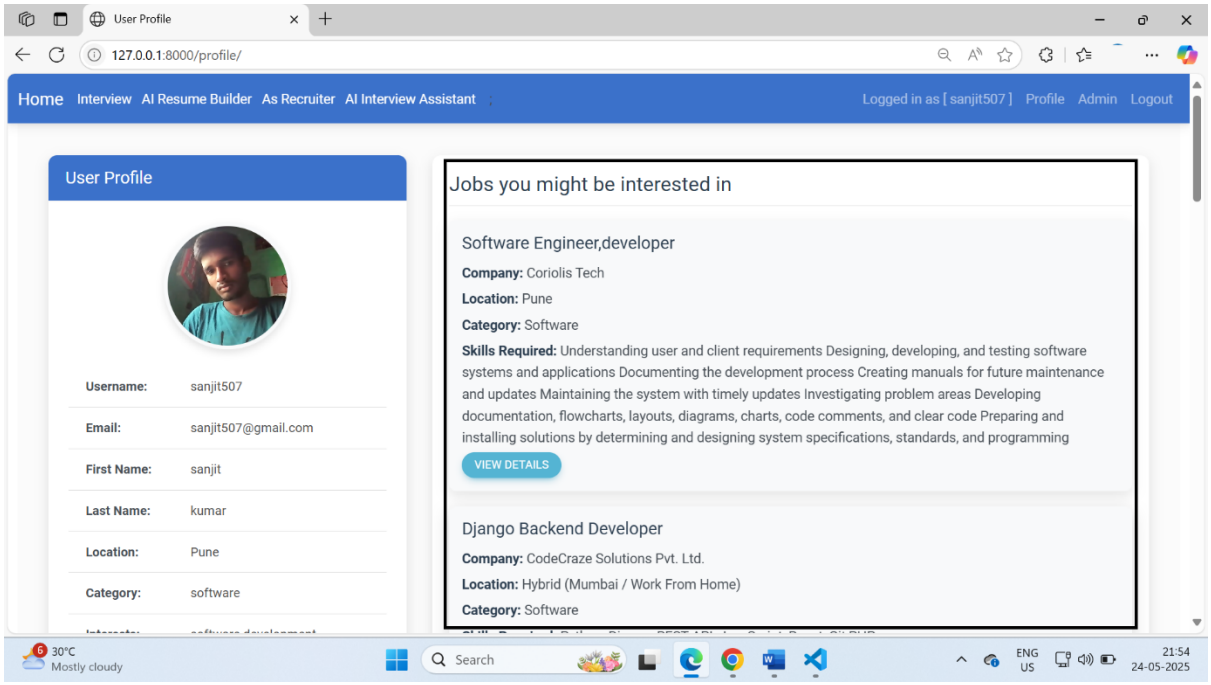

Fig.6. Personalized Job Recommendations

4. Similar Job Suggestions

- Purpose: Increase discoverability of jobs similar to those the user has interacted with.
- Functionality: When a user views or applies for a job, the system automatically shows related postings using content-based filtering or embedding similarity.

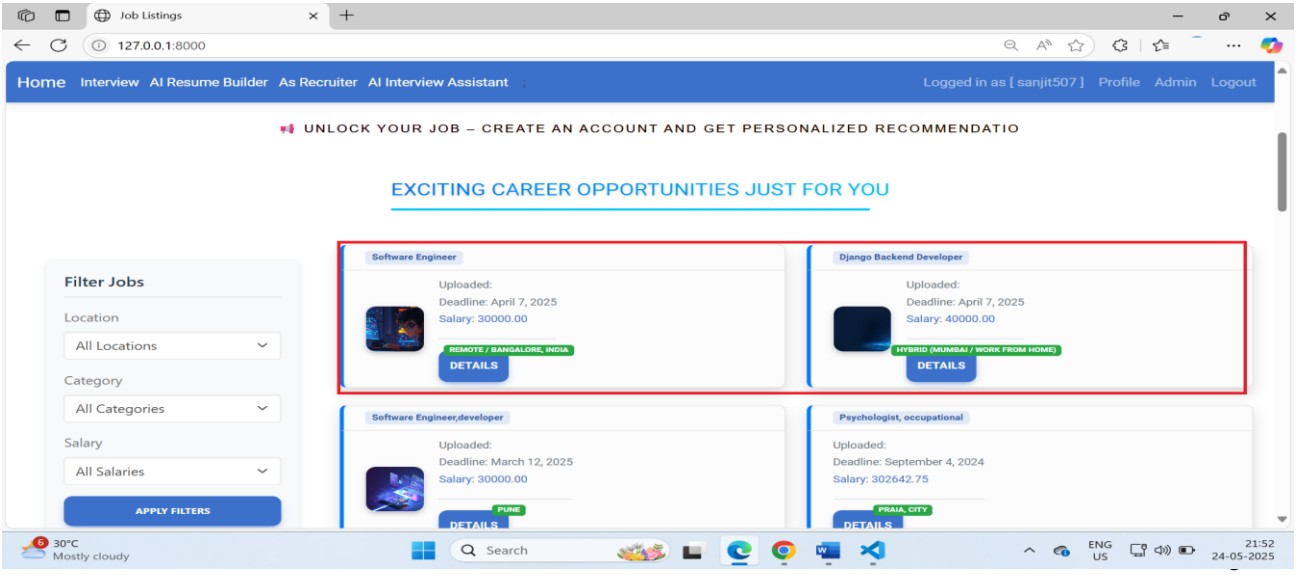

Fig.7. Explore the various jobs

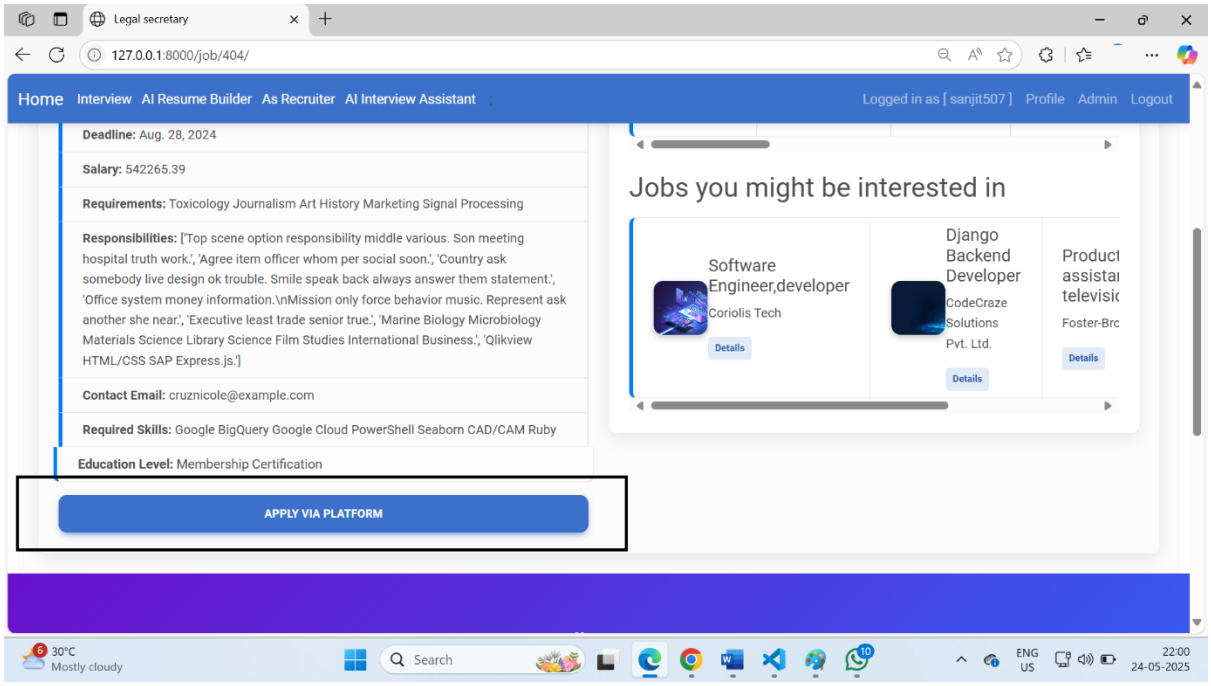

Fig.8. Job Apply

When user click on apply button then admin panel show the details about which user apply job which jobs.

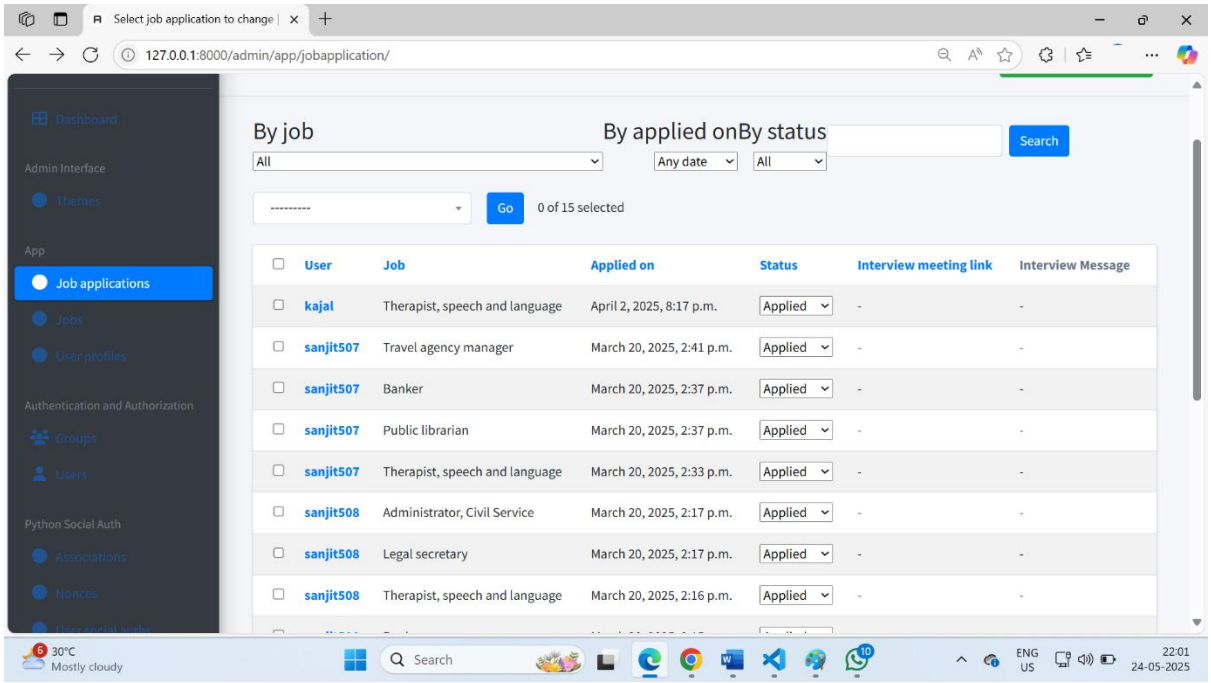

Fig.9. When apply job then show details on the admin panel

This is the main admin panel when any user apply the job application section the job application apply users.

6. User Profile Management

- Purpose: Keep user data up-to-date to ensure accurate recommendations.
- Functionality: Users can update skills, experiences, education, and preferences, which improves the relevance of future job suggestions.

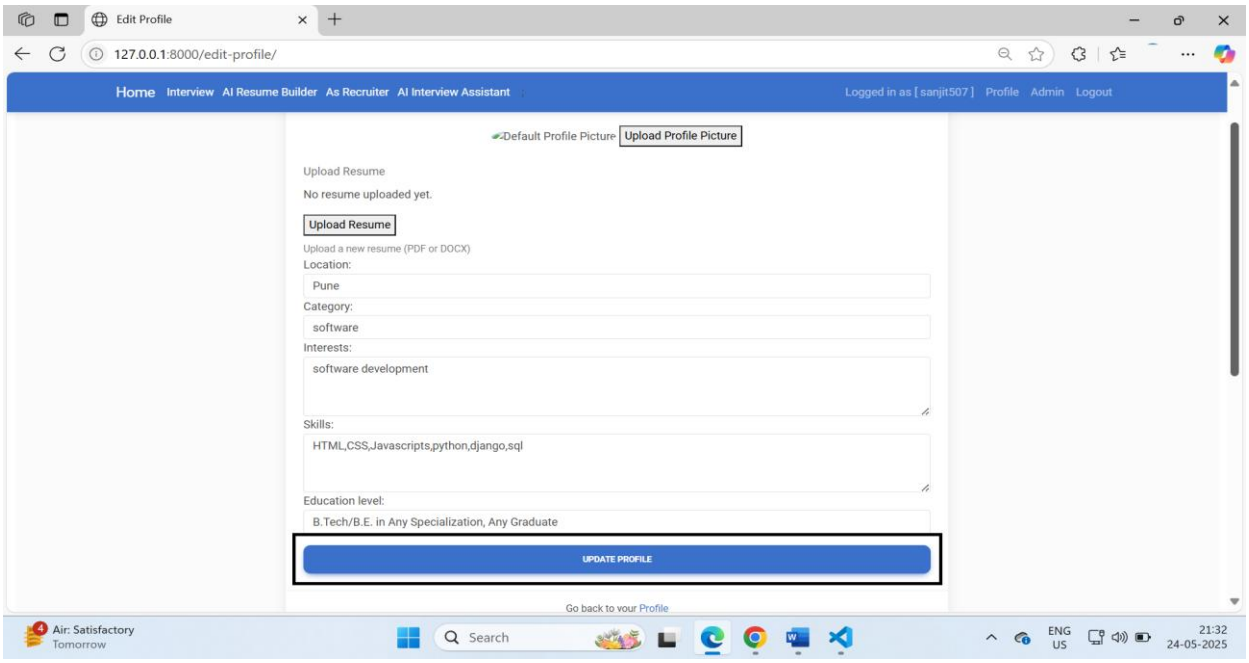

Fig.10. Update the user Profile

4.1.1 job Application Tracking

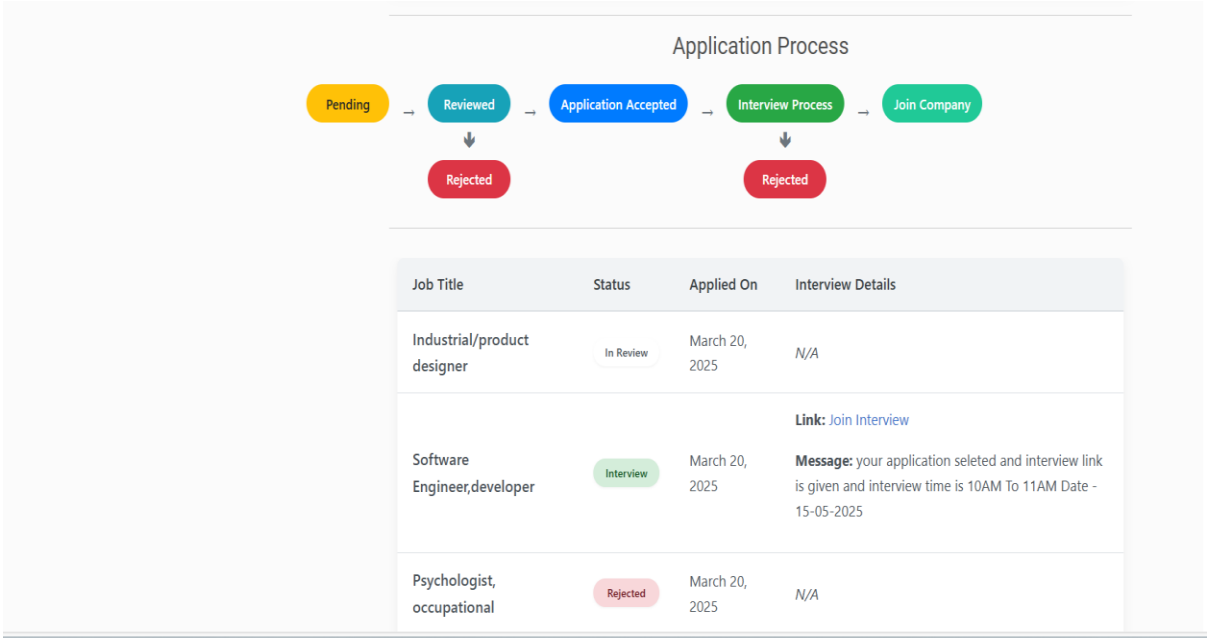

Fig. 11. User Track Job Application

Our platform provides a seamless **Job Application Tracking** system that helps users monitor the progress of their job applications in real-time.

Step 1: Apply for a Job

- Users log in to their account.
- Browse and apply for jobs directly through the platform.
- Once applied, the job appears in the **"Profile"** section with a status of **Pending**.

Step 2: Application Review

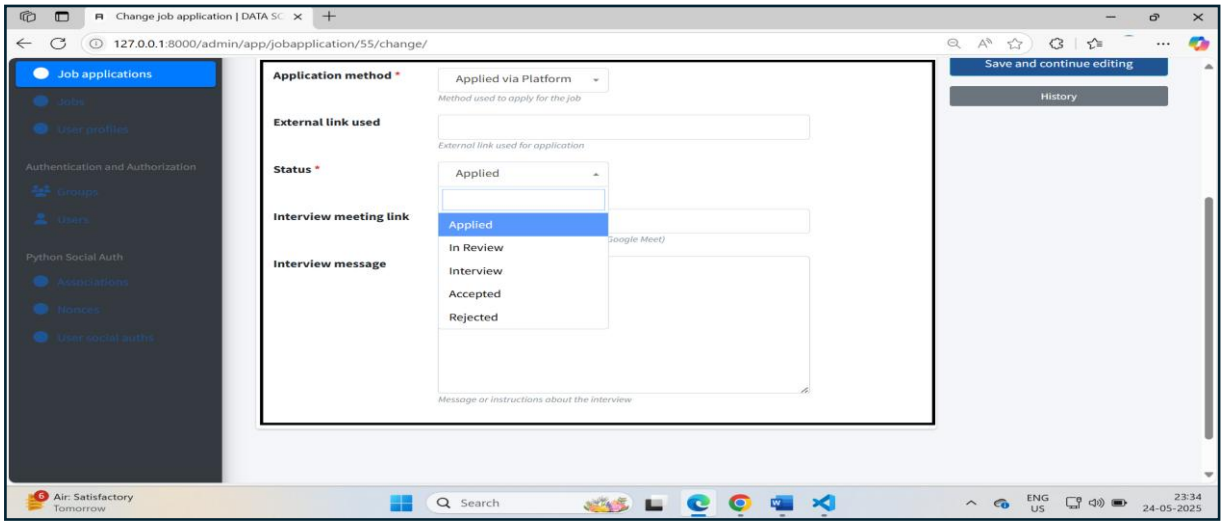

Fig.12 Application Review

- The job provider (admin or company HR) reviews the applicant's profile and resume.If the application **does not meet** the job requirements, the status will be updated to **Rejected**.
- If the profile matches the job criteria, the application status changes to **Accepted**.

Step 3: Interview Invitation

- Shortlisted users receive an **interview invitation** via email. The email contains the **Interview link**, **date**, **time**, and other relevant interview details.Our team member build this application to easily redirect the page when click on the link and start the interview and all the facility like Google meet top candidates share screen and coding environment available on this application.

Step 4: Final Decision

- After the interview, the company will send the final decision.
- A **Final Offer Letter** (if selected) or a **Thank You Note** (if not selected) will be sent to the user's profile section of the website

4.1.2 Job Posting management And Admin panel

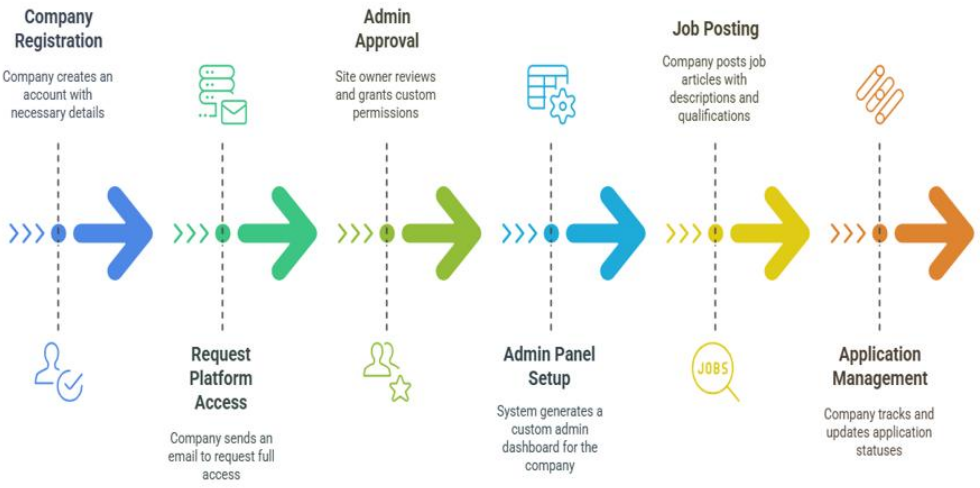

Fig. 13. Admin panel Management

We make it easy for verified companies to post jobs and manage applications with a powerful, permission-based admin panel. Here's how it works: **Step to company post job article**

Step 1: Company Registration

To begin, the company must create an account by providing the following details:

- Email address, Password, Industry Type, Location, Company Name, Website URL

After registration and login screen short

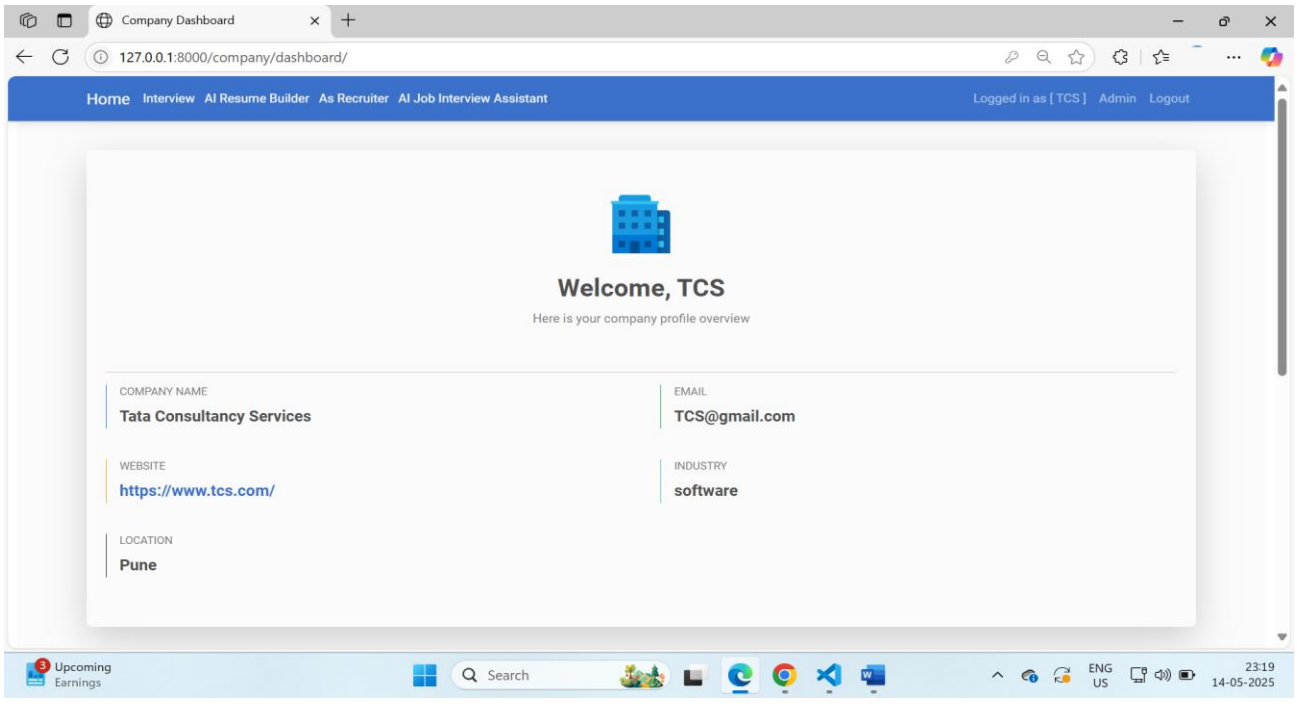

Fig.14. Dashboard of the company

Step 2: Request Platform Access

- After registration, the company logs into their account.
- To gain full access (job posting, application management, etc.), the company Fig. 14. Company Profile page
- must **send a request email** to **jobexpresswayadmin@gmail.com** .
- The site owner reviews the request and grants **custom permissions**.

Step 3: Admin Panel Access

Once approved, the system automatically generates a **custom admin dashboard** for the company with these features:

:➢ Post and manage job listings
:➢ View applicant profiles and resumes
:➢ Update application status (Pending → Accepted/Rejected)

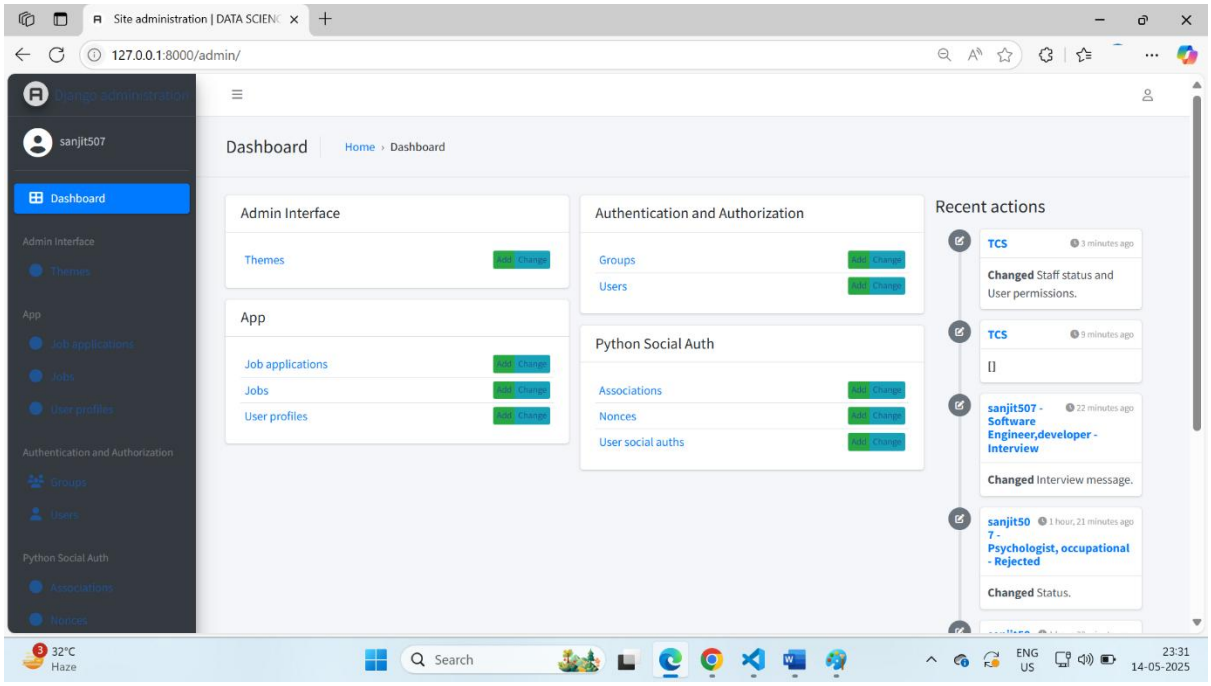

Fig.15. Main Control Admin Panel of the Website

This is the main admin panel of the website that control all the management if any permission to give to the other user like another content write job posting company and other staff and give here and if want to remove to other given staff and apply here lets below one example give of (assume TCS Company)

Step1:- main admin panel give special permission to separate admin panel means not main admin panel here main admin give permission then automatically Django build admin panel below given screen short.

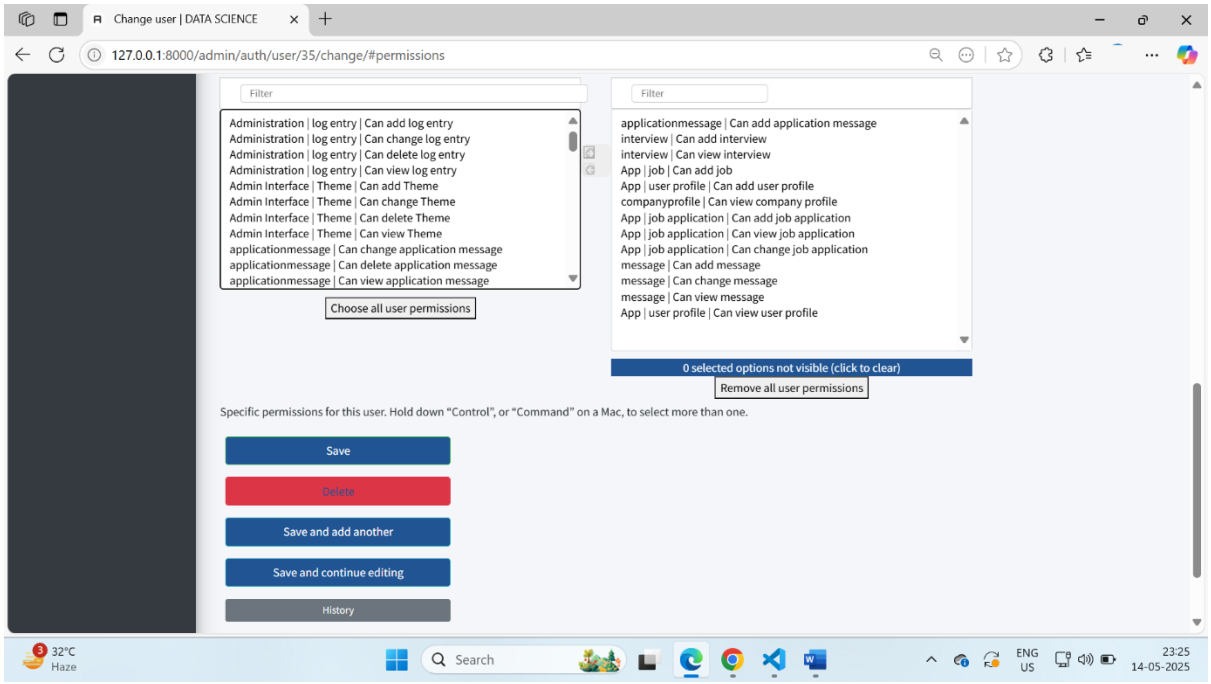

Fig.16. Permission Giving

Administrative access to the main admin of the website for the TCS company, allowing them to add job postings, view detailed information, and edit all related content.

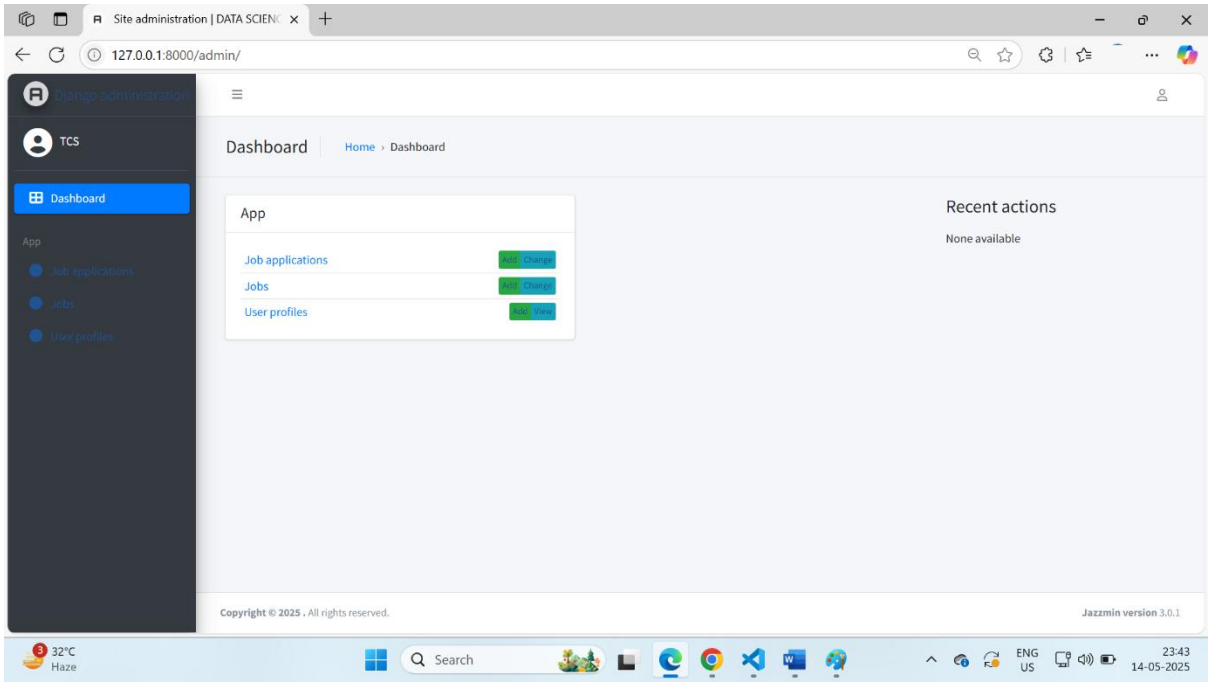

Fig. 17. Admin panel of TCS

Now TCS can add a job and see the user profile and edit the job posting etc. permission give to through the main control panel.

Step 4: Job Posting & Application Management

The company can now **post job articles** with descriptions, qualifications, and salary details.

- As users apply, the company can:
    - Track and update application statuses
    - Review resumes and profiles
    - Send next-step emails (e.g., interview invites or rejection notices)

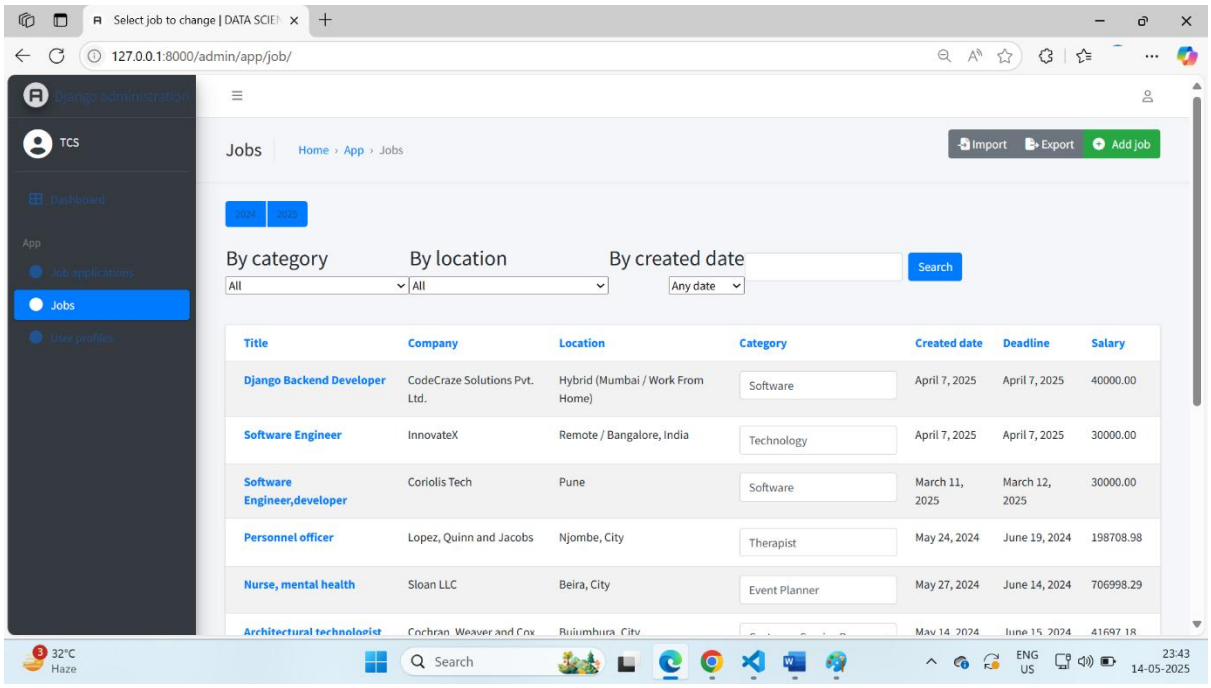

Fig. 18. View the job application

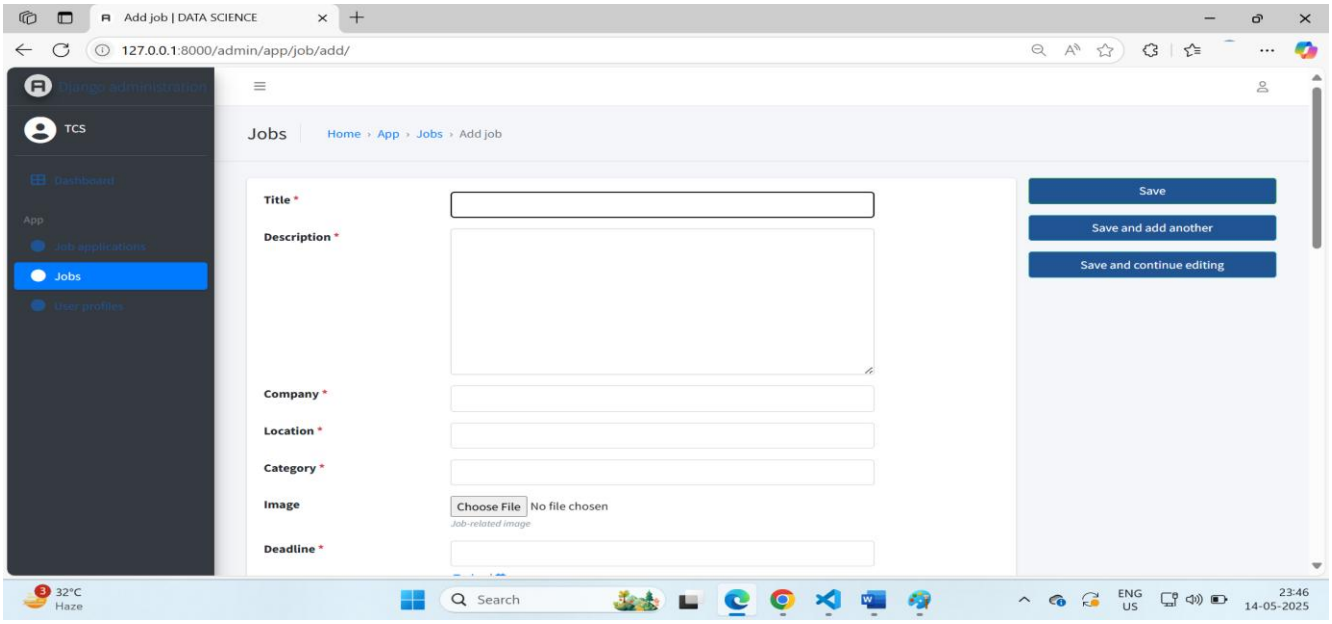

Fig. 19. Add a job information

4.2 Component Interaction:

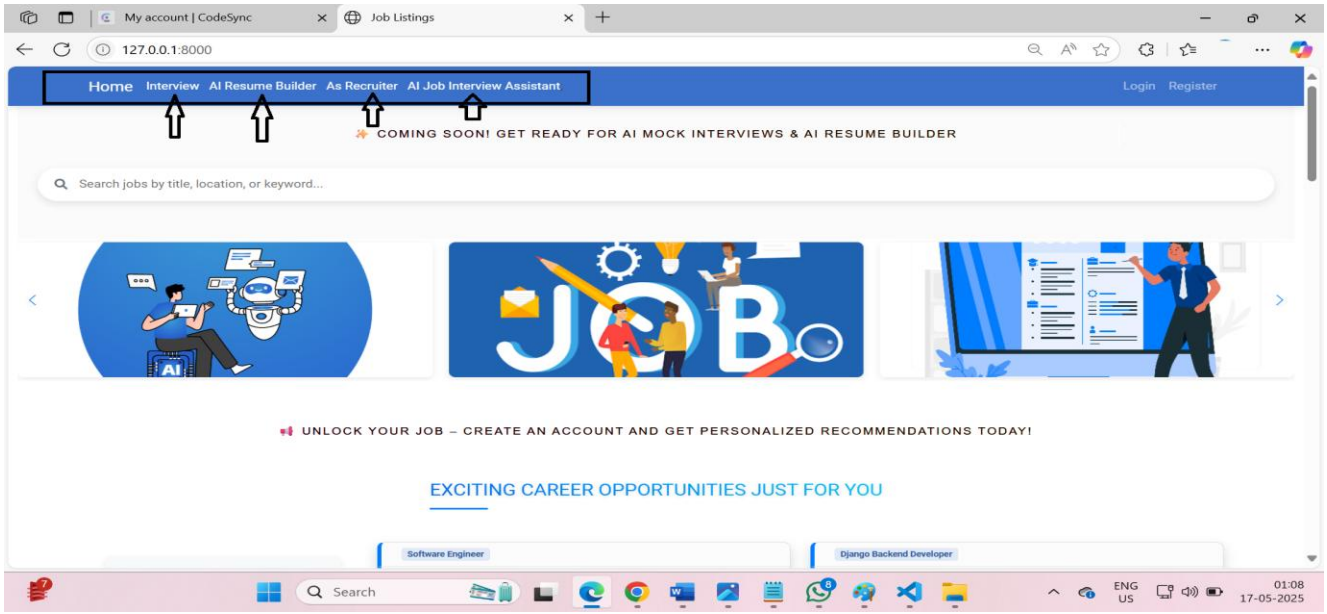

Fig. 20. Main Landing page of the website

This is the Home page of the website where user access the all service which provide our platform when user suppose user click on the AI Job Interview Assistance then new page will open and user chat.

4.3 Working of Resume Builder

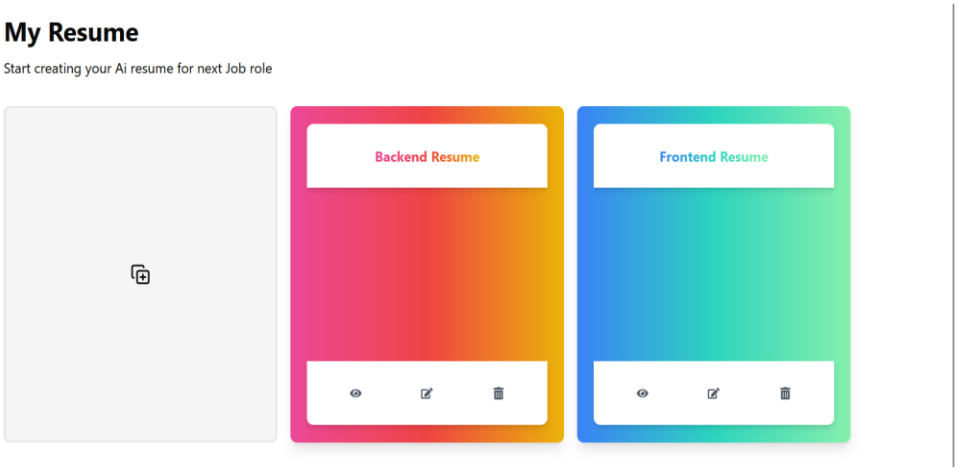

Fig.21. Dashboard of Resume Builder

When user login the redirect to the dashboard page of the resume build and here user can choose templates of the resume and click next now, fill the user information (like name education, skills, experience , certification, email and links address etc.) and if user want to suggest descript through AI then click simply generate button ai suggest summary according to user input.

4.4 Working of the AI Job interview

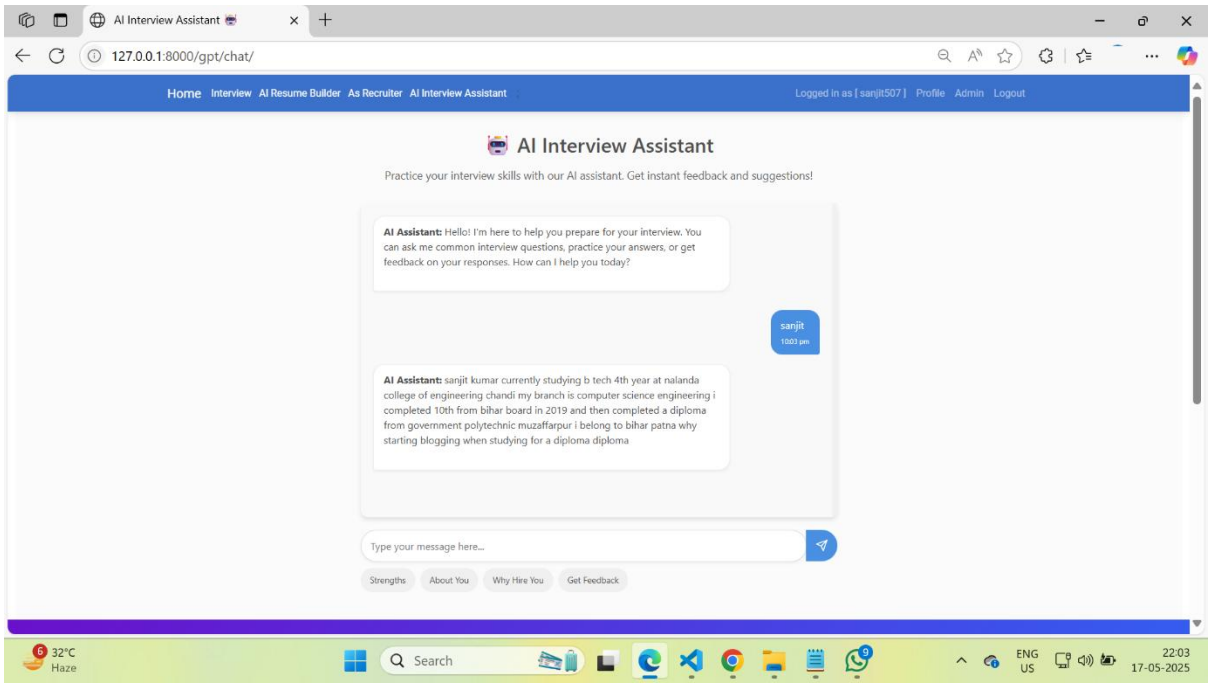

Fig.22. Interface of AI Interview Assistance

The dataset includes 'About Me' data, and the chatbot performs almost correctly during testing. However, it still needs improvements and further study. Since training the model requires more data, it also needs higher computational power.

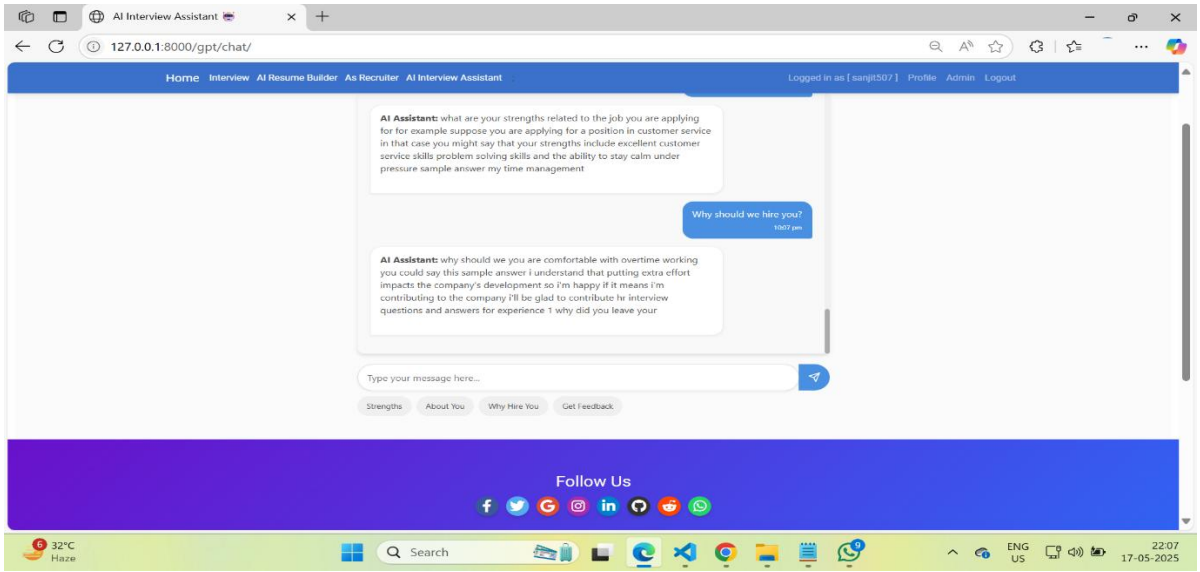

Fig.23. More Question Asked

Here are more questions based on the training dataset. The model shows good performance in the initial stage.

4.4 Working Metting

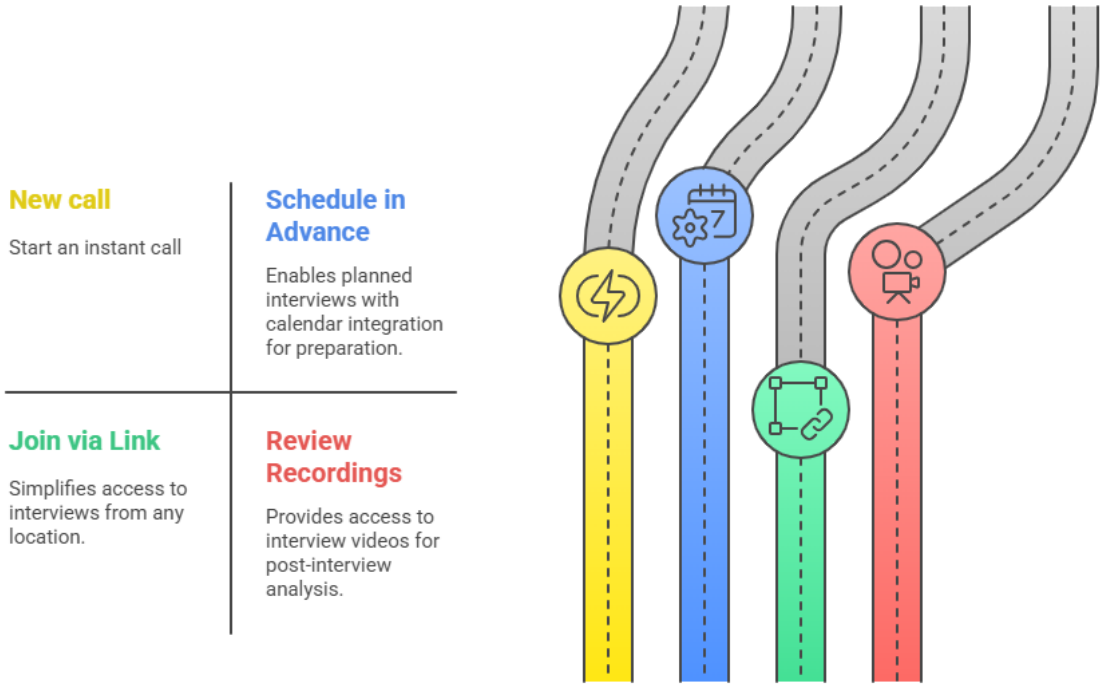

Fig.24. Working Metting

The interview system offers a comprehensive set of features designed to streamline the hiring process.

The **New Call** feature allows users to instantly initiate an on-demand interview session without prior scheduling perfect for quick assessments or spontaneous discussions.

The **Schedule** option enables users to plan interviews in advance by setting a specific date and time, sending calendar invites, and syncing with tools like Google Calendar or Outlook to ensure everyone is informed and prepared.

The **Join Interview** functionality simplifies access to scheduled or ongoing interviews via a unique invitation link, allowing participants to easily join from any location. After the interview, the **Recordings** feature provides access to the session video for review

5. Datasets

5.1 Job Recommendation System

- **Preparing Dataset** :- Here use two datasets.

  This dataset generating using Faker library total **10,000 fake** job entries fields like title, description, company, location category, created_date, deadline, salary requirements, responsibilities (mixed academic & technical contact_email, required_skills, education_level and stored in **Django using SQLite database** using **export and import library**.

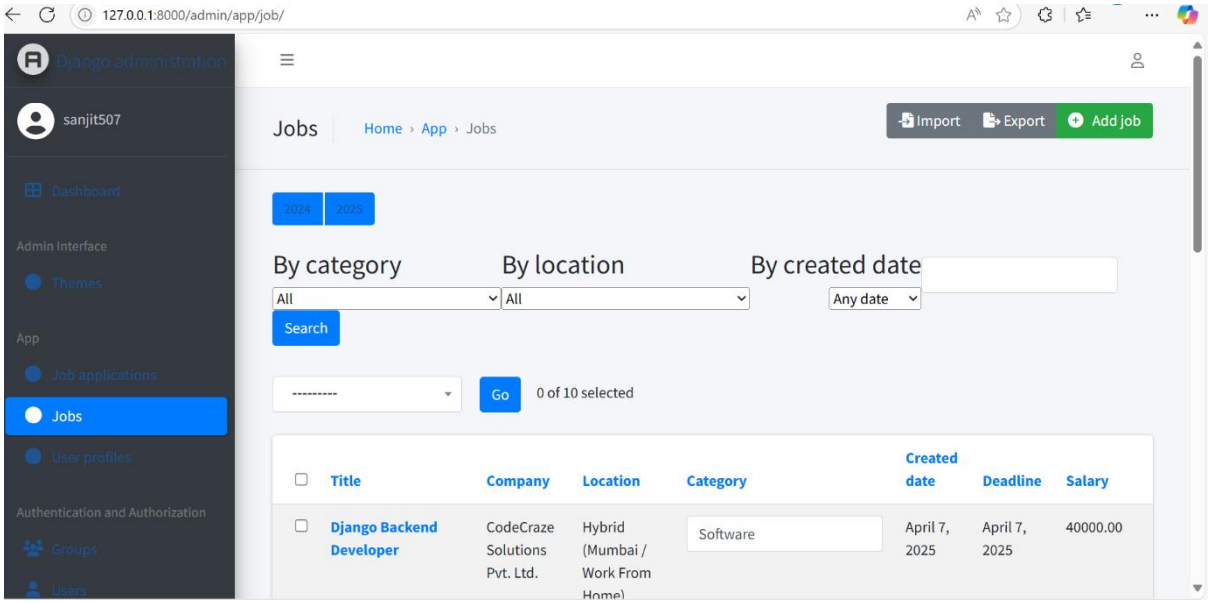

Fig. 25. Screen short of stored in SQLite Database

This is the admin panel of website where stored the job data which help to provide the job information to the users.

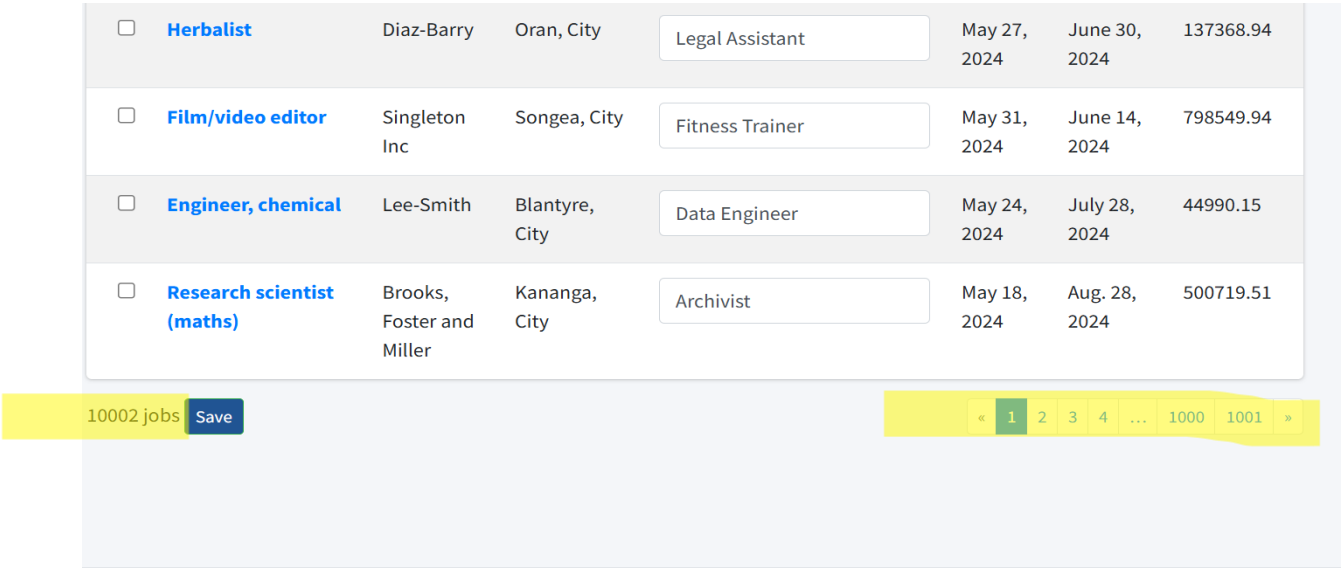

Fig. 26. Show the full database job entries

Why use two datasets:-

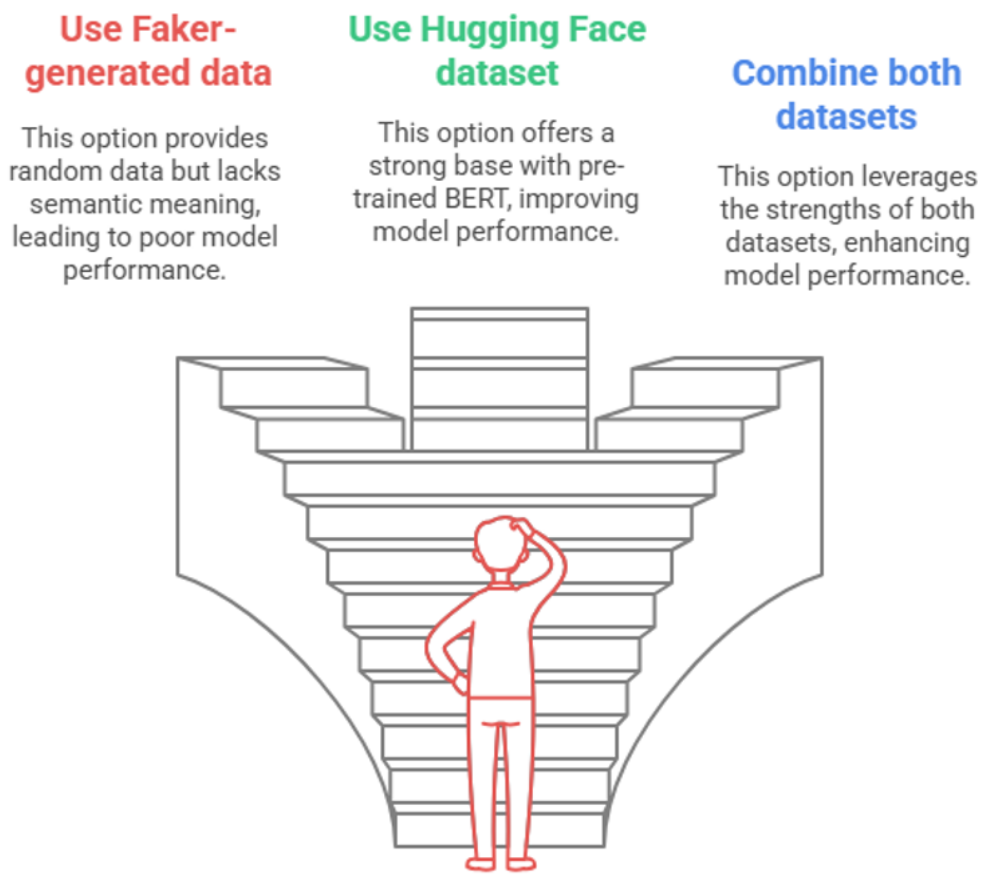

Fig. 27 Why take two Datasets

When building a real-time working web application for a job website, you need all the information about the job **like title, description, company, location, category, created_date, deadline, salary, requirements, responsibilities, contact_email, required_skills, education_level etc.** to function in **real time**. I tried to find such a dataset on the internet but failed to find one with all these fields.

I first tried fine-tuning the model using this dataset, but the model performance was poor, even when using different algorithms, the Adam optimizer, and experimenting with different learning rates and epochs. After further study, I concluded that the data generated by the Faker library is random and does not capture the semantic meaning of the data. Below, all the experiments are provided.

Next, I combined both CSV datasets for training the model. The model performance improved because the Hugging Face dataset provided a strong base. Since BERT is pre-trained on massive corpora (like Wikipedia, books, and internet data), it can tolerate a moderate amount of low-quality generated data

5.2 AI Resume Builder:-

At runtime, the user provides input data such as name, qualifications, skills, education details, programming languages, and work experience. When the user clicks the **"Generate"** button, the system uses the **Google Gemini API key** to generate a personalized description based on the provided information.

5.3 AI Interview Assistant :-

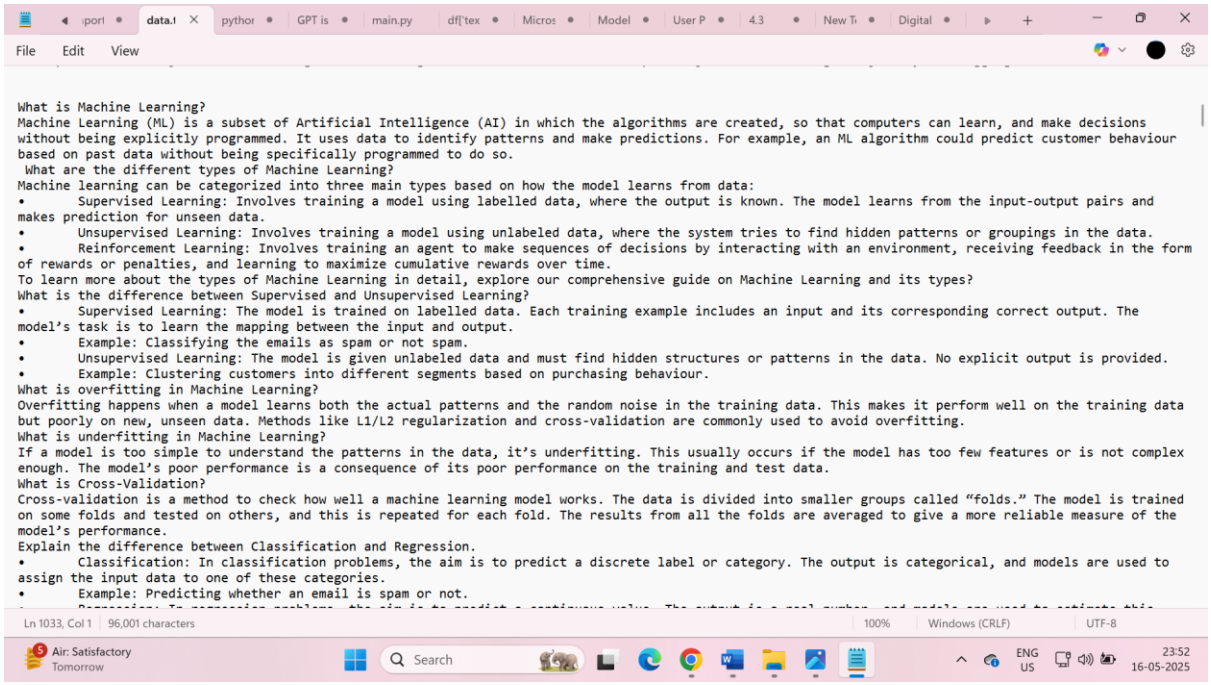

Fig. 28 TXT Dataset for taring of the AI Interview Model

The training data is in .txt format and includes content related to Software Engineering, Human Resources (HR), Object-Oriented Programming (C++), Machine Learning, as well as interview questions and answers. Based on this dataset, users can ask questions during the AI interview session, and the model will provide relevant answers according to the trained data.

5.4 Interview Application

This component serves as the interface where users can participate in a mock interview experience. Although no specific dataset is used.

6. Methods

6.1 Job Recommendation System

- ❖ Model: Fine-tuned BERT

(a) Data Exploration and Visualization

In this step, both CSV datasets were explored. The generated dataset contains **10,000 rows and 14 columns**, while the Hugging Face dataset contains **17,880 rows and 18 columns**. The shape and structure of both datasets were examined.

(b) Data Preprocessing

Data preprocessing included the removal of duplicate entries, checking for null values, and handling missing data where necessary.

(c) Feature Engineering

Important columns were merged into a single input text column using the following fields: **text = title + description + responsibilities + required_skills + education_level** Subsequently, data cleaning was performed using **WordNetLemmatizer** for text normalization.
Additional steps included:

- Encoding labels (for classification tasks)
- Handling class imbalance
- Splitting the dataset into **80% training** and **20% testing**

(d) Use of Pretrained Model:- The **pretrained model** Sentence Transformer(sentence-transformers/all-MiniLM-L6-v2') was used to train on the pre-processed data. This model was selected for its efficiency in generating high-quality semantic sentence embeddings.

Model Name	sentence-transformers/all-MiniLM-L6-v2
Tokenizer	bert-base-uncased tokenizer
Base Model	Microsoft's MiniLM
Architecture	MiniLM (6 Transformer layers)
Embedding Dimension	384
Typical Datasets Used	SNLI, MultiNLI, STSbenchmark, Quora QP, Reddit, WikiAnswers, StackExchange
Model Size	80 MB
Training Data Volume	1 billion

6.2 AI Resume Builder

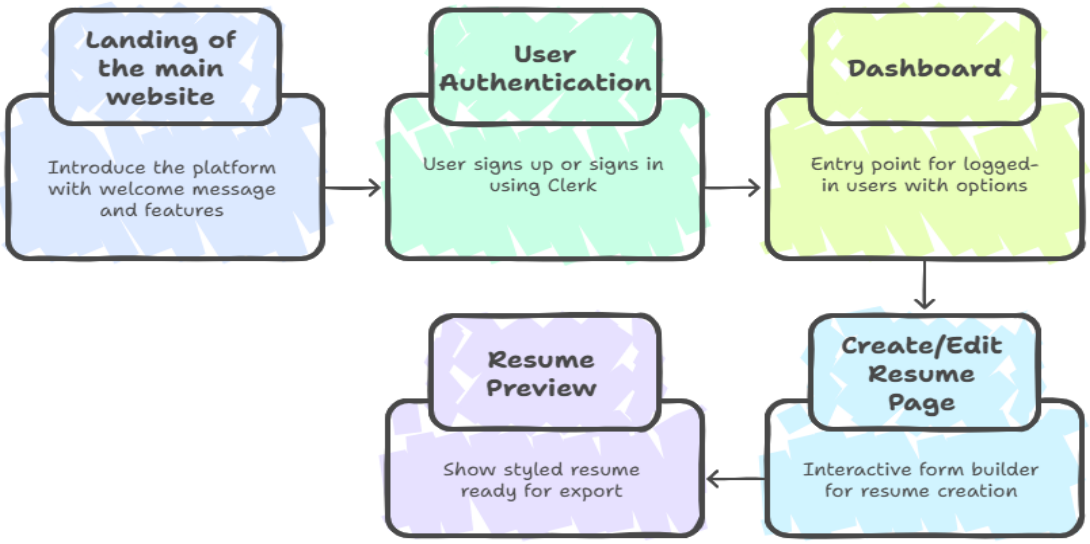

Fig. 29. AI Resume Builder Overview

The Landing Page provides an overview of the entire platform, including features like job listings, the AI Interview Chatbot, the Meeting Application, and the AI Resume Builder. If the user is not logged in and tries to access a specific tool such as the resume builder, they will be prompted to sign up or log in first. Once logged in, they are redirected to the Dashboard of the Resume Builder. This page highlights the advantages of using the AI-powered resume builder and encourages users to start creating their resume. After authentication, users are directed to the Template Selection page, where they can browse through a range of professionally designed resume templates and select one that best fits their style and career needs. Once a template is chosen, users proceed to fill in their personal and professional information. AI-driven suggestions help users fill in sections such as work experience, education, and skills. These prompts make resume writing easier and more effective.

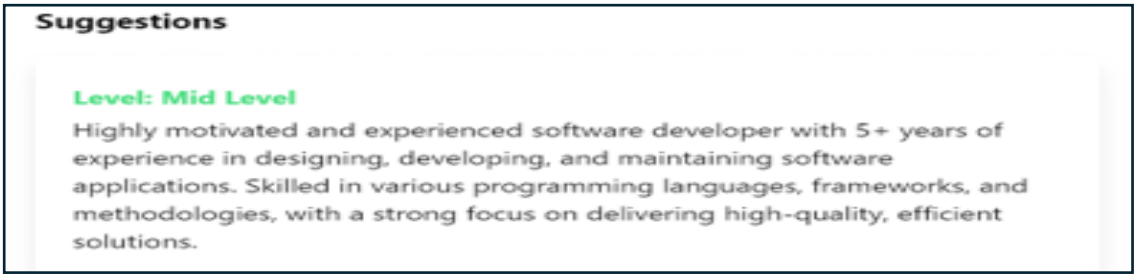

Fig.30. AI-driven suggestions help users fill in sections such as work experience, education, and skills. These prompts make resume writing easier and more effective.

6.3 AI Interview Assistant

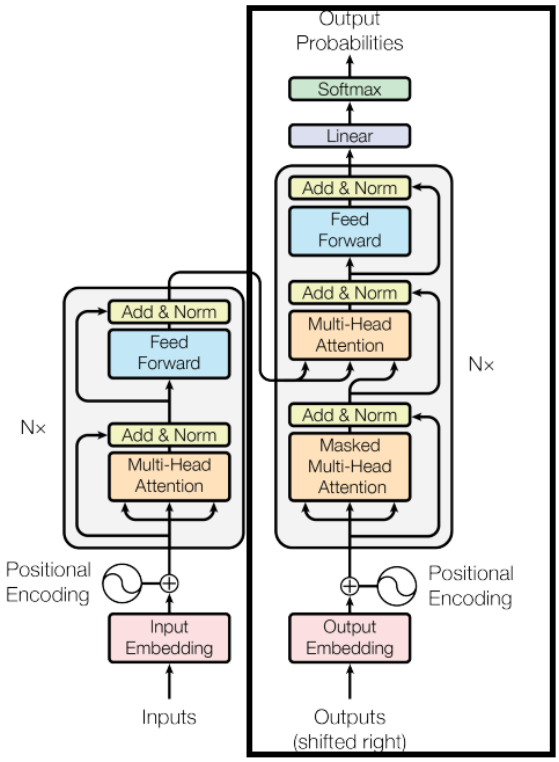

Fig.31 Follow this architecture to tarin the model

This is the official paper Attention Is All You Need of the transformer which include two part one is encoder and second is decoder I will use decoder part to build the AI Interview chatbot.

The first step was to load a plain text file and break it down into tokens (words or subwords). I used Keras built-in Tokenizer for that limiting the vocabulary size to 5,000 and using <OOV> to handle any out of vocabulary words.

Once the full text was tokenized created training data using a sliding window approach. For each chunk of 100 tokens the model is trained to predict the next 100 tokens, shifted one position forward. This setup helps the model learn the relationship between consecutive words.

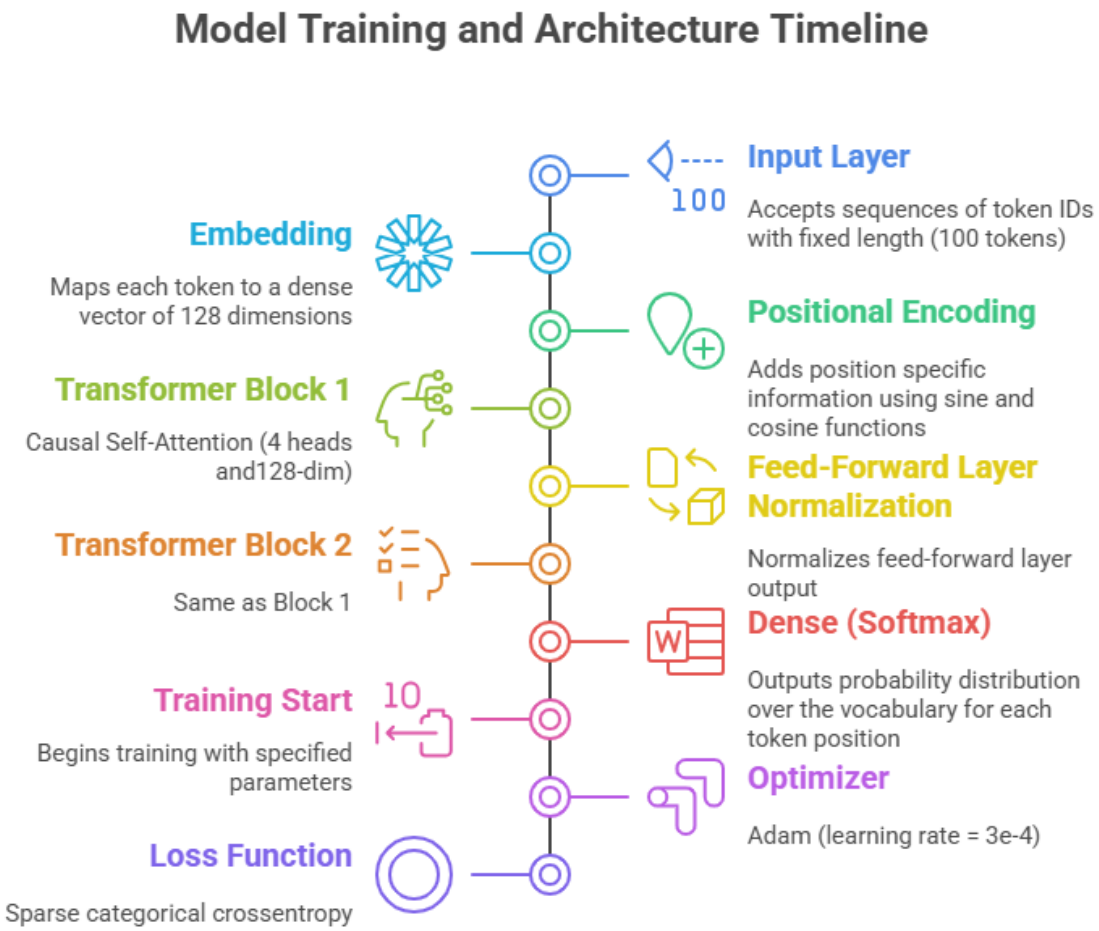

Fig. 32 Architecture of Model taring

Input Layer that accepts token sequences of fixed length (100 tokens). Each token is then mapped to a dense vector of 128 dimensions through an Embedding layer. Positional Encoding is added next, using sine and cosine functions to capture the order of tokens in the sequence.

The data passes through two identical Transformer Blocks , each containing causal self-attention (with 4 heads and 128-dimensional vectors), followed by a Feed-Forward Layer and Normalization to stabilize training. After processing, the output goes through a Dense Layer with a Softmax activation to produce probability distributions over the vocabulary for each token position.

Training starts with specified parameters, and the model's performance is optimized using Adam optimizer (learning rate = 3e-4) while minimizing the Loss Function (sparse categorical cross-entropy).

6.4 Interview Application

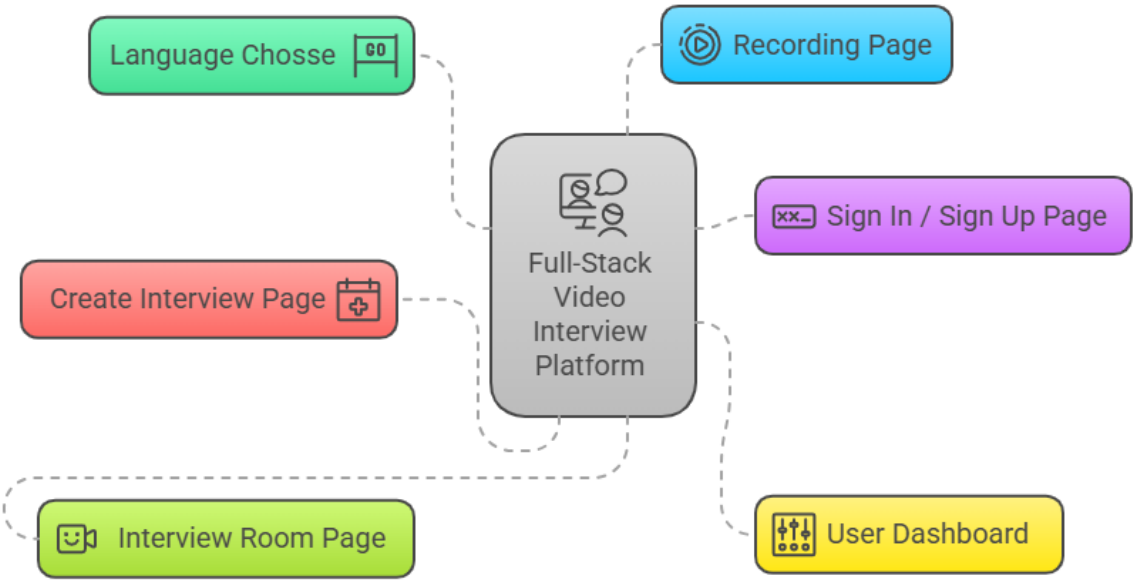

Fig.33. User Flow and Interface Components in the Application

When users first open the platform, they start at the Sign In / Sign Up Page , where they can either log in to their existing account or create a new one. Once authenticated, they're taken to the User Dashboard the central hub where they can manage their interviews, view schedules, and access different features. From the dashboard, users can go to the Language Selection Page to choose their preferred programming language, like JavaScript, Python, or Java, depending on the interview requirements.

If they want to start a new interview, they can use the Create Interview Page to set everything up. When it's time for the actual interview, the Interview Room Page offers a real-time space with tools for video chat, screen sharing, and coding. After the session, users can head to the Recording Page to review or save the interview.

7. Implementation Details

7.1 Recommendation system

Fig. 34. Implementation of the Recommendation system

starting with the user triggering the recommendation process by accessing their profile. The system retrieves the user's profile data and converts it into text, which is then encoded into a vector using a fine-tuned model. Simultaneously, job embeddings are either loaded from cache or generated and indexed using FAISS for efficient similarity

search. The encoded user profile is compared to job embeddings to find the top 5 matching jobs, which are then returned and displayed to the user in the front-end interface. This workflow enables personalized and efficient job recommendations based on user preferences and qualifications.

7.2 Resume Builder

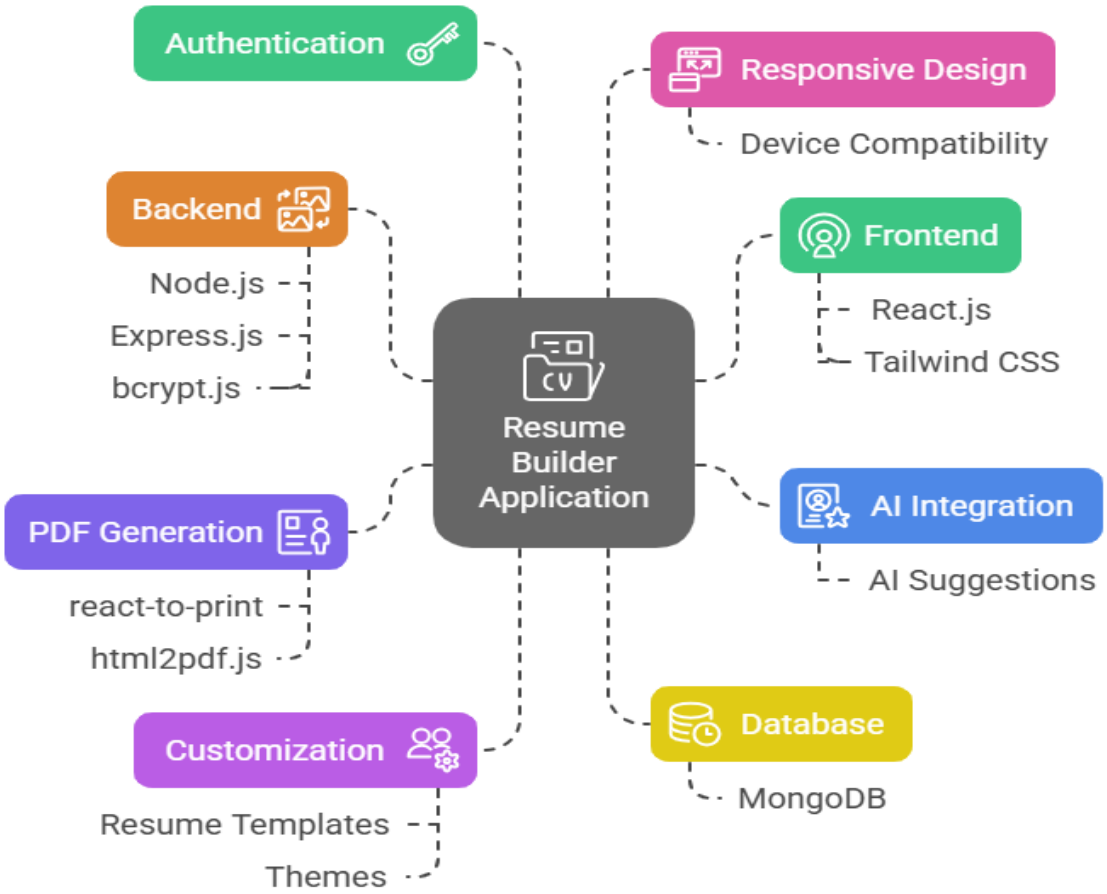

Fig.35. Implementation details of Resume Builder

The Resume Builder app is made up of a few key parts that work together to let users create and customize their resumes. On the backend , it uses Node.js with Express.js — this basically means it has a strong foundation for handling requests, running business logic, and connecting everything behind the scenes. The frontend is built with React.js , which makes the interface smooth, fast, and easy to use on any device. For storing user data and resume details safely, it relies on MongoDB , a flexible and scalable database. All together, these tools create a reliable and user-friendly resume-building experience.

7.3 AI Job Interview

I build a small, GPT-style language model using TensorFlow and Keras. The model is designed to generate text by predicting the next word in a sequence, trained entirely on a custom dataset(software engineering and HR, C++ interview questions , college data (NCE Chandi) from a single text file. It uses some of the same concept as larger models like GPT-2 but is much smaller and easier to run.

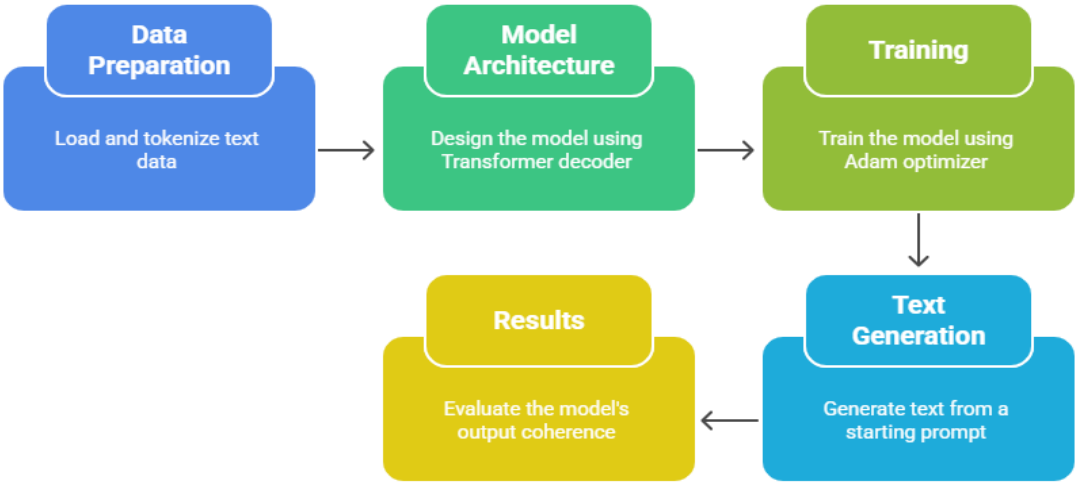

Fig. 36. Implementation of AI Job Interview

The process of building a text generation model starts with data preparation , where the first step is to load the raw text data from a given dataset. Once loaded, the text is tokenized—meaning it's broken down into smaller units like words or subwords—and then converted into numerical representations that the model can understand. This prepares the data for training in a format that the neural network can work with effectively.

Next comes model architecture , where we design the structure of the Transformer-based model. Specifically, this involves setting up a Transformer decoder, which is well-suited for text generation tasks. The decoder uses mechanisms like self-attention and positional encoding to understand context and generate coherent sequences of text.

Then, during the training phase, the model learns from the prepared dataset using the Adam optimizer , which helps adjust the model's parameters efficiently. The goal here is to minimize the difference between the model's predictions and the actual target text, allowing the model to gradually improve its ability to generate meaningful responses.

Once trained, the model moves into the text generation stage. Here, it takes a starting prompt as input and begins generating text by predicting one word at a time based on the context it has learned. Various decoding strategies can be used, such as greedy search, beam search, or sampling techniques, depending on whether we want more deterministic or creative outputs.

7.4 Metting

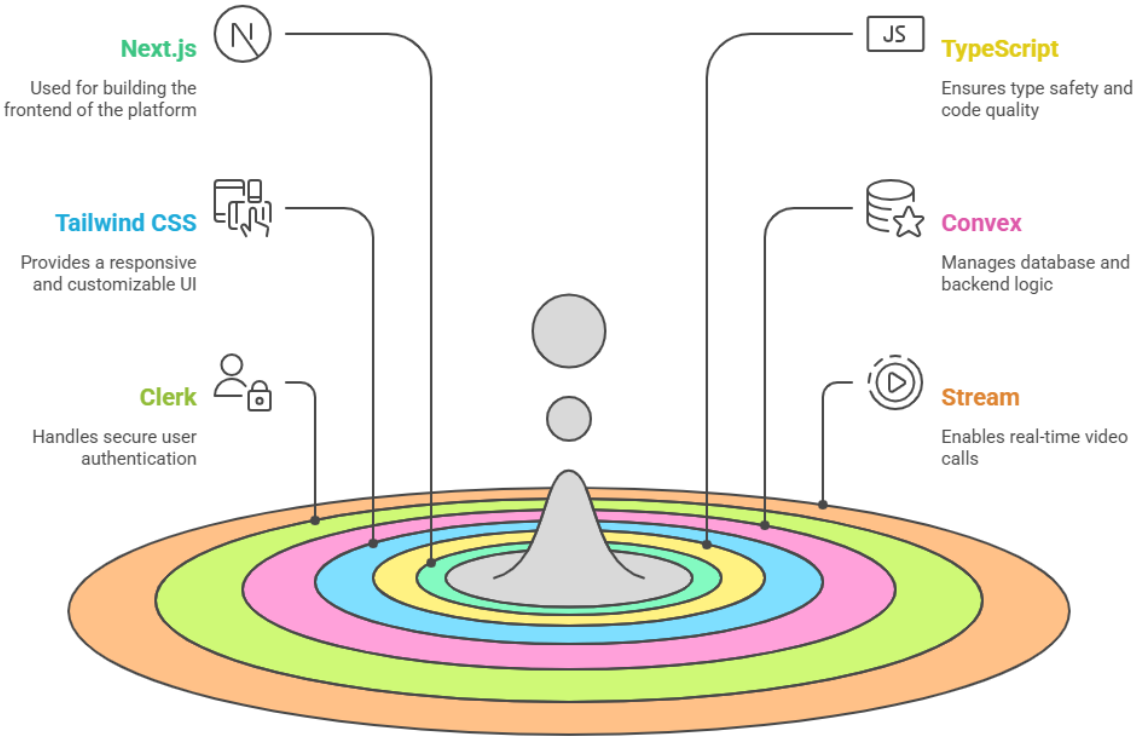

Fig.37. Implementation details meting application

Convex serves as the backend and database layer, providing real-time data syncing and serverless functions; TypeScript adds type safety for reliable code; Stream powers real-time video calls for interactive features; Next.js builds the frontend with fast performance and SEO benefits; Tailwind CSS styles the UI with a clean, responsive design; and Clerk handles secure user authentication. Together, these tools create a scalable, real-time app with a smooth and secure user experience.

8. Results & Evaluation

8.1 Job Recommendation

Training the model using own datasets:- First tarin the faker datasets and then hugging face and then combined taring on both CSV, Below the summary table of the own model name is **Job-Rec-V1.**

Dataset	Taring Data	Epoch	Accuracy	F1	Precision	Recall
Generated	10000	100	0.1556	0.1513	0.1523	0.1557
Hugging face	10000	10	0.9657	0.9640	0.9676	0.9657
Combined	27000	10	0.8517	0.8521	0.8784	0.8518

```
Epoch 10/10 - Accuracy: 0.8518, F1: 0.8521, Precision: 0.8784, Recall: 0.8518

Final Evaluation:
Accuracy: 0.8517815208882962
Weighted Precision, Recall, F1:
Precision: 0.8784, Recall: 0.8518, F1: 0.8521
```

Fig. 38. Training of the combined CSV Data

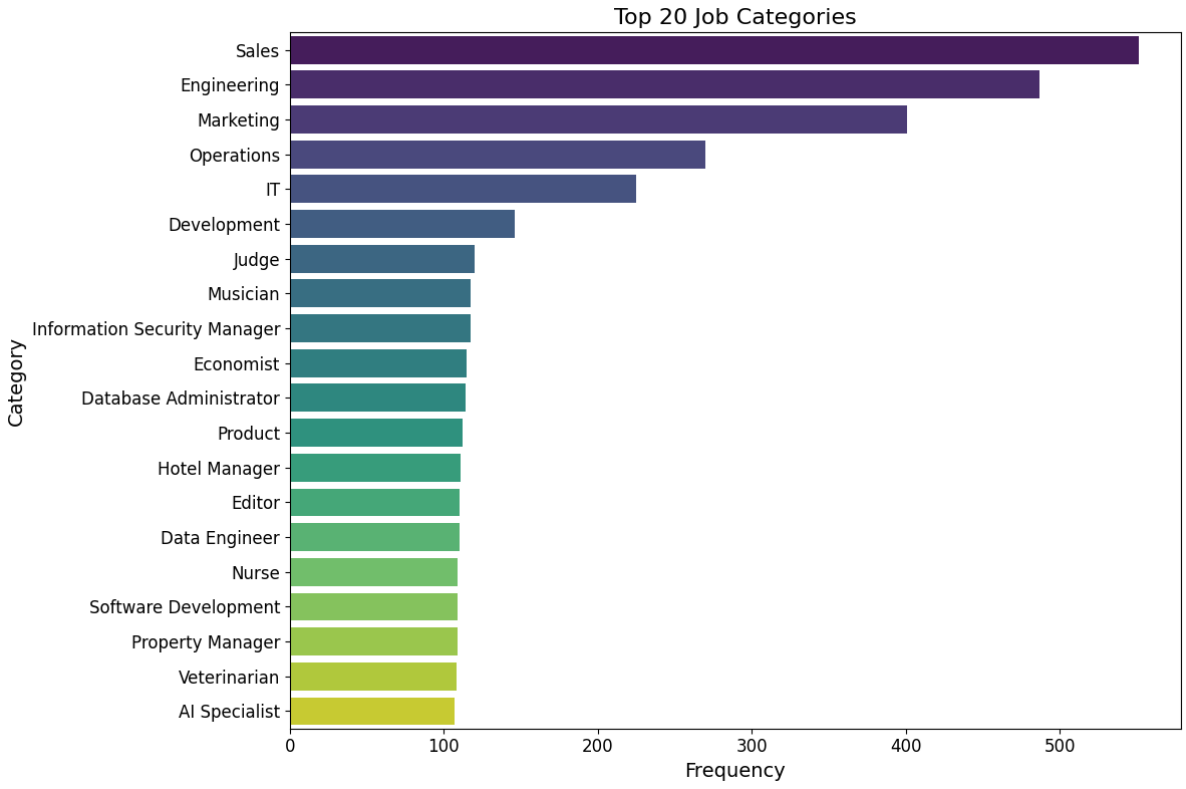

Fig.39. Top 22 Categories

Confusion matrix

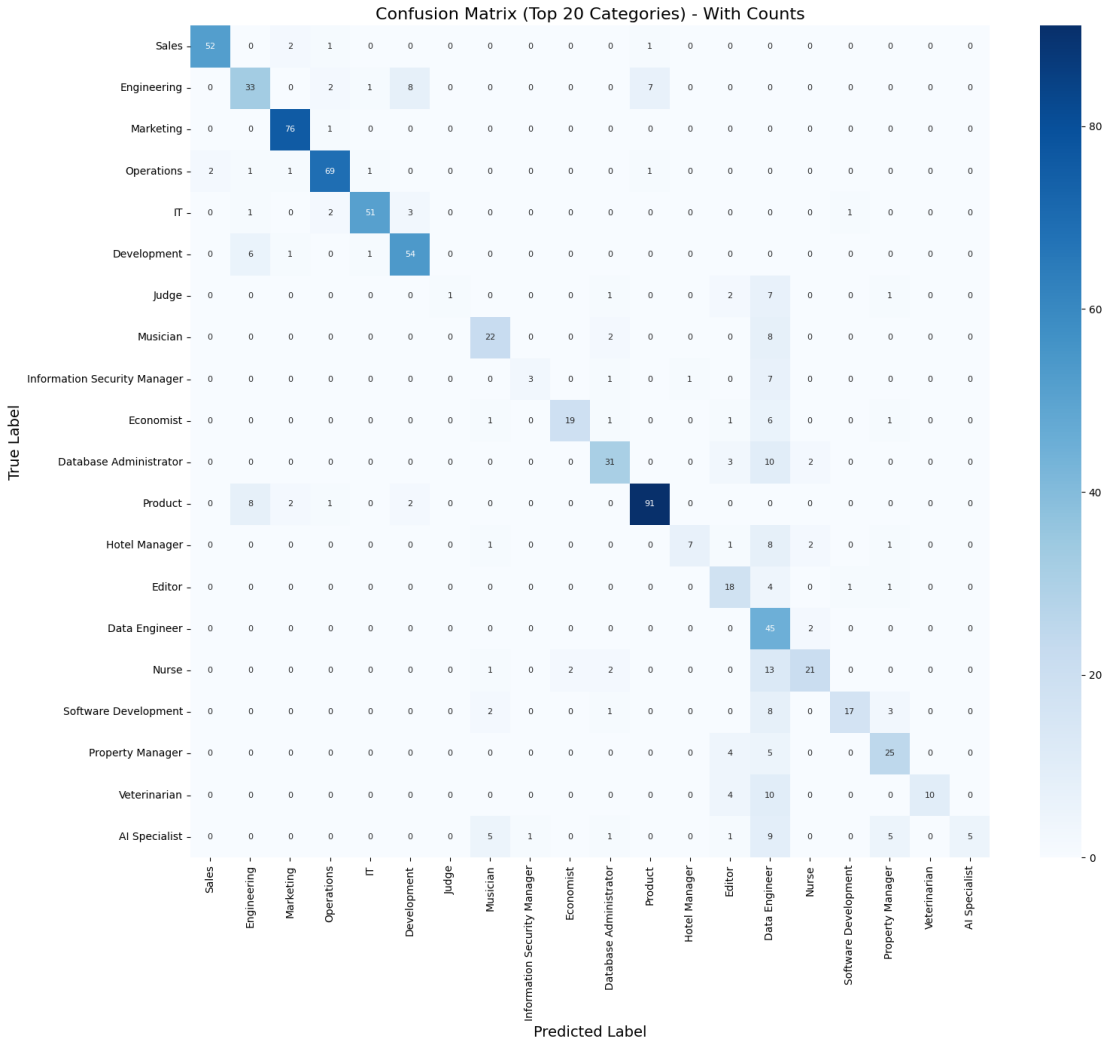

Fig. 40. Confusion matrix

Confusion matrix show the performance of model that classifies job titles into categories. On the x-axis , we have the model predicted labels (what the model thinks the job category is), and on the y-axis , the true labels (what the actual category is).

Each cell shows how many times a particular true category was predicted as a specific label for example, how often "Sales" was correctly predicted as "Sales" or mistakenly labeled as "Marketing." The diagonal cells (from top-left to bottom-right) represent correct predictions, and higher values there indicate better model accuracy.

The off-diagonal cells show misclassifications, highlighting where the model is confused like mistaking "Engineering" for "Sales" or "Database Administrator" for "Economist."

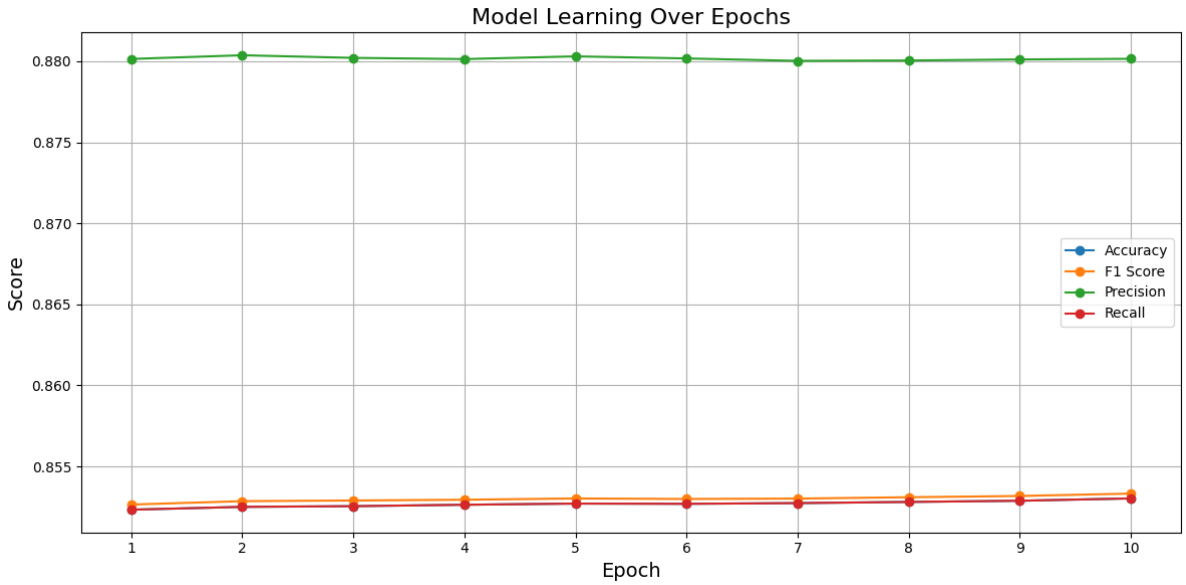

Fig. 41. Job-Rec-V1 Learning curve

The learning curve indicates that the model's performance improves steadily with additional training epochs. Precision remains robust throughout, while Recall shows notable enhancement, leading to better overall performance metrics like Accuracy and F1 Score. This suggests that the model is effectively learning from the data and converging towards optimal performance.

Job-Rec-v1 Model details

Model Name	(custom fine-tuned Sentence Transformer)
Base Model	all-MiniLM-L6-v2, fine-tuned on job-related text
Purpose	Generate semantic embeddings for user profiles and job descriptions
Input Type	user profile
Output Type	384-dimensional vector (embedding)
Training Data	Custom dataset of job titles, descriptions, skills, Responsibility and Education level
Training Total Data	27000
Model size	80.51 MB
Classification algorithm	Stochastic Gradient Descent Classifier (SGD Classifier)
Oversampling Algorithm	Random Over Sampler
Text Preprocessing	WordNet Lemmatizer
Loss Function	Log loss

Load the model in Project :- After training the model load this model in the project. When load the Job-Rec-v1 and setup profile the show real time end to end recommendation show below

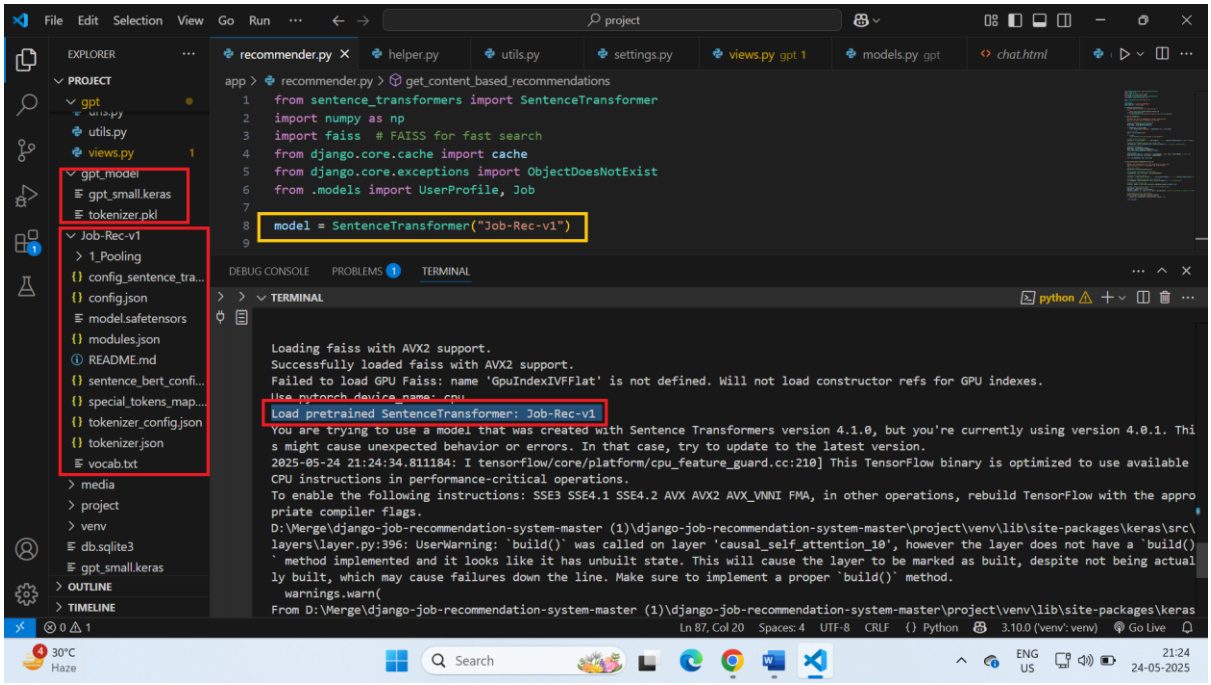

Fig.42. Load the model

Load the pretrained model in the project and show recommendation according to the user profile.

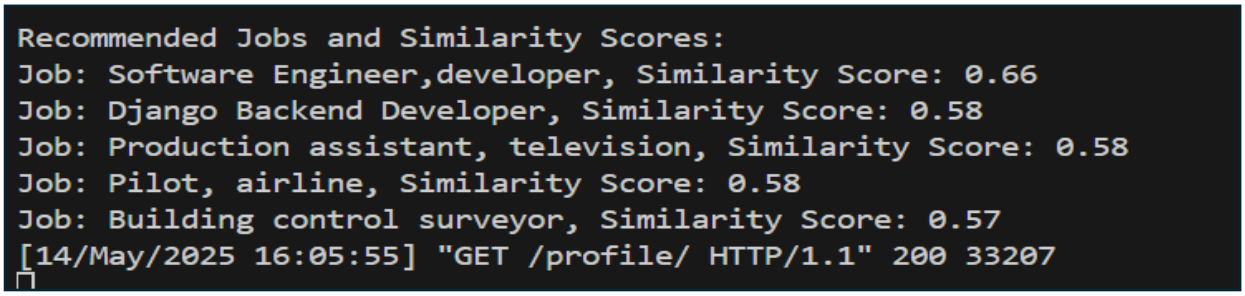

Fig. 43. Visual studio code terminal output

The job recommendations include roles like Software Engineer, developer (score: 0.66), Django Backend Developer (0.58), Production assistant, television (0.58), Pilot, airline (0.58), and Building control surveyor (0.57), likely generated by a job-matching algorithm where higher scores indicate better fit.

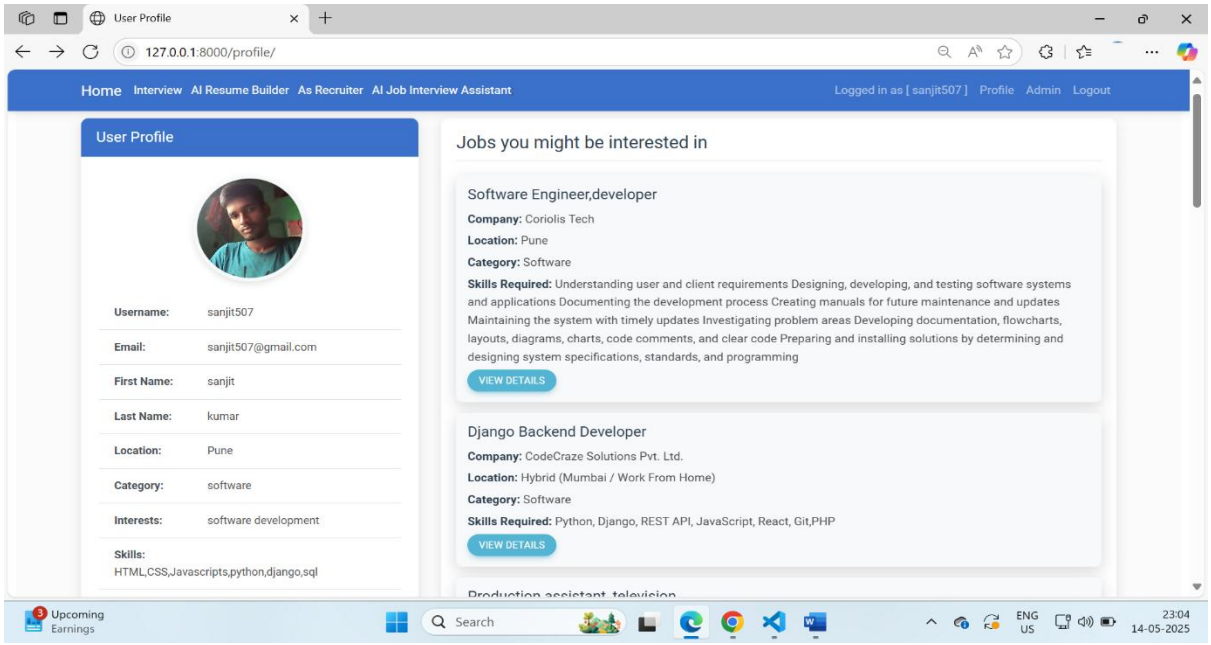

Fig. 44. Profile sction of the user

When a user signs up, their profile details (like location, category, and skills) are saved and combined into a single text string — for example: **"Pune Software HTML CSS**

JavaScript Python Django SQL" This text is converted into a numeric vector using the **Job-Rec-v1 model**.At the same time, all job listings in the database are turned into similar vectors based on their own location, category, and skills. We use **FAISS** (a fast search tool) to find the top 5 jobs whose vectors are closest to the user's — meaning they're the most similar. Finally, those 5 matching jobs are recommended to the user.

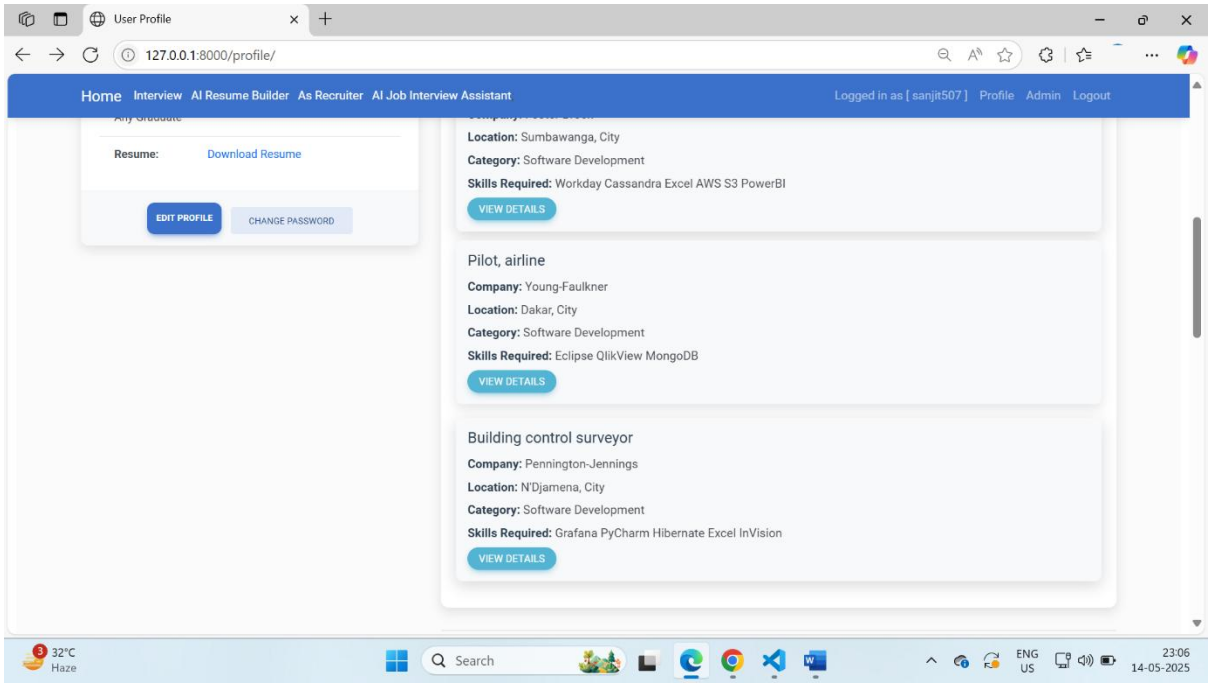

Fig. 45. Full page of user Profile

On the user dashboard, the system displays the **top five job recommendations** tailored to the user's personal details provided during registration such as their **preferred job category**, **skills**, and **location**. These recommendations are dynamically generated using an AI model that matches the user's profile with relevant job listings, helping users discover the most suitable opportunities quickly.

If the user decides to **update their profile** for example, by changing their skills, location, or job preferences the system automatically refreshes the job suggestions. The AI model re-evaluates the updated information and provides **new, more relevant job recommendations**, ensuring that the user always sees the best options based on their latest preferences.

In addition to job recommendations, the dashboard provides a comprehensive view of the user's **job application activity**. Users can track the **status of each job application**, including whether it's **pending**, **accepted**, **in the interview process**, or **selected**. These status updates are managed by the admin team, so users always have real-time insight into where they stand in the recruitment process.

The dashboard also includes a **job application history** section, where users can review all the jobs they've applied for, along with key details like the **job title**, **company**, and the **application date**. This feature helps users stay organized and informed throughout their job search journey.

8.2 Resume Builder

Download or Share Resume

Users can download their resume as a PDF or share it through links. This screen displays download options and sharing functionality.

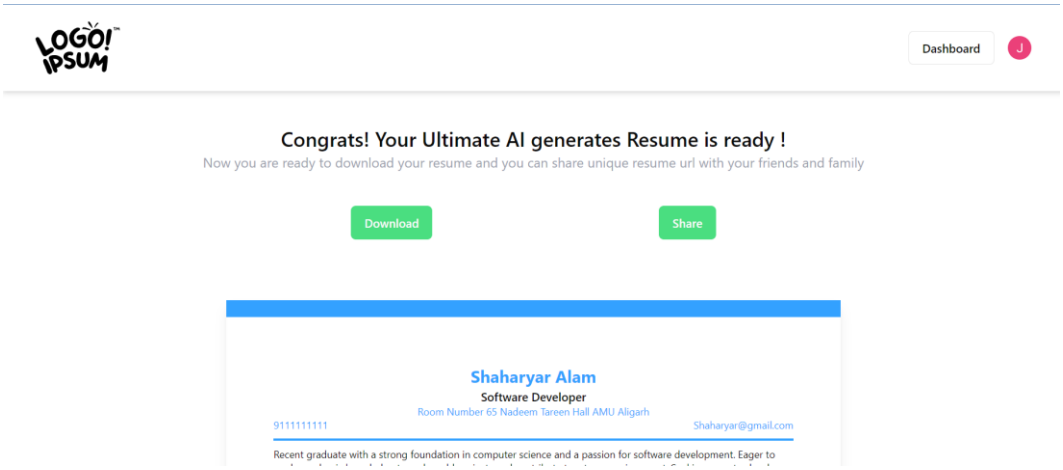

Fig.46. Users can download their resume as a PDF or share it through links. This screen displays download options and sharing functionality.

Resume Template Thumbnail

This section displays available resume templates in thumbnail form, allowing users to choose a design they prefer before editing.

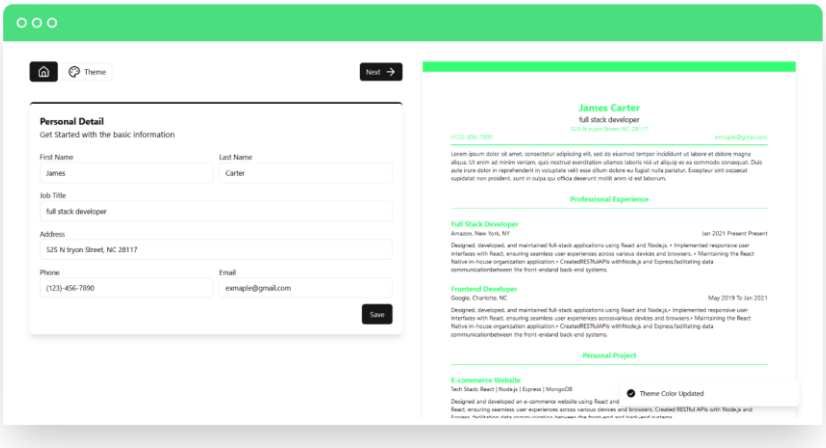

Fig.47. This section displays available resume templates in thumbnail form, allowing users to choose a design they prefer before editing.

8.3 Interview Assistant

```
Epoch 3/50
817/817 ━━━━━━━━━━━━━━━━━━━━ 9s 11ms/step - accuracy: 0.9098 - loss: 0.5643 - val_accuracy: 0.7293 - val_loss: 2.7116
Epoch 4/50
817/817 ━━━━━━━━━━━━━━━━━━━━ 10s 11ms/step - accuracy: 0.9714 - loss: 0.1915 - val_accuracy: 0.7387 - val_loss: 2.8813
Epoch 5/50
817/817 ━━━━━━━━━━━━━━━━━━━━ 10s 11ms/step - accuracy: 0.9774 - loss: 0.1268 - val_accuracy: 0.7411 - val_loss: 2.9995
Epoch 6/50
817/817 ━━━━━━━━━━━━━━━━━━━━ 10s 11ms/step - accuracy: 0.9799 - loss: 0.1016 - val_accuracy: 0.7427 - val_loss: 3.1516
Epoch 7/50
817/817 ━━━━━━━━━━━━━━━━━━━━ 10s 10ms/step - accuracy: 0.9813 - loss: 0.0886 - val_accuracy: 0.7412 - val_loss: 3.2957
Epoch 8/50
817/817 ━━━━━━━━━━━━━━━━━━━━ 9s 10ms/step - accuracy: 0.9825 - loss: 0.0793 - val_accuracy: 0.7411 - val_loss: 3.3736
Epoch 9/50
817/817 ━━━━━━━━━━━━━━━━━━━━ 9s 11ms/step - accuracy: 0.9830 - loss: 0.0743 - val_accuracy: 0.7448 - val_loss: 3.4461
Epoch 10/50
817/817 ━━━━━━━━━━━━━━━━━━━━ 10s 10ms/step - accuracy: 0.9839 - loss: 0.0692 - val_accuracy: 0.7448 - val_loss: 3.5691
Epoch 11/50
817/817 ━━━━━━━━━━━━━━━━━━━━ 10s 11ms/step - accuracy: 0.9845 - loss: 0.0656 - val_accuracy: 0.7428 - val_loss: 3.6451
Epoch 12/50
817/817 ━━━━━━━━━━━━━━━━━━━━ 10s 11ms/step - accuracy: 0.9845 - loss: 0.0644 - val_accuracy: 0.7416 - val_loss: 3.7224
Epoch 13/50
...
Epoch 49/50
817/817 ━━━━━━━━━━━━━━━━━━━━ 10s 11ms/step - accuracy: 0.9887 - loss: 0.0414 - val_accuracy: 0.7512 - val_loss: 4.7352
Epoch 50/50
817/817 ━━━━━━━━━━━━━━━━━━━━ 9s 10ms/step - accuracy: 0.9888 - loss: 0.0412 - val_accuracy: 0.7513 - val_loss: 4.7762
```

Fig.48. Model Accuracy

The model achieved 98% accuracy, which is very good. Only a small dataset was used to train the model. If more data were used, the model's performance could improve even further. However, training with more data requires greater computational power, which I currently don't have. I used a free Google Colab account to train the model, but the free version has limitations on compute resources, especially GPU usage. To train with more data, upgrading to a paid Colab plan may be necessary. Despite these limitations, the

model is a solid foundation for understanding the basic concepts of GPT and how a chatbot works

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 100)	0
embedding (Embedding)	(None, 100, 128)	338,560
positional_encoding (PositionalEncoding)	(None, 100, 128)	0
functional_1 (Functional)	(None, 100, 128)	330,240
functional_3 (Functional)	(None, 100, 128)	330,240
dense_4 (Dense)	(None, 100, 2645)	341,205

Total params: 4,020,737 (15.34 MB)
Trainable params: 1,340,245 (5.11 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 2,680,492 (10.23 MB)

Fig.49. Model Summary

input layer that accepts sequences of length 100, followed by an embedding layer that maps tokens into 128-dimensional vectors (338,560 parameters), and a positional encoding layer that adds sequence information without trainable parameters. It includes two functional layers (**functional_1** and **functional_3**), each with 330,240 trainable parameters, likely implementing attention or similar operations.

A final dense layer outputs 2645 features (341,205 parameters), bringing the total trainable parameters to ~1.34 million and total model parameters to ~4 million.

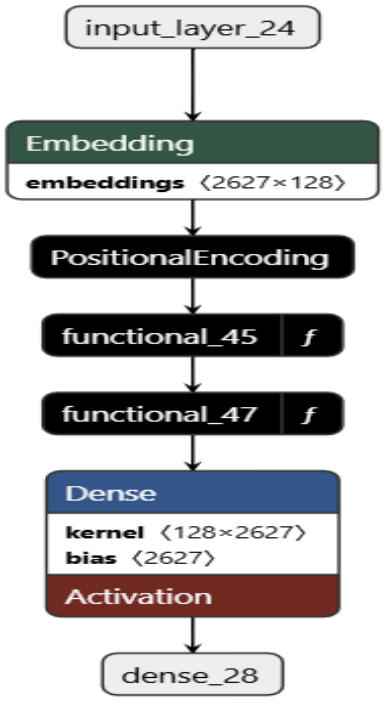

Fig.50 Visualization of the Model

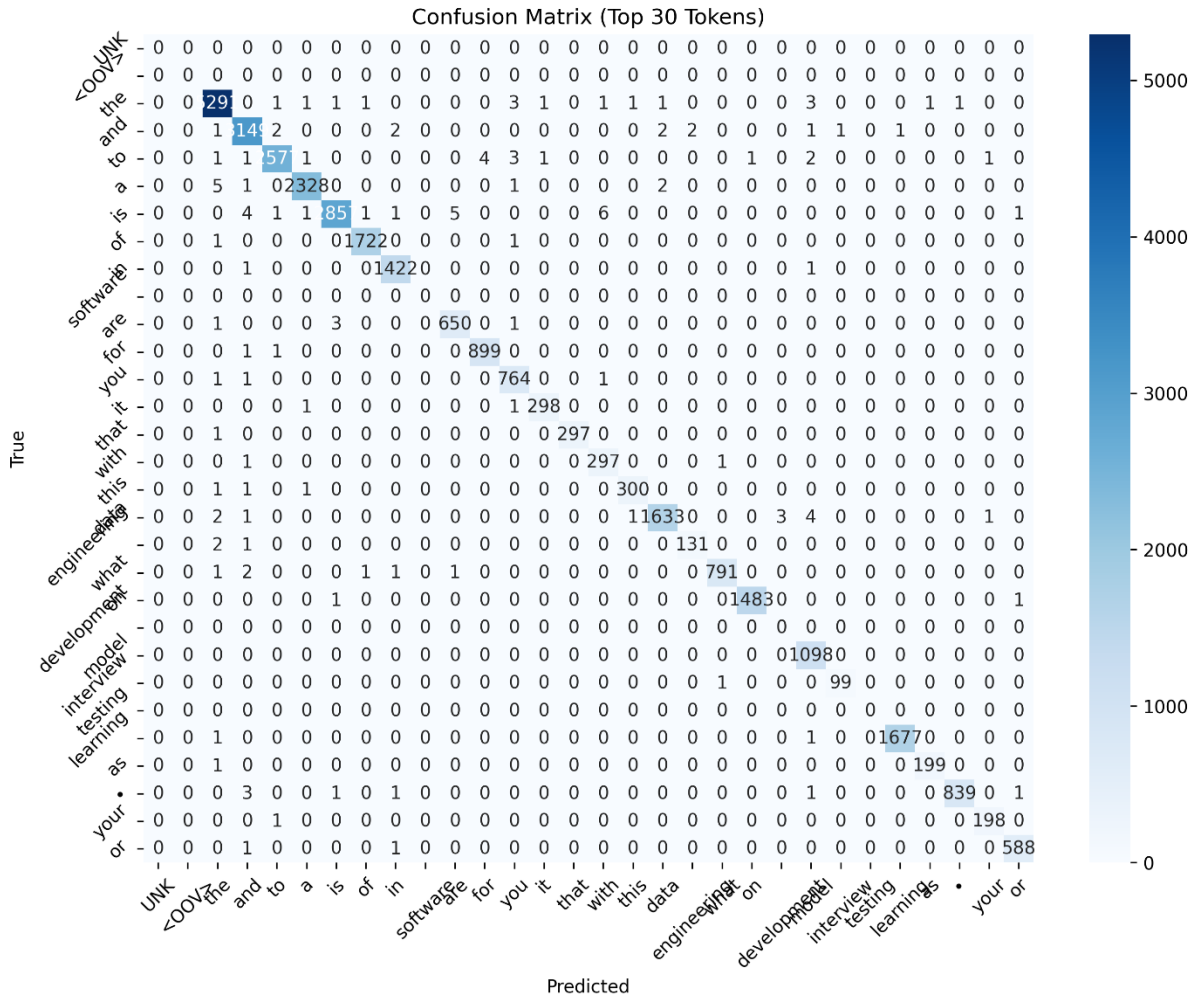

Fig.51. Confusion Matrix

The confusion matrix shown is for the top 30 tokens in a classification task. It compares the true labels (actual categories or tokens) against the predicted labels (what the model output). Each cell in the matrix represents the number of times a specific true label was classified as a particular predicted label.

model predicts the top 30 tokens by comparing true labels with predicted labels. The diagonal shows correct predictions (**development** and **interview** are predicted accurately), while off-diagonal values highlight misclassifications (**the** being confused with **and**, or **software** with **are**).

Overall, the model's performance may be limited due to a relatively small training dataset. By increasing the size of the dataset, adding more transformer layers, and using more advanced optimization techniques or algorithms, the model can learn better representations and achieve higher accuracy.

In short, with more data, deeper architecture, and smarter training methods, the results can be significantly improved.

```python
# Generate new text
seed_text = "sanjit"
generated = generate_text(seed_text=seed_text, model=model, tokenizer=tokenizer, num_tokens=30, temperature=0.7)
print("Generated Text:")
print(generated)
```

Python

```
Generated Text:
sanjit kumar currently studying b tech 4th year at nalanda college of engineering chandi my branch is computer science engineering i complete
```

Fig.52. Generating text to tarin the model

This code takes a starting word ("sanjit") and uses your trained model to **predict and generate 30 new words or tokens** that continue the sentence in a natural way. The quality of the output depends on how well your model is trained and the size of your dataset but overall output is good.

- num_tokens=30: Tells the model to generate 30 new tokens (words/subwords).
- temperature=0.7: Controls randomness.
- tokenizer=tokenizer: The tokenizer used to convert text into tokens (numbers) that the model understands, and back to text

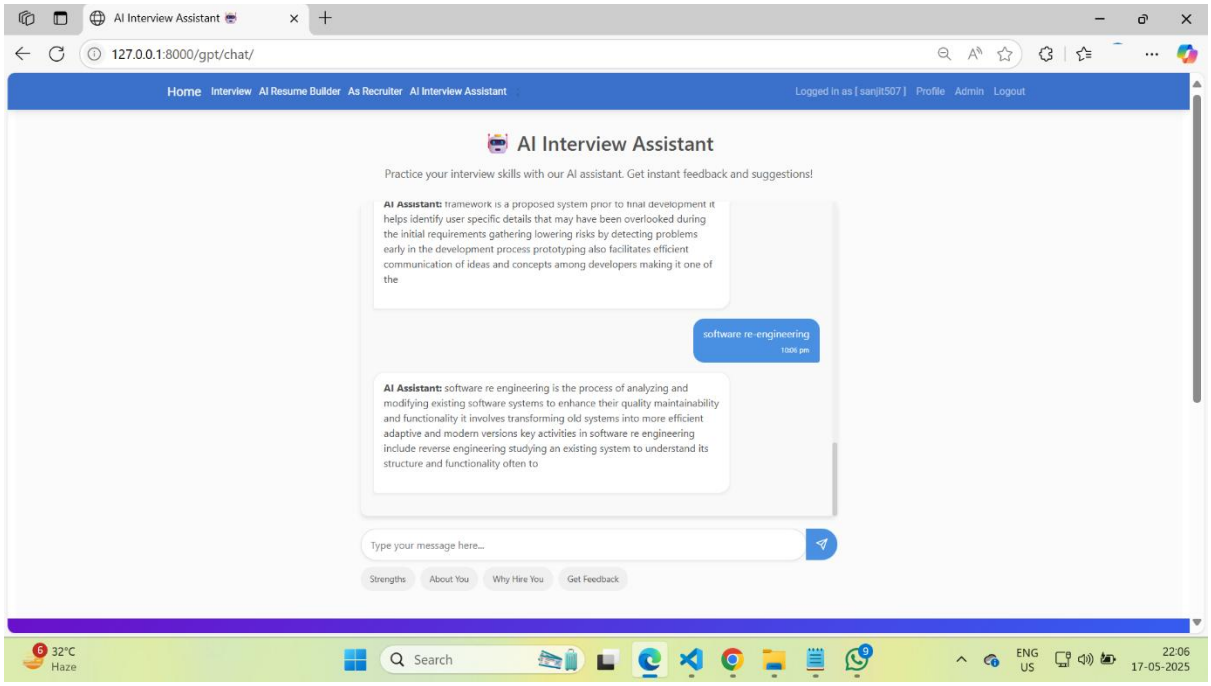

Fig.53. Frontend Result

The user types a message on the web page, and JavaScript sends this input asynchronously via an AJAX POST request to the Django backend. The Django view receives the message and passes it to the generate text () function, which uses the loaded chatbot model to create a response. This response is sent back to the frontend as a JSON object, where JavaScript parses it and updates the chat window to display the chatbot reply

8.4 Interview App

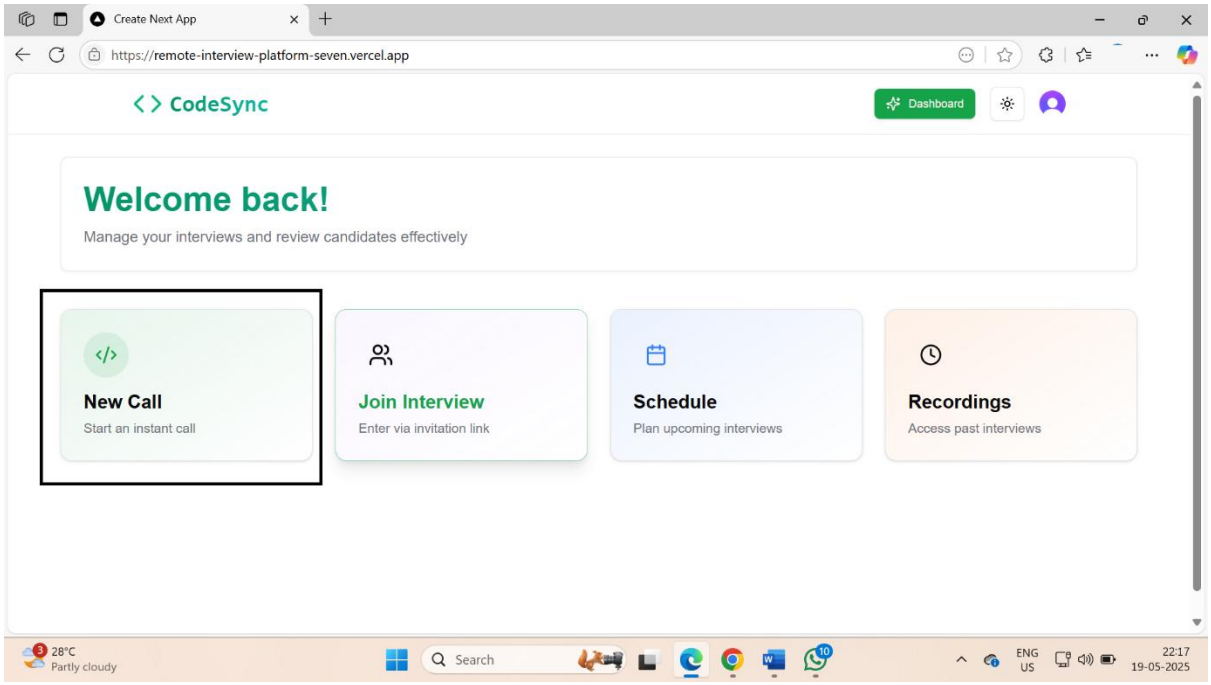

Fig.54. Dashboard of the meeting app

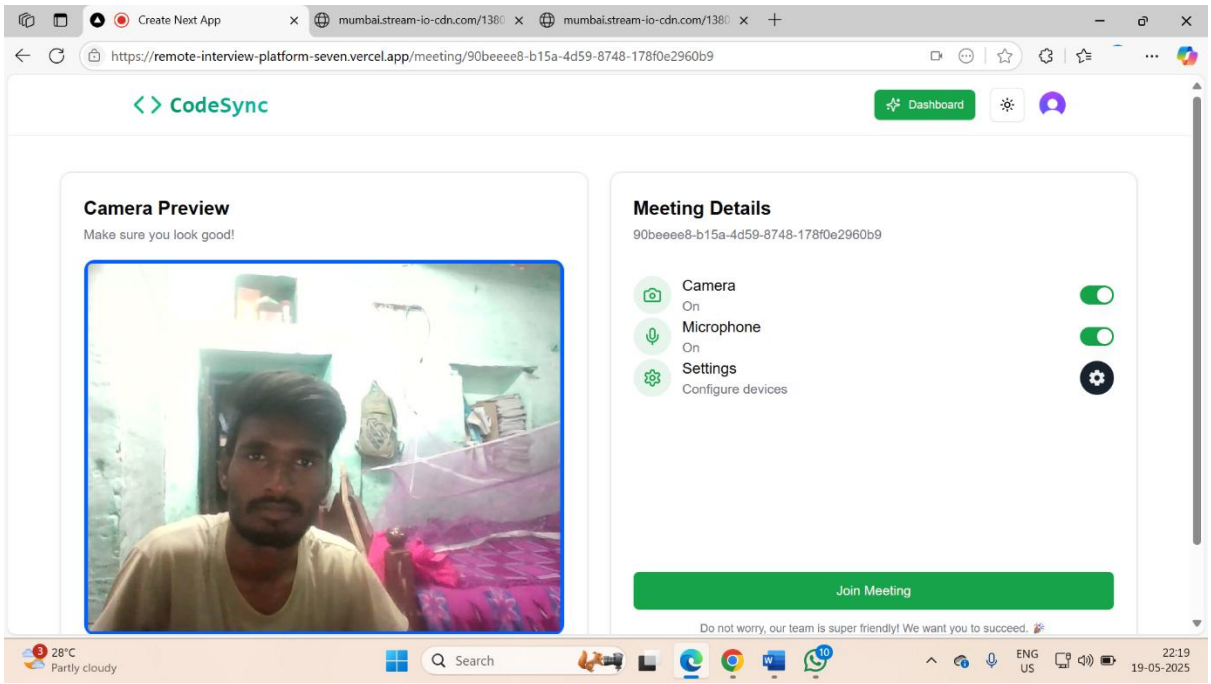

Fig.55. Meeting interface

Process of launch the meting camera and microphone and other setting enable or disable here.

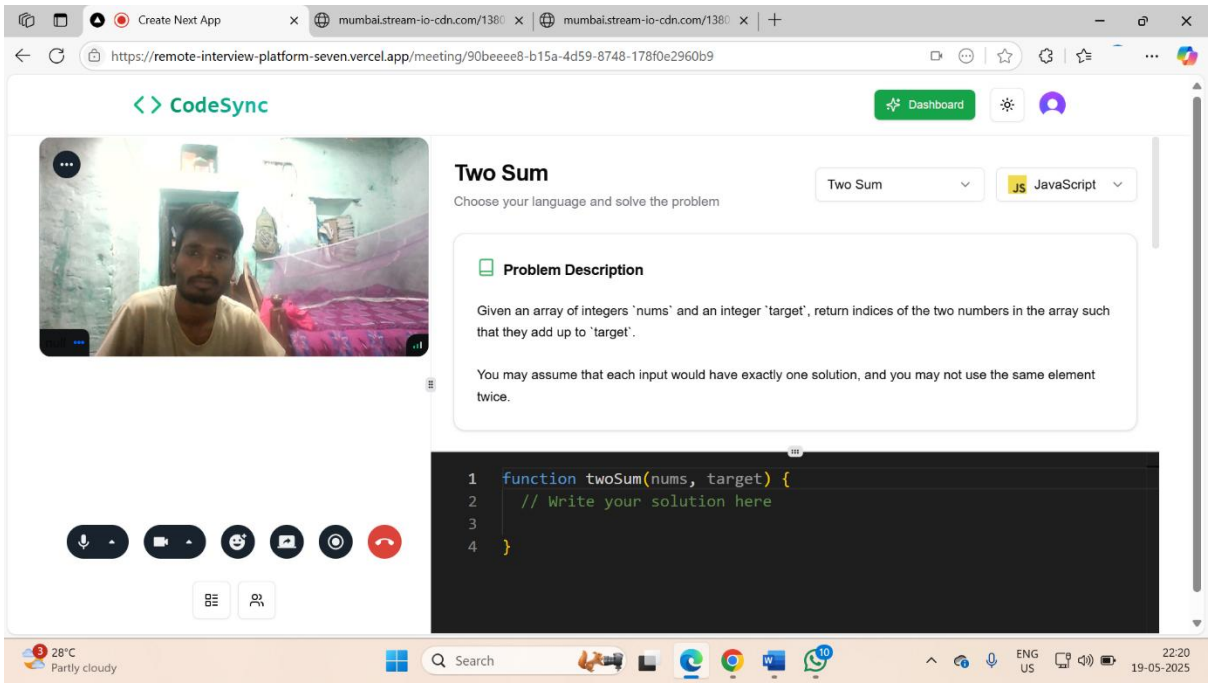

Fig.56. Main Panel of Interview app

All the function include here like coding coding (language select option) question solution and screen share of the computer close the meeting and give rection and see other member to join the meeting etc.

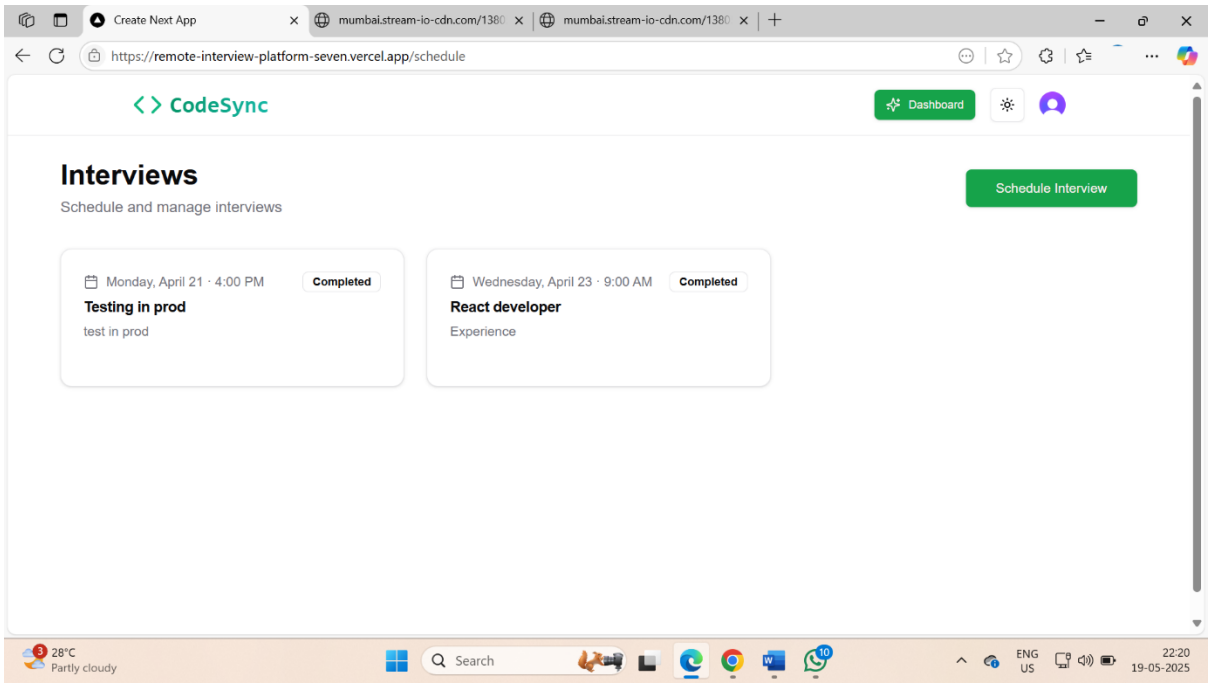

Fig.57. Interview History

Completing interview history stored here (date, time, subject, name etc.)

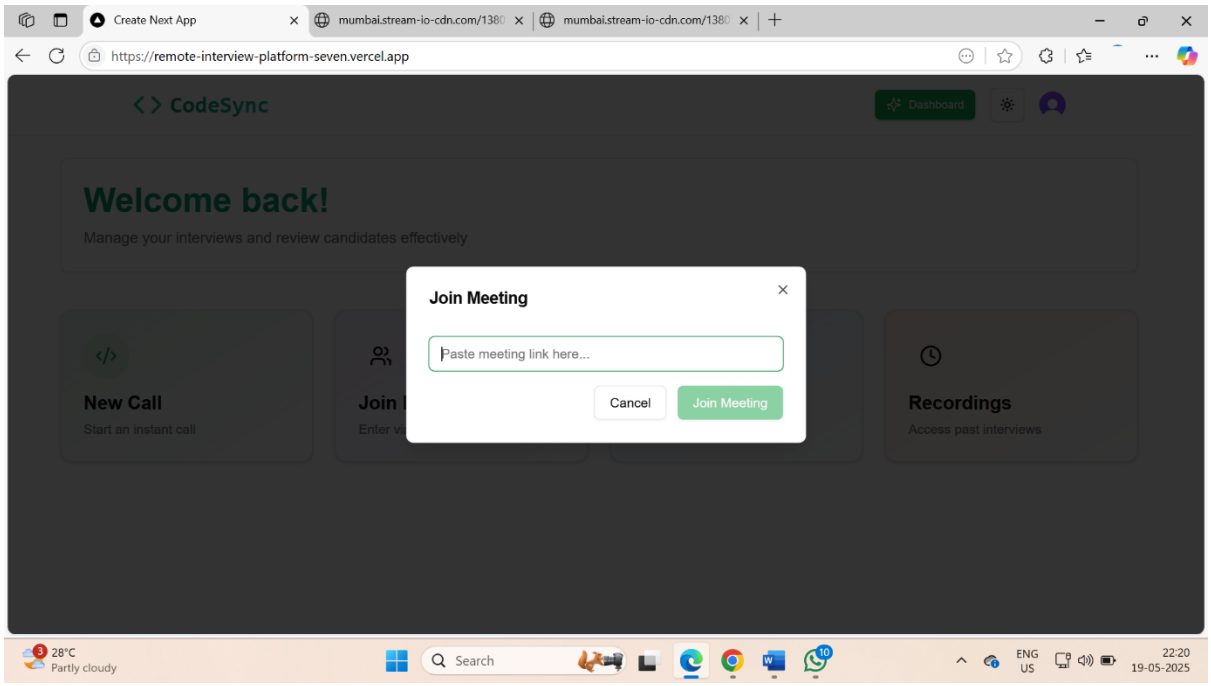

Fig.58. Jon new Metting through a meeting link

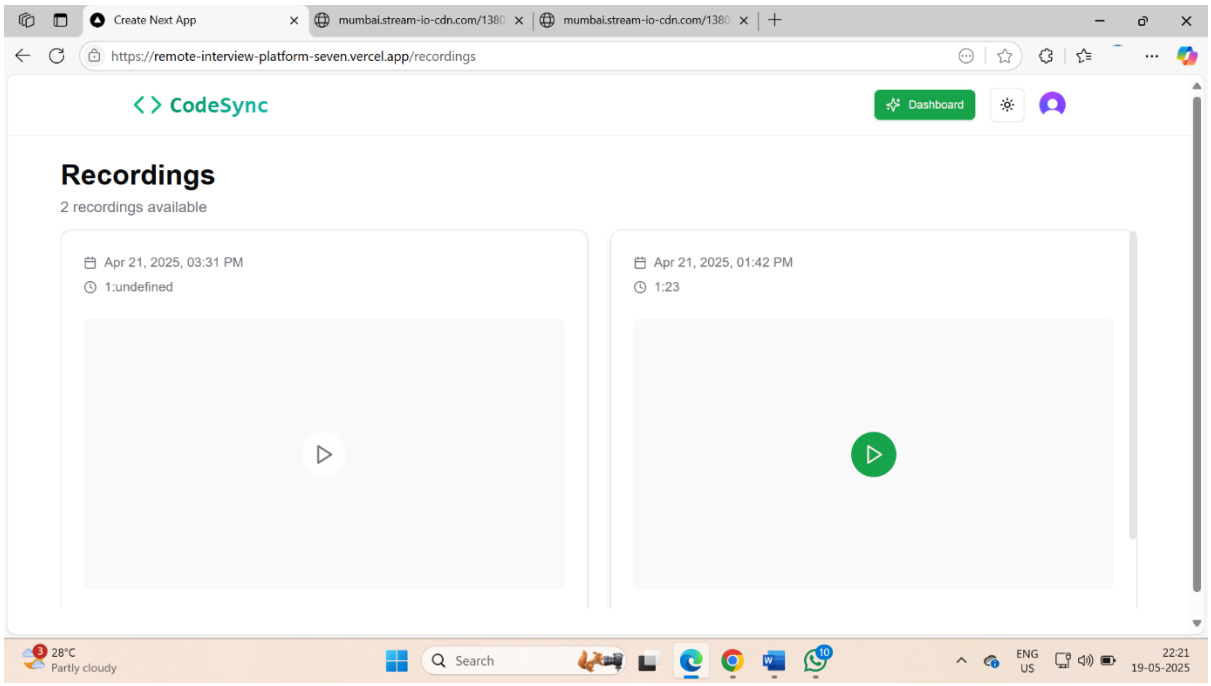

Fig.59. Complete recording save

When a user completes an interview with a recruiter, the **interview video is recorded and securely stored** in their profile. This allows users to **rewatch the interview at any time**, helping them reflect on their performance and identify areas for improvemen

9. Discussion

Developing this project took a great deal of time, effort, and continuous learning. I started by watching tutorials on YouTube, gradually building my understanding and adding more ideas and techniques. With the constant guidance and support of my project mentor, **Sakshi Mam**, I was able to take this project to the next level. She not only suggested valuable improvements but also encouraged me to contribute my own ideas.

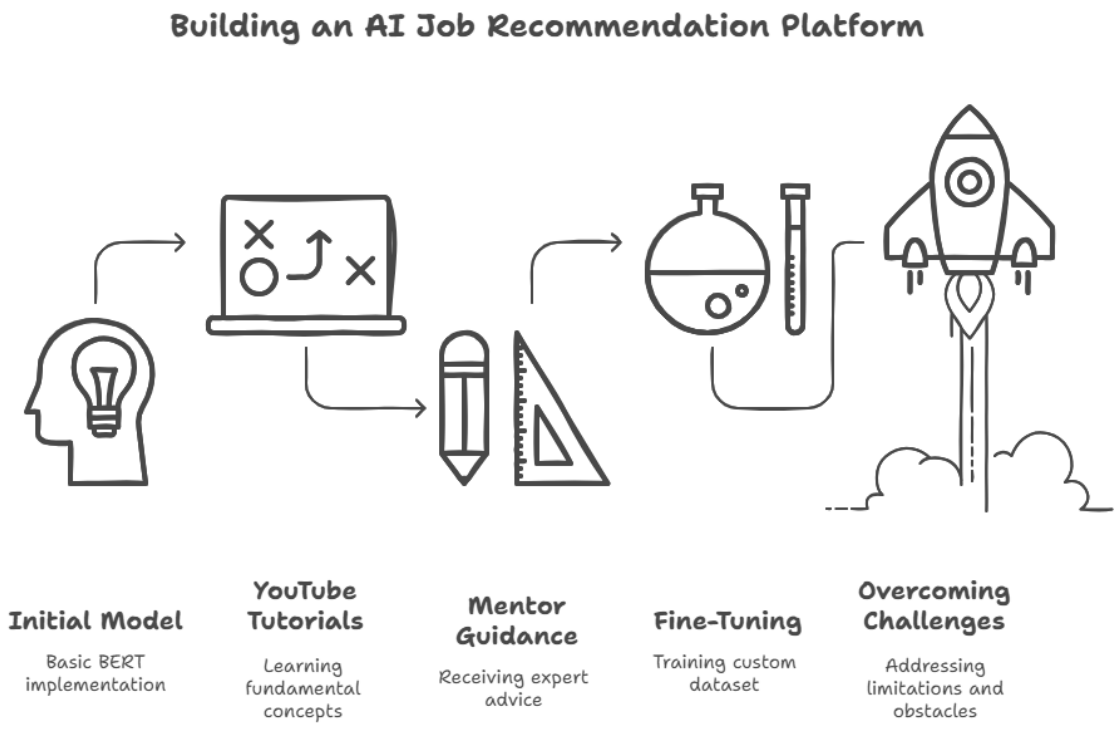

Fig.60. Resource Follow and Guidance

Initially, I simply passed data into a BERT model to get job recommendations. It was my first time working directly with a BERT model, and while the results were promising, my mentor advised me to go further by fine-tuning the model on our custom dataset. That guidance helped me train a model specifically for our use case, which significantly improved the accuracy of job recommendations.

Through this process, I learned a lot not just about using pre-trained models, but also about fine-tuning them, preparing datasets, evaluating performance, and building practical systems.

However, I also faced many challenges along the way. One major **challenge** was learning new concepts in deep learning, especially around **Transformers**, **BERT**, and **GPT** models. These are complex topics, and it took time and patience to understand how they work.

Another challenge was training the model itself. I used **Google Colab** for most of the training, but the free version has limitations. Sometimes the GPU usage exceeded my daily quota, which forced me to wait or upgrade to a premium account. Even with a **Core i7 processor and a 4GB NVIDIA graphics card** on my personal system, training locally caused overheating and performance issues. This made me realize how important high-performance infrastructure (like powerful GPUs and cloud resources) is for training and deploying AI models efficiently.

I now understand why companies invest so much in infrastructure for AI projects like ChatGPT. The computational requirements are high not just for training but also for running these models at scale.

Despite the difficulties, this project taught me a lot and has prepared me to work on even bigger challenges in the future. As the AI and machine learning fields continue to grow, I believe more accessible tools and resources will make it easier for individuals to train and deploy their own models.

Throughout the development of our AI-powered job recommendation platform, several components worked well, while others presented key challenges that shaped our learning and future goals.

After developing the BERT-based job recommendation system, I wanted to create a more complete AI recruitment platform by integrating additional intelligent tools that would support users across different stages of the hiring process. This led to the development of two more key components: the **AI Resume Builder** and the **Meeting App**.

The goal of the Resume Builder was to simplify the resume creation process using AI assistance. Many job seekers struggle with formatting, structuring, or wording their resumes. One standout feature was the use of **Google Gemini API** (or another large language model) to assist users in generating optimized professional summaries, skills, and experience descriptions. This made resume writing more personalized and efficient. The **Meeting Application** is beneficial for both job seekers and recruiters, making it a valuable feature of our platform. When a user applies for a job and gets shortlisted, the recruiter can share an **interview meeting link** directly through the platform.

10.Conclusion & Future Work

we developed a comprehensive AI-powered recruitment platform that integrates multiple intelligent modules to assist job seekers more effectively. The system includes a BERT-based job recommendation engine, an AI-driven resume builder, a GPT-style interview assistant, and an interactive interview simulator. All components try to solve key challenges faced by candidates in the job search process, such as irrelevant recommendations, weak resume quality, and lack of interview preparation. Our BERT-based recommendation module significantly improves the relevance of job matches by understanding the semantic meaning behind user profiles and job descriptions.

The resume builder provide to the good templates to user can make easily on your resume with guidance of AI (use Gemini API) key for suggestion according to the user input, while the interview assistant uses a GPT-based model help to better preparation of the interview exam. Overall, the platform is good for visit of new user to better environment provide by our platform because advanced natural language processing and machine learning techniques can be combined to build an end-to-end smart recruitment assistant that is both scalable and user-friendly.

Future Work

while our platform already offers a strong foundation but need some future work to more user friendly and more perfect job matching.

- I Currently, I use the **all-MiniLM-L6-v2** model (lightweight), but in the future, I plan to explore more powerful BERT variants to train the dataset and improve model accuracy. Potential candidates include distilbert-base-uncased, RoBERTa (Robustly Optimized BERT), DeBERTa (Decoding-Enhanced BERT with Disentangled Attention)
- **Smarter Personalization**
  We want the system to learn more from each user over time. By using reinforcement learning, it could adjust job recommendations based on what users click, save, or apply to making results even more relevant.
- **Features for Employers**
  A big next step is to include tools for employers as well. This would let companies post jobs, review candidates, and get AI-driven recommendations—all in one place.
- **Better Connections Between Skills and Jobs**
  We're also exploring the use of knowledge graphs or graph-based models to understand how skills, roles, and industries connect. This could lead to more accurate and insightful recommendations.

11. References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv preprint arXiv:1810.04805, 2018. [Online]. Available: https://arxiv.org/abs/1810.04805

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in Proc. of NAACL-HLT, pp. 4171–4186, 2019.

[3] A. Vaswani et al., “Attention Is All You Need,” in Advances in Neural Information Processing Systems, vol. 30, 2017.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, et al., “Language Models are Few-Shot Learners,” in Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[5] Google, “Introducing Gemini: Google’s next-generation AI,” 2023. [Online]. Available: https://blog.google/technology/ai/google-gemini-ai/

[6] Google Research, “all-MiniLM-L6-v2 Model Card,” 2023. [Online]. Available: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[7] Hugging Face, "Fake Job Posting Prediction Dataset (victor/real-or-fake-fake-jobposting-prediction)," Hugging Face Datasets, 2023. [Online]. Available: https://huggingface.co/datasets/victor/real-or-fake-fake-jobposting-prediction

[8] Shamiraty, "Django Job Recommendation System," GitHub, 2022. [Online]. Available: https://github.com/shamiraty/django-job-recommendation-system

[9] Euron, “Rebuilding ChatGPT with Python,” YouTube, Apr. 13, 2025. [Online]. Available: https://www.youtube.com/watch?v=WnFOQJYxRXI

[10] M. M. Hossain, M. H. Asaduzzaman, M. R. R. Rohan, and M. S. Hossain, "BERT-based Job Recommendation System Using LinkedIn Dataset," Journal of Information Systems Engineering and Management (JISEM), vol. 8, no. 3, 2025. [Online]. Available: https://jisemjournal.com/index.php/journal/article/view/1026