RAJALAKSHMI ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

# JOBFIT PRO

Submitted by:

**HARSAVARDHINI R (221801016)**
**KAVIYA S (221801024)**
**MONISHA M (221801034)**

**AD19541 SOFTWARE ENGINEERING METHODOLOGY**

**Department of Artificial Intelligence and Data Science**

**Rajalakshmi Engineering College, Thandalam**

## BONAFIDE CERTIFICATE

Certified that this project report "**JOBFIT PRO**" is the bonafide work of **"HARSAVARDHINI R (221801016), KAVIYA S (221801024), MONISHA M (221801034)"** who carried out the project work under my supervision.

**Submitted for the Practical Examination held on ________________________**

| | |
|---|---|
| **SIGNATURE** | **SIGNATURE** |
| **Dr.J.M.GNANASEKAR**<br>**Professor & Head,**<br>**Dept. of Artificial Intelligence and Data Science,**<br>**Rajalakshmi Engineering College,**<br>**Thandalam, Chennai-602105** | **Dr.MANORANJINI J**<br>**Associate Professor,**<br>**Dept. of Artificial Intelligence and Data Science,**<br>**Rajalakshmi Engineering College ,**<br>**Thandalam, Chennai-602105** |

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

## ABSTRACT

In today's competitive job market, efficiently identifying and shortlisting the most qualified candidates for open positions is a critical yet time-consuming task for recruiters. This project aims to streamline the recruitment process by developing a resume filtering system using Meta's Llama 3.1 8B Instruct model, an advanced natural language processing (NLP) model optimized for understanding and evaluating textual information. The system is implemented through a user-friendly web platform that facilitates seamless interaction between job seekers and hiring administrators. Job seekers can log into the platform, search for suitable job roles, and submit applications by filling in personal details and uploading their resumes. Each job listing has an application deadline, after which the hiring administrator can review and process applications. Once candidates have submitted their applications the Llama 3.1 model analyzes the content of each resume in relation to the job requirements. The model's assessment considers factors such as experience, education, skill relevance, and role-specific keywords, generating a ranking based on candidate suitability for each specific position. This automated screening generates a top-10 list of the most qualified applicants, enabling recruiters to focus their attention on a refined list of high-potential candidates.By leveraging Llama 3.1's capabilities, this project introduces an AI-driven approach that not only reduces the time and effort required in the resume screening process but also mitigates human biases, ensuring a fair and objective evaluation. Furthermore, the system's scalability allows it to handle large volumes of applications, making it suitable for various industries and roles. The project ultimately delivers a reliable, accurate, and efficient solution that enhances the hiring experience for both applicants and recruiters, empowering organizations to make data-driven hiring decisions while offering job seekers a transparent and equitable application process.

# TABLE OF CONTENTS

## LIST OF FIGURES

# CHAPTER I
# INTRODUCTION

## 1.1 GENERAL

The use of AI in recruitment is transforming traditional hiring methods by streamlining the process of candidate selection and reducing time-to-hire. With advancements in natural language processing (NLP) and machine learning, AI-driven models, such as Meta's Llama 3.1, are designed to efficiently handle large volumes of candidate data, identify the best matches for job roles, and present recruiters with an optimized pool of applicants.

AI-powered resume filtering enables companies to quickly and accurately evaluate a candidate's qualifications based on job-specific criteria. This technology helps mitigate human bias, allowing for fairer candidate assessments that prioritize skills and experience over subjective impressions. Additionally, these systems are scalable and adaptable, making them valuable for companies of all sizes and across industries.

## 1.2 NEED FOR THE STUDY

The recruitment process is a cornerstone for organizational growth and success, yet traditional hiring methods are often time-consuming, costly, and prone to biases. With the increasing volume of applications for each position, especially in highly competitive fields, recruiters face challenges in efficiently filtering through resumes to identify top candidates. Manual resume screening can lead to inconsistent evaluations, unconscious biases, and errors, ultimately impacting the quality of hires and the fairness of the recruitment process.

The need for this study arises from the demand for scalable, unbiased, and efficient solutions to streamline resume filtering. Leveraging AI-driven tools, like Meta's Llama 3.1 model, offers the potential to enhance the hiring process by automating and standardizing initial candidate evaluations based on data-driven insights. This study aims to address critical recruitment issues by implementing AI to improve the speed, accuracy, and objectivity of resume screening, providing both recruiters and applicants with a fairer, more transparent process.

Furthermore, this study contributes to the evolving field of AI in human resources, exploring how advanced NLP models can be applied effectively in real-world hiring scenarios. By investigating the application of AI for resume filtering, this project seeks to provide a framework that organizations can adopt to optimize hiring, reduce time-to-hire, and improve overall candidate quality, ultimately driving better organizational outcomes.

## 1.3 OVERVIEW OF THE PROJECT

This project is designed to create an AI-powered resume filtering system that leverages Meta's Llama 3.1 8B Instruct model to streamline the recruitment process for both applicants and hiring administrators. Built as a web-based platform, the system provides a seamless experience for job seekers to log in, search for suitable job openings, and apply by submitting personal details and uploading their resumes. Each job listing has an associated deadline, ensuring that applications are submitted in a timely manner for the hiring process.

Once candidates apply for a role, their resumes are evaluated by the Llama 3.1 model, which is trained to interpret and assess text data. This AI model analyzes each resume in-depth, assessing candidate qualifications based on experience, education, skill relevance, and alignment with specific job requirements. Using these criteria, the model ranks the applicants and generates a top-10 list of the most suitable candidates for each role.

For hiring administrators, this project offers a user-friendly dashboard that provides a prioritized list of applicants, helping them to quickly identify the top candidates for in-depth review. This automated shortlisting enables recruiters to bypass the manual filtering process, saving time and resources while ensuring a fair, objective evaluation of each applicant.

The project not only enhances the efficiency of the hiring process but also minimizes human biases and standardizes candidate evaluations, making the system fairer and more transparent. By implementing this AI-driven solution, the project aims to redefine traditional recruitment by enabling data-driven, scalable hiring processes that benefit both organizations and job seekers.

## 1.4 OBJECTIVES OF THE STUDY

The primary objective of this project is to develop an AI-powered resume filtering system that automates the initial stages of candidate selection, significantly reducing the time and manual effort required for recruitment. By leveraging Meta's Llama 3.1 model, the system aims to enhance the accuracy of candidate evaluation, ranking applicants based on qualifications, experience, and job-specific skills to ensure that the most suitable candidates are identified for each role. An essential goal is to promote fairness by implementing standardized, data-driven algorithms that minimize human biases, offering an objective assessment of all applicants.

Furthermore, the project seeks to increase efficiency by providing hiring administrators with a streamlined list of the top candidates for each role, thereby reducing time-to-hire and allowing recruiters to focus on in-depth evaluations. Another critical objective is to create a user-friendly web platform that enables seamless interaction for both job seekers and hiring administrators, facilitating easy navigation, application submission, and review processes. Scalability and flexibility are also prioritized, allowing the solution to adapt to various industries, company sizes, and applicant volumes, ensuring it meets the diverse needs of the recruitment landscape.

By achieving these objectives, the project not only aims to optimize the recruitment process for organizations but also enhances the experience for job seekers, providing a transparent, fair, and efficient application process that equitably evaluates candidates based on their skills and qualifications.

# CHAPTER 2
# REVIEW OF LITERATURE

## 2.1 LITERATURE SURVEY

The application of AI and Natural Language Processing (NLP) in resume screening has been extensively explored in previous studies. Joulin et al. (2017) proposed deep learning models, particularly using a bag-of-words approach, to rank resumes based on job descriptions. However, their method lacked the ability to understand the broader context and nuanced language found in resumes. Similarly, Mohammad et al. (2015) focused on traditional keyword-based filtering systems used in Applicant Tracking Systems (ATS). While effective for matching specific keywords, this approach often overlooked resumes with synonyms or varied phrasing, failing to assess resumes holistically.

Zhao et al. (2018) introduced machine learning-based methods, specifically decision trees and random forests, to rank resumes. While this approach automated resume ranking, it was still limited by its reliance on predefined criteria and lacked adaptability to different job descriptions and resume formats. In contrast, Chen et al. (2020) explored deep learning models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to classify and rank resumes based on their content. These models were successful in improving semantic understanding but required large datasets for training and were computationally expensive.

Another critical concern in AI-based recruitment systems is bias. Binns et al. (2021) examined how AI models could unintentionally inherit biases from training data, leading to discrimination in recruitment decisions. This highlights the need for ethical considerations in AI system design. Buolamwini and Gebru (2018) proposed methods for debiasing AI systems, including data augmentation and fairness-aware learning, to mitigate these biases. However, their research primarily focused on visual recognition, though it provides insights into mitigating bias in AI recruitment systems.

While AI models provide benefits in terms of speed and consistency, Huang et al. (2021) demonstrated that human recruiters still outperform AI when it comes to evaluating soft skills or assessing candidates for roles that require creativity. Despite this, AI systems can be more consistent and faster, especially when analyzing large volumes of resumes. Finally, Li et al. (2022) proposed a hybrid approach that combines traditional ATS systems with advanced deep learning models, suggesting that this fusion of methods could lead to more accurate and nuanced evaluations of candidates. However, such hybrid systems require more resources and are more complex to implement.

Overall, the literature reveals that while traditional keyword-based filtering and machine learning approaches have improved resume screening, there remains a significant gap in fully understanding the context and potential of candidates. Additionally, issues related to bias, fairness, and the complexity of deep learning models must be carefully managed for these systems to be truly effective and ethical in recruitment processes.

# CHAPTER 3
# SYSTEM OVERVIEW

## 3.1 EXISTING SYSTEM

The existing system for customer segmentation and marketing in retail primarily relies on traditional methods such as the RFM (Recency, Frequency, Monetary) model, which segments customers based on their purchasing history. This model helps categorize customers into different groups by considering how recently they purchased, how often they make purchases, and the monetary value of their transactions. Techniques like K-means clustering is used alongside RFM to further classify customers into segments, allowing for basic targeted marketing.

Moreover, techniques such as Association Rule Mining (e.g., Apriori or FP-Growth) are employed to identify relationships between products frequently bought together, which can be used for cross-selling or bundling promotions. Additionally, Naive Bayes and other basic classification algorithms are applied to predict the likelihood of customers responding to promotions based on their historical purchasing patterns or demographic information.

However, these traditional approaches have several limitations. The RFM model is constrained by its use of only three metrics, potentially overlooking other important aspects of customer behavior and preferences. As a result, customer segments may not be nuanced enough to reflect the diversity of individual preferences. Traditional methods also create static customer segments that may not adapt to changing customer behaviors, leading to outdated marketing strategies. Furthermore, Association Rule Mining techniques can struggle with scalability and memory issues when handling large datasets, reducing their effectiveness in big data scenarios. These limitations underscore the need for more sophisticated, data-driven approaches that can offer dynamic and personalized marketing solutions.

## 3.2 PROPOSED SYSTEM

The proposed system focuses on using Meta’s Llama 3.1 8B model for automated resume filtering. This system leverages the advanced capabilities of Llama 3.1 to analyze and evaluate resumes based on specific job requirements, providing an accurate ranking of candidates. The process begins when a candidate submits their resume through a job portal. The resume is parsed, and the relevant data, such as skills, experience, education, and certifications, are extracted. Meta Llama 3.1 is then used to assess how well the candidate's qualifications match the job description by scoring the resumes according to predefined criteria.

The AI-driven model processes the resumes, using its advanced NLP capabilities to evaluate the context and meaning behind the content, rather than just relying on keywords. This ensures that even resumes with varied formats or phrasing are accurately assessed. The system ranks the candidates based on their alignment with the job requirements, producing a filtered list of the most suitable applicants. The top-ranked candidates are then presented to the hiring administrators for review, allowing them to focus on the best candidates for further selection. The system ensures accuracy and fairness in candidate evaluation by relying on objective data rather than subjective judgment, reducing bias and improving efficiency in the recruitment process.

## 3.3 FEASIBILITY STUDY

The proposed AI-powered resume filtering system, utilizing Meta Llama 3.1 through Ollama, run locally on a laptop with Visual Studio Code, is technically, operationally, and financially feasible for the development phase. The use of Meta Llama 3.1 enables the application of state-of-the-art Natural Language Processing (NLP) to analyze and rank resumes based on job-specific criteria, which significantly enhances the efficiency of the recruitment process. Running Ollama locally provides flexibility and control over the development environment, allowing for easy testing and iterative improvements without incurring high initial infrastructure costs. This setup is ideal for the development phase as it offers a manageable scale, but as the system matures and the volume of resumes grows, the feasibility study suggests considering cloud-based resources to accommodate the increased demand for data processing. Overall, the combination of cutting-edge AI and a locally controlled environment makes this an effective and practical approach for the system's initial implementation.

**Technical Feasibility**: The system is technically sound, as Meta Llama 3.1 offers robust Natural Language Processing (NLP) capabilities for accurately parsing and analyzing resumes. Running the model locally using Ollama in Visual Studio Code provides a flexible and efficient environment for development and testing. However, since Llama 3.1 is a large model, performance optimization may be needed when processing large volumes of resumes. As the project progresses, considerations for scaling to cloud infrastructure may be required to handle increased data processing demands.

**Operational Feasibility**: The system's design effectively automates the resume filtering process, streamlining the recruitment workflow. By using AI to analyze and rank resumes based on job requirements, the system minimizes human bias and improves the consistency of candidate evaluations. This approach enhances the efficiency of hiring managers and reduces the time spent manually screening resumes. During the development phase, the local setup is sufficient, but as the user base grows or larger datasets are handled, scaling the system through cloud resources would ensure smooth operation.

**Financial Feasibility**: Running the Llama 3.1 model locally on your laptop significantly reduces development and operational costs in the initial phases, eliminating the need for expensive cloud-based infrastructure. The system’s operational costs are low, with resources being used on-demand for processing resumes. As the system scales and processes a higher volume of resumes, moving to a cloud-based infrastructure might be necessary. Furthermore, the system could be monetized through a subscription-based model or licensing, providing a potential revenue stream for future sustainability.

**Legal and Ethical Feasibility**: Legal and ethical considerations are critical for ensuring that the system complies with data privacy laws such as GDPR and CCPA. The candidate data must be securely stored, processed, and anonymized, with clear consent obtained before use. The AI model must also be regularly audited to ensure it does not perpetuate discrimination or biases based on protected characteristics. Ethical practices must be adhered to in order to maintain fairness and transparency in the evaluation process.

# CHAPTER 4
# SYSTEM REQUIREMENTS

## 4.1 HARDWARE REQUIREMENTS

**Local Machine Requirements (For Running Meta Llama 3.1 Locally)**:

**Processor (CPU)**:

- A modern multi-core processor capable of handling large-scale computations.
- Recommended: Intel Core i7, AMD Ryzen 7 (or higher) for optimal performance.

**Graphics Processing Unit (GPU)**:

- A dedicated GPU for faster training and inference, as Meta Llama 3.1 involves NLP tasks that benefit from parallel processing.
- Recommended: NVIDIA GTX 1660, RTX 2060, or higher for local processing.
- **Optional**: For better performance, NVIDIA RTX 3000 series or equivalent GPUs can significantly reduce the processing time.

**Memory (RAM)**:

- Adequate RAM is essential for handling large datasets during training and inference.
- Recommended: Minimum 16 GB of RAM; 32 GB or more is preferred for large-scale tasks.

**Storage**:

- An SSD (Solid-State Drive) ensures faster read/write operations, especially for large models and datasets.

- Recommended: At least 500 GB SSD for the operating system and software; 1 TB or more for model and data storage.

**Network Connection**:

- A fast internet connection for downloading model dependencies, data processing, and uploading resumes.
- Recommended: Broadband with at least 10 Mbps download/upload speeds.

**Cloud Infrastructure Requirements (If Using Cloud Resources)**:

**Cloud Provider**:

- Cloud services such as AWS, Google Cloud, or Microsoft Azure can be used for scalable resources.
- Cloud providers offering GPU instances are recommended for efficient processing.

**Compute Instances**:

- Powerful compute instances with multi-core CPUs and GPUs for training and inference.
- Recommended: Instances with NVIDIA Tesla V100 or A100 GPUs for training and large-scale inference.

**Storage**:

- Cloud storage like AWS S3, Google Cloud Storage, or Azure Blob Storage to handle large datasets and model storage.
- Recommended: Scalable cloud storage with at least 1 TB capacity for resumes and data.

**Network Bandwidth**:

- High-speed network connection to handle data transfer, especially if large resume files are processed frequently.
- Recommended: At least 1 Gbps for cloud instances.

## 4.2 SOFTWARE REQUIREMENTS

**Operating System:**

- **Supported OS**: Windows, Linux (Ubuntu preferred for machine learning setups), or macOS.
- **Recommendation**: Ubuntu 20.04 or later, as it offers extensive support for machine learning libraries and Docker.

**Programming Languages:**

- **Primary Language**: Python (version 3.8 or later) for model training, data processing, and backend development.
- **Other Languages**: HTML, CSS, and JavaScript (for frontend development).

**AI and Machine Learning Libraries:**

- **PyTorch**: Essential for running the Meta Llama 3.1 model, as it is optimized for NLP tasks.
- **Transformers**: Hugging Face's library, needed to manage and fine-tune large language models like Meta Llama.
- **Natural Language Toolkit (NLTK)** and **spaCy**: For text preprocessing, such as tokenization and stop-word removal.

**Web Development Frameworks:**

- **Backend Framework**: Flask or Django to build the backend server, handle user requests, and process data.

- **Frontend Framework**: HTML/CSS for static content, with JavaScript-based libraries like React.js or Vue.js for a dynamic, responsive interface.

**Database:**

- **Database Management Systems**: MySQL, PostgreSQL, or MongoDB to store user data, job listings, and application records.

**Model Deployment and Management:**

- **Ollama**: Required to run Meta Llama 3.1 locally, enabling integration with Visual Studio Code for easy model management.
- **CUDA Toolkit**: For leveraging GPU support with NVIDIA cards, improving processing speed during training and inference (if using GPU).
- **Docker** (optional): For containerizing the application, ensuring a consistent setup across different development environments.

**Development and Collaboration Tools:**

- **Integrated Development Environment (IDE)**: Visual Studio Code, PyCharm, or Jupyter Notebook for writing and testing code.
- **Version Control**: Git for tracking changes, enabling collaboration, and managing code versions.
- **Postman** (optional): For testing backend APIs and ensuring smooth communication between the frontend and backend.

**Optional Cloud Services (For Scaling):**

- **Cloud Platforms**: AWS, Google Cloud, or Microsoft Azure for running scalable instances, if additional computing power or storage is needed.
- **Compute Instances**: GPU instances (e.g., AWS EC2 with Tesla V100) for faster inference.
- **Cloud Storage**: Services like AWS S3 or Google Cloud Storage for handling and backing up large datasets and user-uploaded resumes.

# CHAPTER 5
# SYSTEM DESIGN

## 5.1 SYSTEM ARCHITECTURE

The system architecture for the AI-driven resume filtering system leveraging Meta Llama 3.1 is designed with a modular and layered approach, ensuring optimized processing of user data, seamless application of machine learning algorithms, and effective presentation of results. This architecture is built to support scalability, flexibility, and maintainability, crucial for handling complex resume analysis and matching tasks. Below is an in-depth overview of each core component and its role in the Architecture.

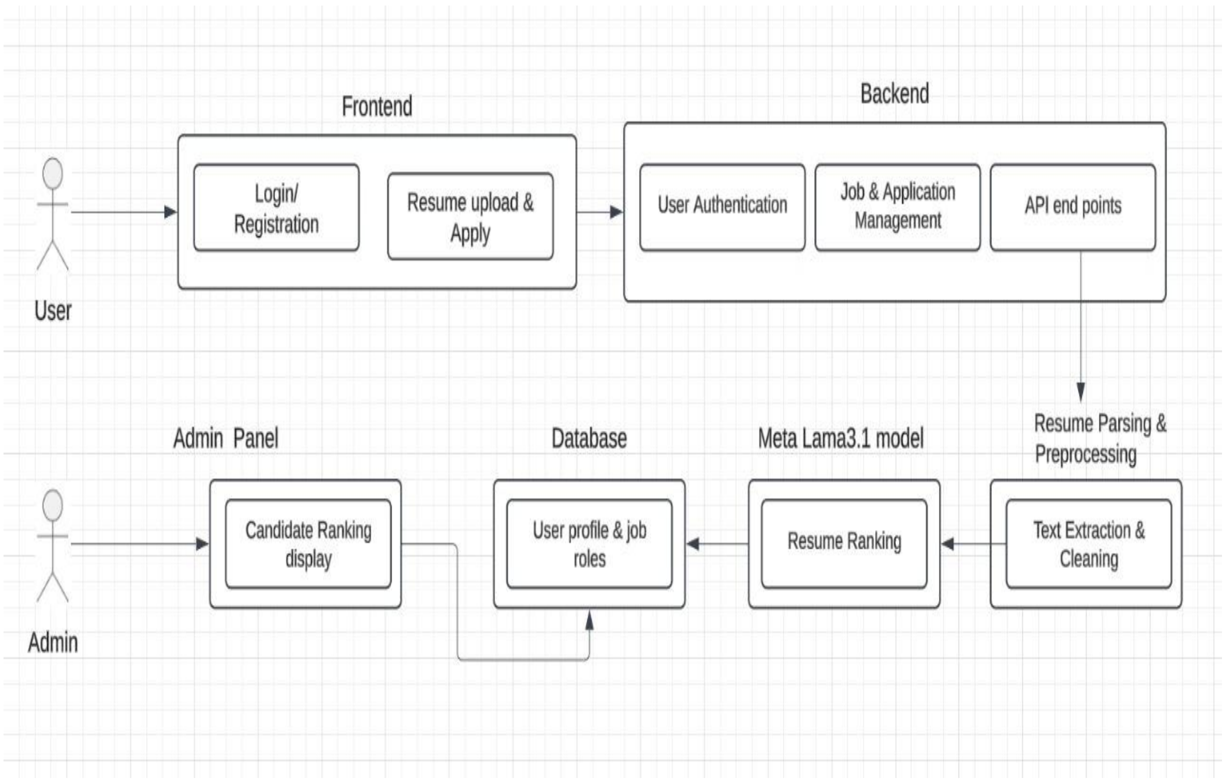


Figure 1: System Architecture

## 5.2 MODULE DESCRIPTION

### 5.2.1 USER INTERFACE (FRONTEND)

- **Role**: Provides an intuitive platform for candidates to interact with the system. Users can log in, search for relevant job roles, upload resumes, and submit their applications.
- **Technologies**: Built using HTML, CSS, and JavaScript to create a responsive and user-friendly interface. The UI allows for easy navigation and interaction.
- **Features**: Includes job search, resume upload, application submission, and feedback display.

### 5.2.2 BACKEND (DJANGO FRAMEWORK)

- **Role**: Manages all interactions between the frontend, database, and Meta Llama 3.1 model. The backend is responsible for user authentication, job data management, resume processing, and handling requests from the frontend.
- **Technologies**: Built with Django, a Python-based framework that facilitates secure, scalable, and maintainable web applications.
- **Components**:
    - **User Authentication & Authorization**: Handles login and security protocols.
    - **Job & Application Management**: Manages job postings, applications, and deadline checks.
    - **API**: Provides endpoints for frontend requests, including resume submission and application status updates.
    - **Resume Processing**: Extracts and preprocesses resume data to prepare it for AI model analysis.

### 5.2.3 RESUME PARSING & PREPROCESSING

- **Role**: Extracts and cleans data from uploaded resumes to prepare them for AI processing.
- **Technologies**: Libraries like PyPDF2 and nltk for text extraction and natural language processing. The system handles tokenization, lemmatization, and other preprocessing steps.
- **Process**: Text data is extracted, standardized, and formatted to ensure compatibility with Meta Llama 3.1.

### 5.2.4. META LLAMA 3.1 MODEL (AI BASED RESUME FILTERING)

- **Role**: The core component that ranks resumes based on relevance to job descriptions. It evaluates candidates by matching their qualifications, skills, and experiences to job requirements.
- **Technologies**: PyTorch and Hugging Face Transformers are used to implement and run the Meta Llama 3.1 model.
- **Process**: After resumes are preprocessed, they are fed into the model, which outputs a ranking score. Resumes are then sorted based on their scores, identifying the top candidates.

### 5.2.5 DATABASE

- **Role**: Stores user profiles, job roles, applications, and resume data.
- **Technologies**: Relational databases like MySQL or PostgreSQL, or a NoSQL database like MongoDB.
- **Data Management**: Manages structured data for candidate profiles, job details, and ranked resumes, enabling efficient retrieval during admin review.

### 5.2.6 Admin Panel

- **Role**: Provides the administrator (hirer) with an interface to view, evaluate, and manage the ranked resumes.

- **Features**: Displays the top 10 ranked candidates for each role, along with resume details, application status, and options to shortlist or reject candidates.
- **Integration with Django**: Built as part of the Django backend, ensuring secure access and smooth data handling.

# CHAPTER 6
# UML DIAGRAMS

## 6.1 DATA FLOW DIAGRAM

**Data Flow Diagram (DFD) Description for Job Fit Pro**

The Data Flow Diagram (DFD) for the Job Fit Pro system illustrates how data flows between the various modules and components, emphasizing the interaction between users and the system. Below is a detailed description of each level and the flow of data

**Level 0 (Context Diagram):**

The Context Diagram provides a high-level view of the system, showing the interaction between the external actors (Job Applicant, Job Poster, and Admin) and the Job Fit Pro system. Data flows in and out of the system as follows:

- **Job Applicant** interacts with the system to submit personal details, upload resumes, and apply for job roles.
- **Job Poster** provides job requirements and manages postings.
- **Admin** oversees the process, manages candidate notifications, and reviews ranked resumes.

**Level 1 (High-Level Processes):**

**1. User Interaction Module**

- **Collect User Information**: Users interact with the system by registering, logging in, and uploading their resumes.
- **User Registration**: Candidates register to create their profiles, storing relevant details in the **User Profile Database**.
- **User Login**: Registered users log in to access the platform, allowing them to view job postings and submit applications.
- **Resume Upload**: Users upload their resumes, which are stored in the **Resume Database** for further processing and filtering

**2. Company/HR Module**

- **Manage Job Postings**: HR representatives or administrators manage job postings and job roles to specify the requirements for each role.
- **Edit Job Requirements**: HR can edit job requirements as needed, ensuring that the job criteria in the **Job Requirements Database** are up to date.
- **Job Role Creation**: New job roles are created and stored in the **Job Requirements Database**, which the filtering system later references for matching candidates' resumes.

**3. Resume Filtering and Ranking Module**

- **Filter and Rank Resumes**: This core module is responsible for evaluating resumes based on job criteria and ranking them.
- **Rank Candidates**: Using AI-powered algorithms, candidates are ranked based on how well their resumes align with job requirements.
- **Apply Filtering Criteria**: Specific filtering criteria are applied to narrow down the list of candidates to those who meet the minimum qualifications for the role.
- **Data Storage**: The **Intermediate Filtered Resumes** and the **Ranked Candidates Database** store filtered and ranked resumes for easy access during the final review.

**4. Admin Module**

- **Notify Candidates**: Administrators (hirers) can send notifications to candidates, updating them on the status of their applications.
- **Email Log**: Notifications sent to candidates are recorded in an **Email Log** for administrative tracking and communication history.
- **Fetch Top N Candidates**: The admin can retrieve the top candidates based on their ranking scores for each job role, making it easier to review the most suitable resumes.

- **Send Notifications**: The system sends notifications to the top-ranked candidates, keeping them informed of the progress and next steps in the hiring process.

## 6.2 USE CASE DIAGRAM

### Use Case Diagram Description for Job Fit Pro

The use case diagram for the Job Fit Pro system provides a visual representation of the interactions between different user roles (actors) and the system's functionalities. The diagram outlines the key use cases that facilitate efficient job application management, highlighting the relationships and flow of actions within the system.

### 1.Actors Involved

1. **Job Applicant**: A candidate interested in searching for jobs and applying for available roles.
2. **Job Poster**: An individual responsible for creating and posting job roles within the system.
3. **Admin**: The administrator who manages the selection process and oversees candidate notifications.

### 2. Use Cases

#### 1.Login:

- All users (Job Applicants, Job Posters, and Admins) need to log into the system to access its features. Logging in is a primary entry point that authenticates users and grants them specific permissions based on their roles.

**2.View Job Roles**:

- The Job Applicant can view available job roles within the system. This use case allows applicants to browse job postings, view job descriptions, and decide which roles to apply for.

**3.Posts Job Roles**:

- Job Posters are responsible for posting new job roles. This includes specifying job descriptions, qualifications, and other requirements, which are made available for job applicants to view.

**4.Apply Job Roles**:

- Job Applicants can apply for job roles that match their skills and interests. This use case includes filling in personal details, uploading resumes, and submitting applications for specific roles.

**5.Filter Resumes Using LLM Model**:

- After candidates apply for job roles, the system automatically filters and ranks resumes using an integrated machine learning model. This filtering helps streamline the selection process by identifying the most relevant candidates based on predefined criteria.

**6.Select Candidates**:

- The Admin reviews the list of candidates filtered by the LLAMA 3.1 model and selects suitable candidates for further evaluation or consideration. This use case ensures that only the top candidates are shortlisted.

**7. Send Email Notification to Selected Candidates**:

- Once candidates are selected, the Admin sends email notifications to the selected candidates, informing them about the next steps or their application status. This feature includes automated email generation for efficiency.

**3.Relationships in the Diagram**

- **Includes**: This relationship signifies that one use case is part of another. For example:
    - "View Job Roles" is part of "Apply Job Roles," as applicants need to view available roles before applying.
    - "Filter Resumes Using LLAMA 3.1 Model" is part of "Apply Job Roles" and "Select Candidates" to ensure resumes are filtered after submission and before selection.
- **Extends**: This relationship indicates optional or additional functionality, allowing extended processes based on certain conditions.
    - "Login" extends to various use cases like "Apply Job Roles" and "Posts Job Roles," ensuring only authenticated users can access these features.

## 6.3 SEQUENCE DIAGRAM

### Sequence Diagram Description for Job Fit Pro

The sequence diagram for the Job Fit Pro system details the workflow that enables candidates to apply for jobs and automatically have their resumes ranked against job requirements using a machine learning model (LLAMA 3.1). This system also facilitates admins in managing job postings, viewing ranked applications, and notifying shortlisted candidates.

**1.Candidate Views Available Jobs**

- The process begins with the **Candidate** interacting with the **Job Posting System** to view available job listings.
- The **Job Posting System** fetches the job listings and displays them to the candidate.

**2.Candidate Applies for a Job**

- After reviewing job listings, the **Candidate** selects a job and submits their application along with their resume.
- The **Application System** receives the application and forwards it to the **LLAMA 3.1 Model** for evaluation.

**3. ML Model Ranks Resume**

- The LLAMA 3.1 Model analyzes the candidate's resume and calculates a match score based on how closely the resume aligns with the job requirements.
- After evaluation, the LLAMA 3.1 Model returns a ranked score for the resume to the Application System.

**4. Update Application Status**

- After evaluation, the LLAMA 3.1 Model returns a ranked score for the resume to the Application System. Based on the ranking received, the **Application System** updates the application status for the candidate (e.g., "Under Review," "Shortlisted").

**5.Notification to Candidate**

- After evaluation, the LLAMA3.1 Model returns a ranked score for the resume to the Application System.

- If the candidate is shortlisted, the Application System triggers the Notification System to send a notification email to the candidate regarding their application status.

**6.Admin Reviews Applications**

- The **Admin** logs into the system and accesses the **Application System** to review the applications for the job.
- The admin can see the ranked list of candidates and may choose to further shortlist top candidates based on their scores and other criteria.

**7. Generate Reports (Optional)**

- The **Admin** can use the **Report Generator** to create reports on the job applications, such as the number of applicants, average match scores, and other metrics for analysis.

**8. Display Application Status to Candidate**

- The **Candidate** can log in to the **Application System** at any time to view the current status of their application.

## 6.4 CLASS DIAGRAM

**1.Candidate Views Available Jobs**

- The process begins with the Candidate interacting with the Job Posting System to view available job listings.
- The Job Posting System fetches the job listings and displays them to the candidate.

**2.Candidate Applies for a Job**

- After reviewing job listings, the Candidate selects a job and submits their application along with their resume.

- The Application System receives the application and forwards it to the LLAMA 3.1 Model for evaluation.

**3. ML Model Ranks Resume**

- The LLAMA 3.1 Model analyzes the candidate's resume and calculates a match score based on how closely the resume aligns with the job requirements.
- After evaluation, the LLAMA 3.1 Model returns a ranked score for the resume to the Application System.

**4. Update Application Status**

- After evaluation, the LLAMA 3.1 Model returns a ranked score for the resume to the Application System. Based on the ranking received, the Application System updates the application status for the candidate (e.g., "Under Review," "Shortlisted").

**5.Notification to Candidate**

- After evaluation, the LLAMA3.1 Model returns a ranked score for the resume to the Application System.
- If the candidate is shortlisted, the Application System triggers the Notification System to send a notification email to the candidate regarding their application status.

**6.Admin Reviews Applications**

- The Admin logs into the system and accesses the Application System to review the applications for the job.
- The admin can see the ranked list of candidates and may choose to further shortlist top candidates based on their scores and other criteria.

# CHAPTER 7
# SOFTWARE DEVELOPMENT LIFECYCLE

## 7.1 WATERFALL MODEL

For the Job Fit Pro project, which automates resume filtering for companies using the Meta Llama 3.1 8b instruct model, we followed the Waterfall Model, a structured, sequential development approach. In the Requirements Gathering and Analysis phase, we collaborated with stakeholders to identify core functionalities, such as user login, job search, application submission, and resume ranking, while determining the technical stack (Django for the backend, HTML/CSS for the frontend, and Meta Llama for ML-based resume evaluation). During System Design, we planned a robust architecture, defining components like User, JobPosting, Application, and Admin, and developed UI mockups for intuitive navigation. The Implementation phase saw the translation of design into code, with Django handling backend logic and ML model integration for automated resume filtering. Frontend development ensured a user-friendly interface for candidates and admins. Unit Testing was performed on each module to ensure functionality, covering areas like login, application submission, and resume ranking. Following this, the system was deployed, with Django and Meta Llama seamlessly integrated, ensuring smooth user interaction. Finally, maintenance plans were established for ongoing system monitoring and updates. The Waterfall Model facilitated a clear, methodical progression through each phase, ensuring that each requirement was thoroughly addressed before moving on to the next, providing a stable foundation for the final application.

# CHAPTER 8
# PROGRAM CODE AND OUTPUTS

## 8.1 SAMPLE CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Job FitPro - Homepage</title>
   {% load static %}
   <link rel="stylesheet" href="{% static 'css/style.css' %}">
   <script src="script.js" defer></script>
   <style>
      /* Resetting some basic styles */
* {
   margin: 0;
   padding: 0;
   box-sizing: border-box;
}

/* Navbar styling */
.navbar {
   display: flex;
   justify-content: space-between;
   align-items: center;
   padding: 20px;
   background-color: #2a2a72;
   color: white;
}
```

```
.navbar .logo {
   font-size: 24px;
   font-weight: bold;
}

.nav-links {
   list-style: none;
}

.nav-links li {
   display: inline;
   margin-left: 20px;
}

.nav-links a {
   color: white;
   text-decoration: none;
   font-weight: bold;
   transition: color 0.3s;
}

/* Styling for regular links */
.nav-links a:not(.button) {
   color: white;
}

.nav-links a:hover {
   color: #ffb703;
}

/* Button styles for Sign Up and Login */
```

```
.button {
   padding: 8px 16px;
   border-radius: 4px;
   font-weight: bold;
   text-decoration: none;
}

.sign-up {
   background-color: #008080;
   color: #2a2a72;
   border: 1px solid transparent;
   transition: background-color 0.3s, color 0.3s;
}

.sign-up:hover {
   background-color: #008080;
   color: #fff;
}

.login {
   background-color: transparent;
   color: white;
   border: 2px solid #008080;
   transition: background-color 0.3s, color 0.3s;
}

/* .login:hover {
   background-color: #ffb703;
   color: white;
} */
```

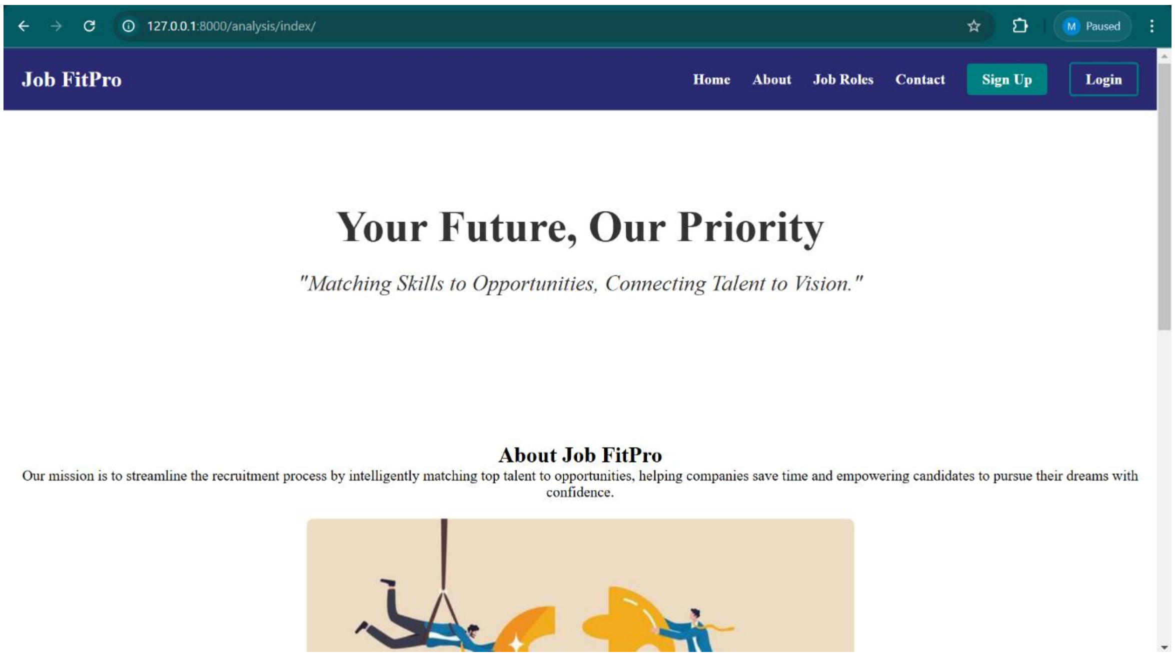
```
/* Hero Section styling */
.hero {
   background-image: url('./interview.avif');
   background-size: cover;
   color: #333333;
   text-align: center;
   padding: 100px 20px;
}

.hero h1 {
   font-size: 48px;
   margin-bottom: 20px;
}

.hero p {
   font-size: 24px;
   font-style: italic;
}

/* About Section styling */
.about, .goals {
   padding: 60px 20px;
   text-align: center;
}

.about img, .goal-images img {
   margin-top: 20px;
   max-width: 100%;
   border-radius: 8px;
}
```

```
/* Goals Section styling */
.goal-images {
   display: flex;
   justify-content: center;
   gap: 20px;
   margin-top: 20px;
}

/* Footer styling */
footer {
   background-color: #2a2a72;
   color: white;
   text-align: center;
   padding: 10px 0;
   font-size: 14px;
}
.auth-section {
   padding: 60px 20px;
   text-align: center;
   background-color: #f9f9f9; /* Light gray background for contrast */

}

.auth-section h2 {
   font-size: 36px;
   margin-bottom: 20px;
}

.auth-section form {
   max-width: 400px;
   margin: 0 auto;
```

```
    text-align: left;
}

.auth-section label {
    margin-top: 10px;
    display: block;
    font-weight: bold;
}

.auth-section input {
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    border: 1px solid #ccc;
    border-radius: 4px;
}

.auth-section .button {
    margin-top: 20px;
    width: 100%;
}

.auth-section p {
    margin-top: 20px;
}


.google-signup {
    background-color: #db4437; /* Google Red */
    color: white;
    padding: 10px 16px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-weight: bold;
    transition: background-color 0.3s;
    margin-bottom: 20px; /* Space between Google button and form */
}

.google-signup:hover {
    background-color: #c13527; /* Darker red on hover */
}

    </style>
</head>
<body>
    <!-- Navbar -->
    <nav class="navbar">
        <div class="logo">Job FitPro</div>
        <ul class="nav-links">
            <li><a href="#">Home</a></li>
            <li><a href="{% url 'about' %}">About</a></li>
            <li><a href="{% url 'jobroles' %}">Job Roles</a></li>
            <li><a href="{% url 'contact' %}">Contact</a></li>
            <li><a href="{% url 'signup' %}" class="button sign-up">Sign Up</a></li>
            <li><a href="{% url 'login' %}" class="button login">Login</a></li>
        </ul>
    </nav>

    <!-- Hero Section with Quote -->
    <header class="hero">
        <h1>Your Future, Our Priority</h1>
        <p>"Matching Skills to Opportunities, Connecting Talent to Vision."</p>
    </header>

    <!-- About Section -->
    {% load static %}
    <section class="about">
        <h2>About Job FitPro</h2>
        <p>Our mission is to streamline the recruitment process by intelligently
matching top talent to opportunities,
            helping companies save time and empowering candidates to pursue their
dreams with confidence.</p>
        <img src="{% static 'images/collab.jpeg' %}" width = "600px"
height="400px" alt="Recruitment Image">
    </section>

    <!-- Goals Section -->
    <!-- {% load static %} -->
    <section class="goals">
        <h2>Our Goal</h2>
        <p>Job FitPro aims to harness the power of AI to create a seamless and
efficient recruitment experience.
            Our model ranks and filters resumes to help companies connect with the best
candidates quickly and easily.</p>
        <div class="goal-images">
            <img src="{% static 'images/airecruit.avif' %}" width = "300px"
height="200px" alt="Goal Image 1">
            <img src="{% static 'images/resumereview.avif' %}" width = "300px"
height="200px"alt="Goal Image 2">
            <img src="{% static 'images/inter.avif' %}" width = "300px"
height="200px"alt="Goal Image 3">
        </div>
```

```
  </section>
  <!-- Footer -->
  <footer>
    <p>&copy; 2024 Job FitPro. All rights reserved.</p>
  </footer>
</body>
</html>

from django.shortcuts import render

# Create your views here.
import os
from django.shortcuts import render
from django.http import HttpResponse, JsonResponse
from .models import UploadedFiles
from django.conf import settings
from PyPDF2 import PdfReader
import ollama
import concurrent.futures



def load_text_from_pdf(pdf_file):
  pdf_reader = PdfReader(pdf_file)
  text = ""
  for page in pdf_reader.pages:
    text += page.extract_text()
  return text

def analyze_resume_job_description(resume_text, job_description_text):
  response = ollama.chat(model="llama3.1", messages=[
        {'role': 'user', 'content': f"Analyze the following resume and job description to
determine if the candidate is suitable for the job and give me percentage of
suitability.Mention percentage of suitability to job role Resume: {resume_text} Job
Description: {job_description_text}"}
    ])
    return response['message']['content']

def upload_resume(request):
    if request.method == 'POST':
        # Get uploaded resume file and job description from the form
        resume_file = request.FILES['resume']
        job_description_text = request.POST['job_description']

        # Extract text from the PDF
        resume_text = load_text_from_pdf(resume_file)

        # Analyze the resume against the job description
        analysis_result = analyze_resume_job_description(resume_text,
job_description_text)

        # Save the result to a text file
        output_file_path = os.path.join(settings.MEDIA_ROOT, 'output.txt')
        with open(output_file_path, 'w') as file:
            file.write(analysis_result)

        return JsonResponse({'result': analysis_result, 'file_url': f'/media/output.txt'})

    return render(request, 'analysis/upload.html')
def about(request):
    return render(request, 'analysis/about.html')
def contact(request):
    return render(request, 'analysis/contact.html')
def index(request):
    return render(request, 'analysis/index.html')
def login(request):
    return render(request, 'analysis/login.html')
def signup(request):
    return render(request, 'analysis/signup.html')
def jobroles(request):
    return render(request, 'analysis/jobroles.html')
```

## 8.2 OUTPUT SCREENSHOTS:



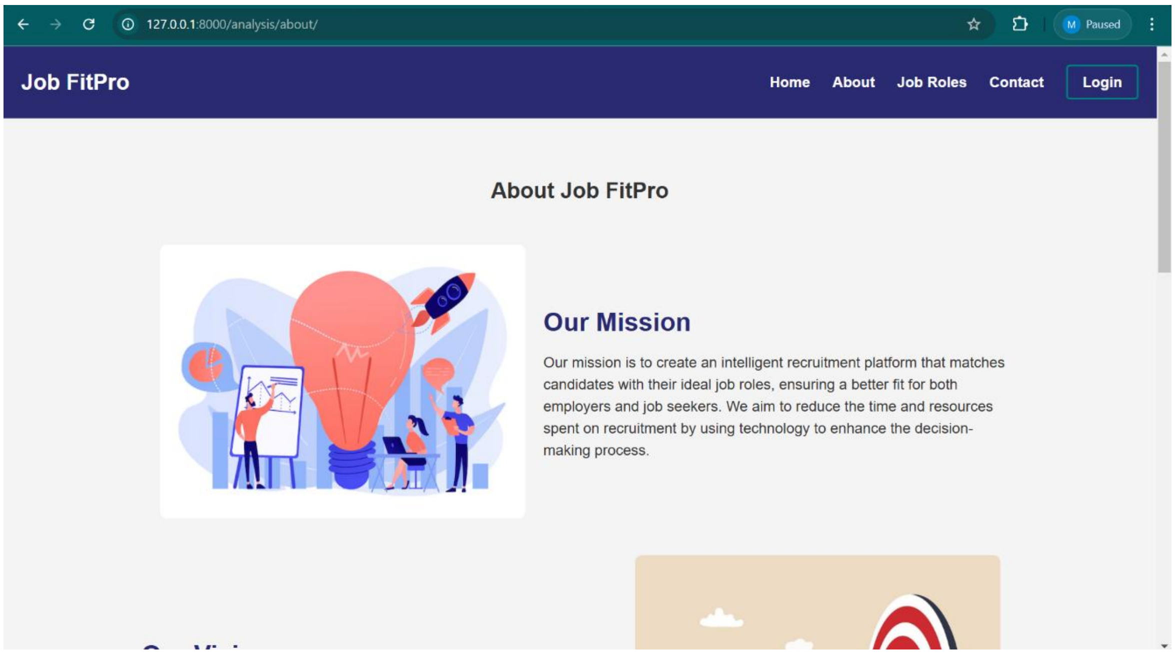
Your Future, Our Priority

"Matching Skills to Opportunities, Connecting Talent to Vision."

About Job FitPro

Our mission is to streamline the recruitment process by intelligently matching top talent to opportunities, helping companies save time and empowering candidates to pursue their dreams with confidence.

Figure 2. Home Page

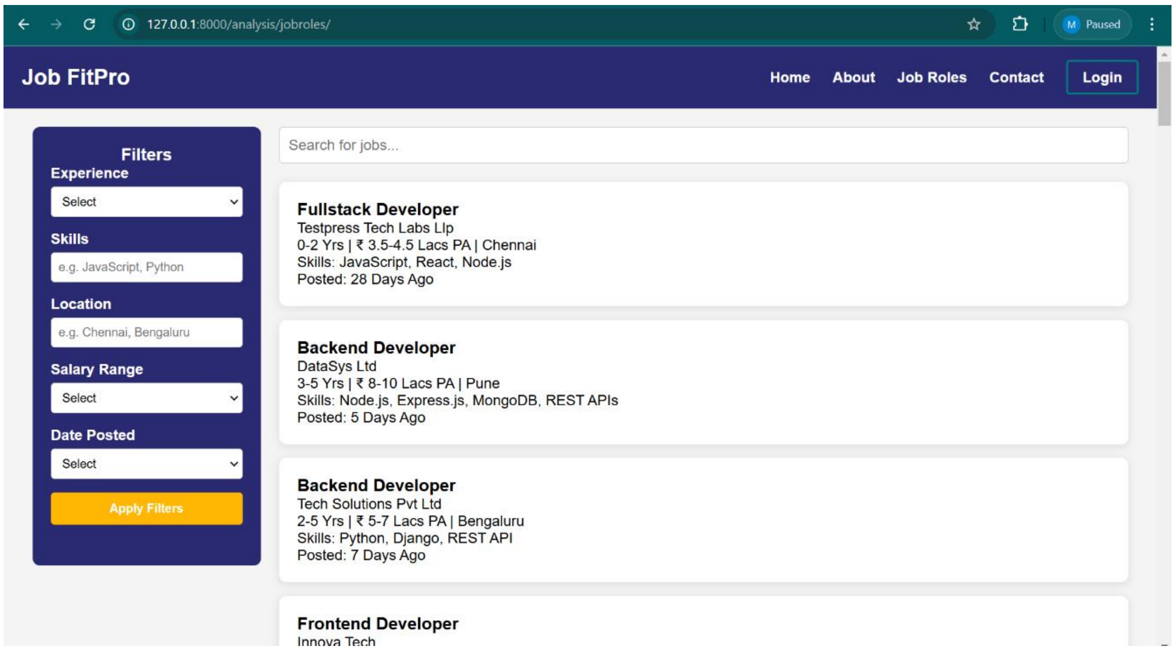


About Job FitPro

Our Mission

Our mission is to create an intelligent recruitment platform that matches candidates with their ideal job roles, ensuring a better fit for both employers and job seekers. We aim to reduce the time and resources spent on recruitment by using technology to enhance the decision-making process.

Figure 3. About page



127.0.0.1:8000/analysis/jobroles/

Job FitPro — Home | About | Job Roles | Contact | Login

**Filters**

Experience: Select

Skills: e.g. JavaScript, Python

Location: e.g. Chennai, Bengaluru

Salary Range: Select

Date Posted: Select

Apply Filters

Search for jobs...

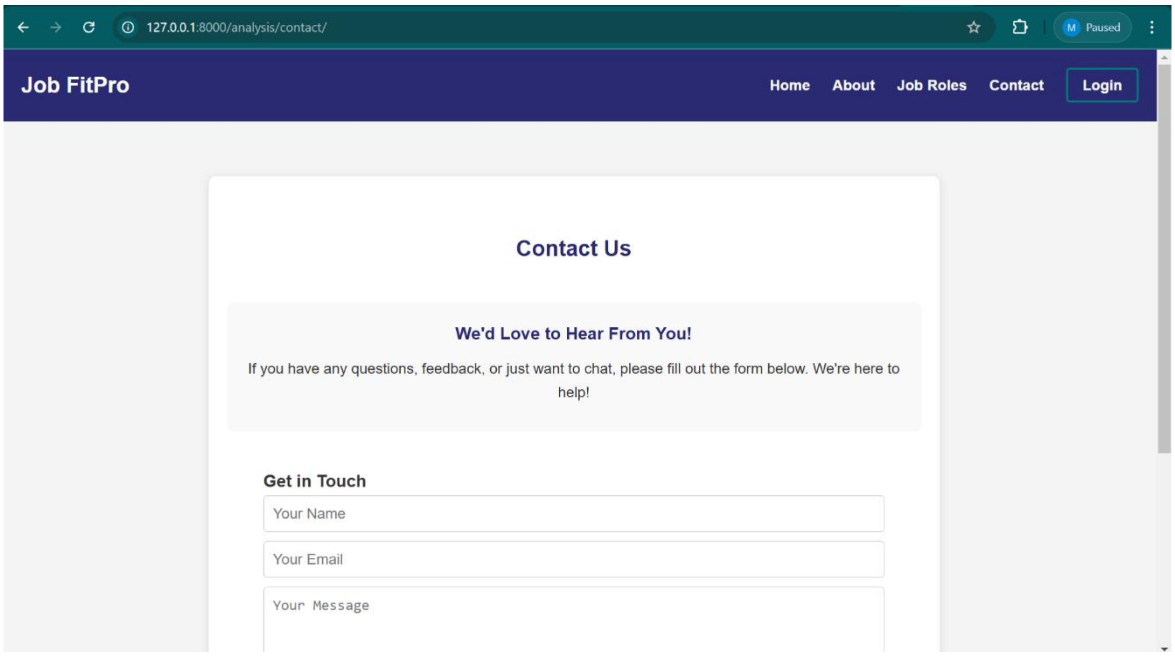
**Fullstack Developer**
Testpress Tech Labs Llp
0-2 Yrs | ₹ 3.5-4.5 Lacs PA | Chennai
Skills: JavaScript, React, Node.js
Posted: 28 Days Ago

**Backend Developer**
DataSys Ltd
3-5 Yrs | ₹ 8-10 Lacs PA | Pune
Skills: Node.js, Express.js, MongoDB, REST APIs
Posted: 5 Days Ago

**Backend Developer**
Tech Solutions Pvt Ltd
2-5 Yrs | ₹ 5-7 Lacs PA | Bengaluru
Skills: Python, Django, REST API
Posted: 7 Days Ago

**Frontend Developer**
Innova Tech

Figure 4. Job Searching Page



127.0.0.1:8000/analysis/contact/

Job FitPro — Home | About | Job Roles | Contact | Login

**Contact Us**

**We'd Love to Hear From You!**

If you have any questions, feedback, or just want to chat, please fill out the form below. We're here to help!

**Get in Touch**

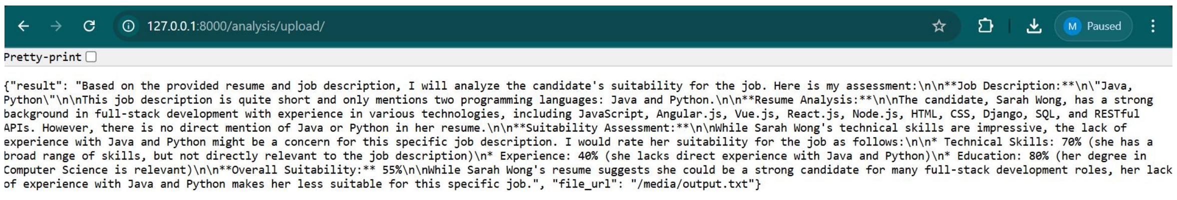
Your Name

Your Email

Your Message

Figure 5. Contact Page

Figure 6 . Ranking Canditates based on Resume



```
{"result": "Based on the provided resume and job description, I will analyze the candidate's suitability for the job. Here is my assessment:\n\n**Job Description:**\n\"Java,
Python\"\n\nThis job description is quite short and only mentions two programming languages: Java and Python.\n\n**Resume Analysis:**\n\nThe candidate, Sarah Wong, has a strong
background in full-stack development with experience in various technologies, including JavaScript, Angular.js, Vue.js, React.js, Node.js, HTML, CSS, Django, SQL, and RESTful
APIs. However, there is no direct mention of Java or Python in her resume.\n\n**Suitability Assessment:**\n\nWhile Sarah Wong's technical skills are impressive, the lack of
experience with Java and Python might be a concern for this specific job description. I would rate her suitability for the job as follows:\n\n* Technical Skills: 70% (she has a
broad range of skills, but not directly relevant to the job description)\n* Experience: 40% (she lacks direct experience with Java and Python)\n* Education: 80% (her degree in
Computer Science is relevant)\n\n**Overall Suitability:** 55%\n\nWhile Sarah Wong's resume suggests she could be a strong candidate for many full-stack development roles, her lack
of experience with Java and Python makes her less suitable for this specific job.", "file_url": "/media/output.txt"}
```

Figure 6. Ranking Candidates based on Resume

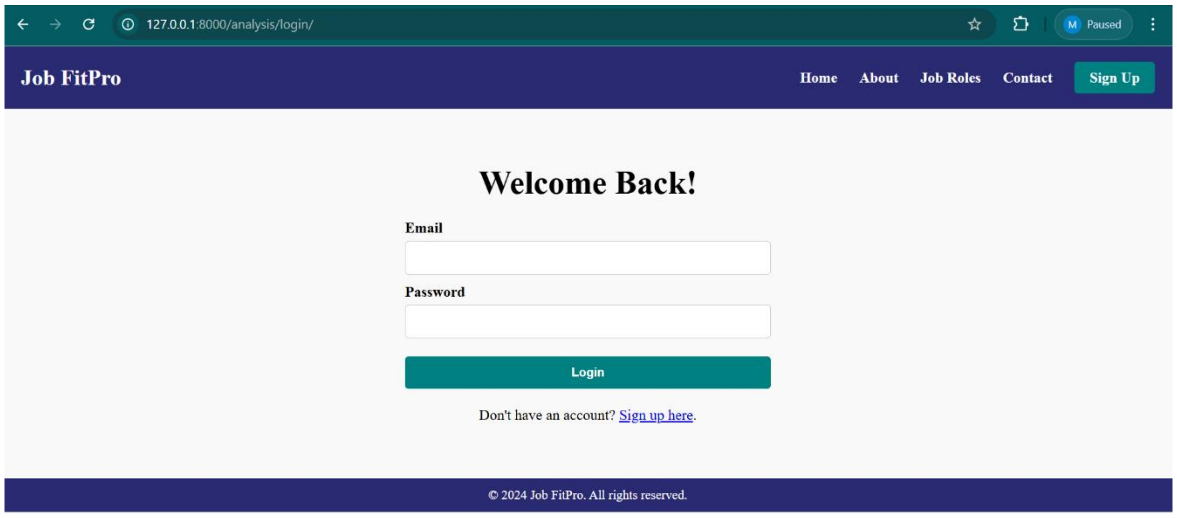

127.0.0.1:8000/analysis/login/
Paused
Job FitPro
Home
About
Job Roles
Contact
Sign Up
Welcome Back!
Email
Password
Login
Don't have an account? Sign up here.
© 2024 Job FitPro. All rights reserved.

Figure 7. Login Page

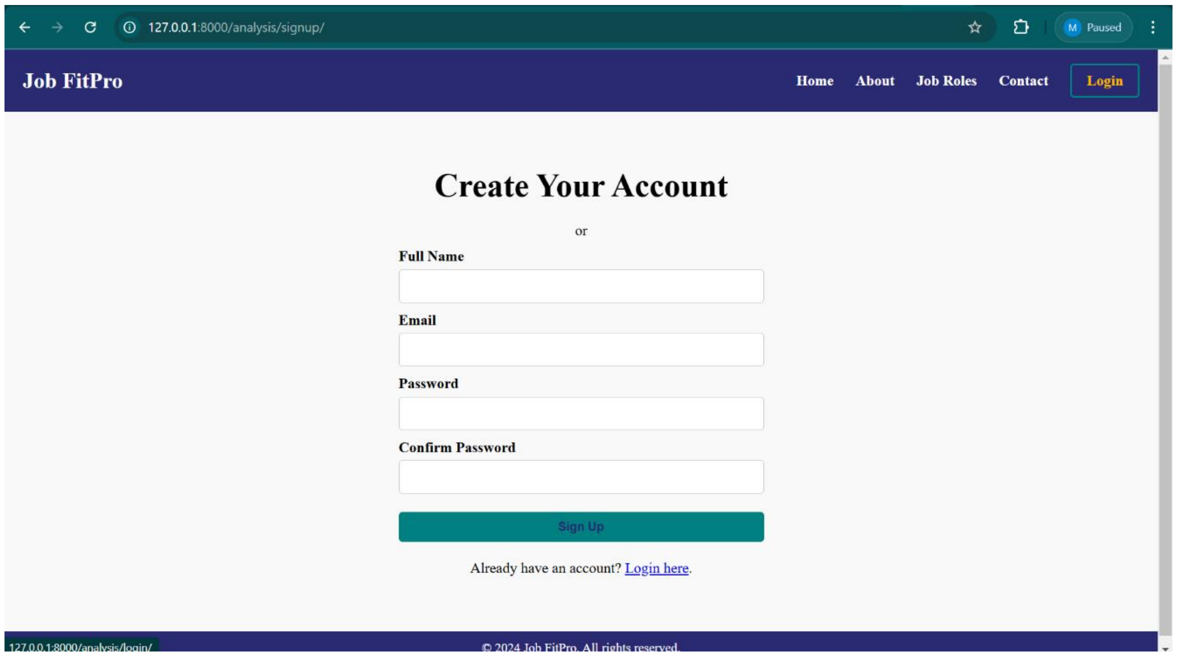

127.0.0.1:8000/analysis/signup/
Paused
Job FitPro
Home
About
Job Roles
Contact
Login
Create Your Account
or
Full Name
Email
Password
Confirm Password
Sign Up
Already have an account? Login here.
127.0.0.1:8000/analysis/login/
© 2024 Job FitPro. All rights reserved.

Figure 8. Sign up Page

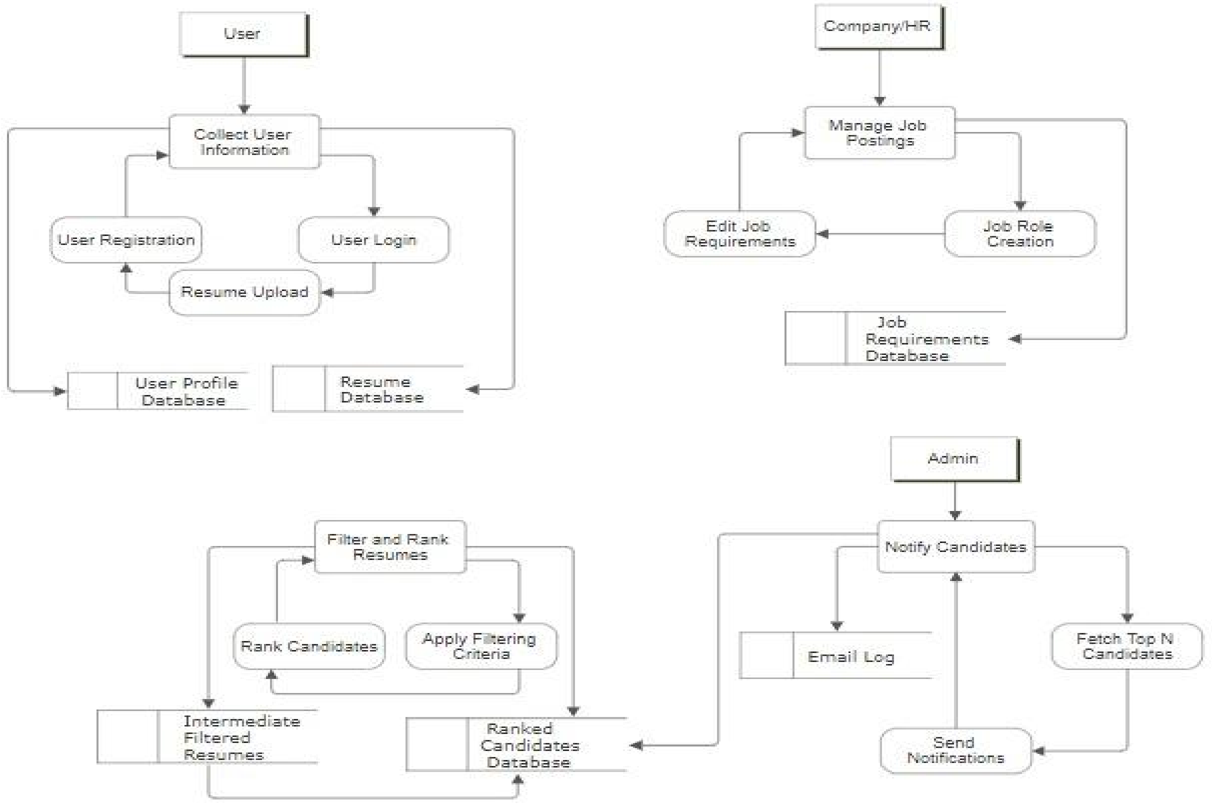


Figure 9. Data Flow Diagram

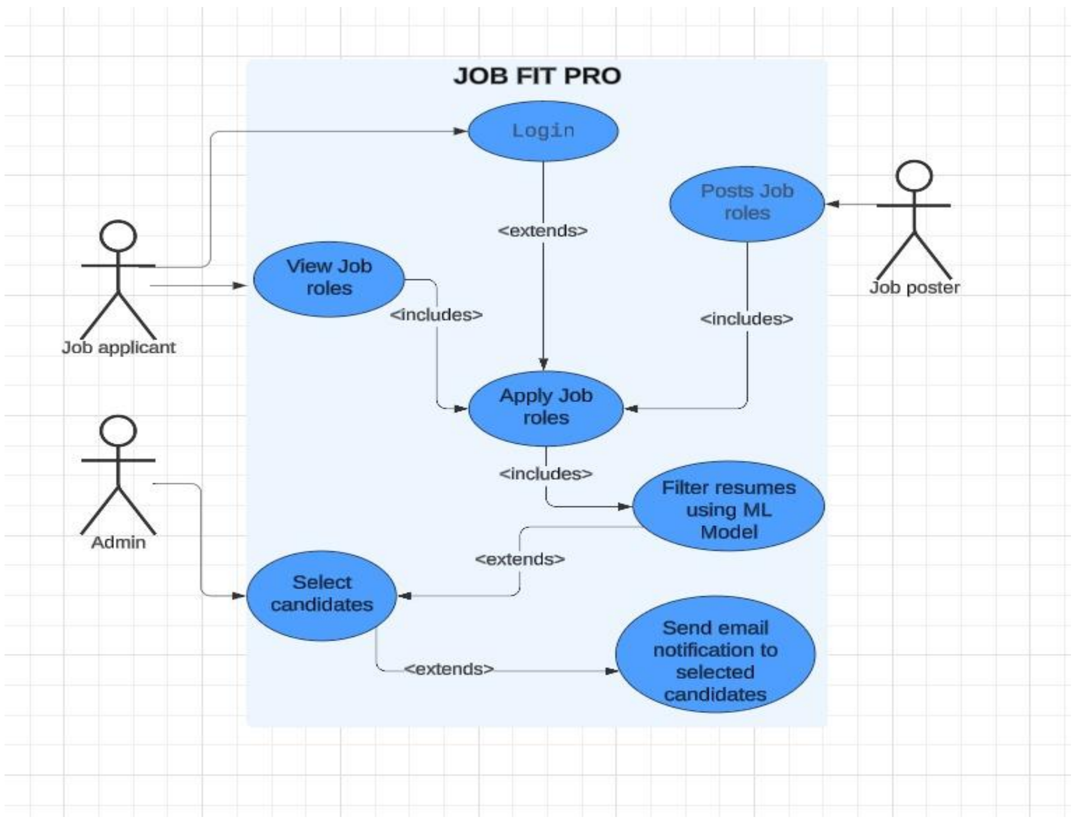


Figure 10. Use Case Diagram

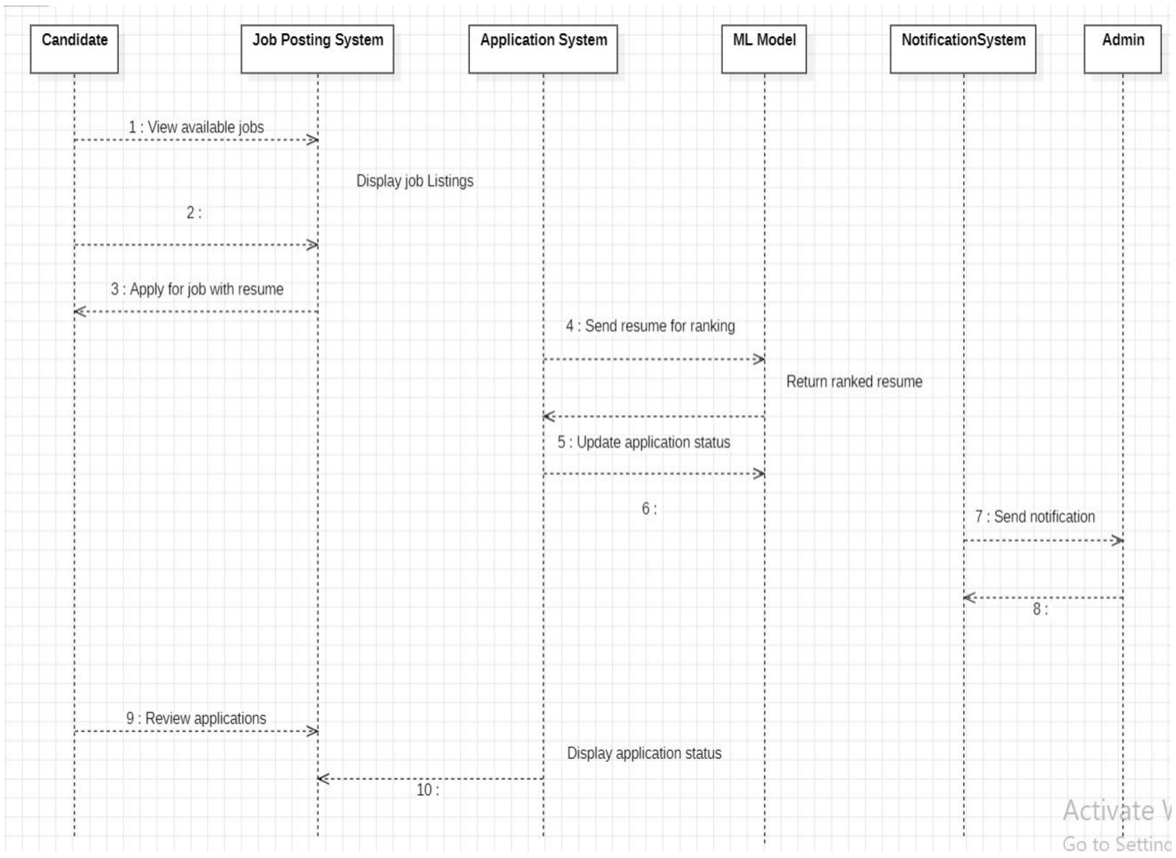


Figure 11. Sequence Diagram

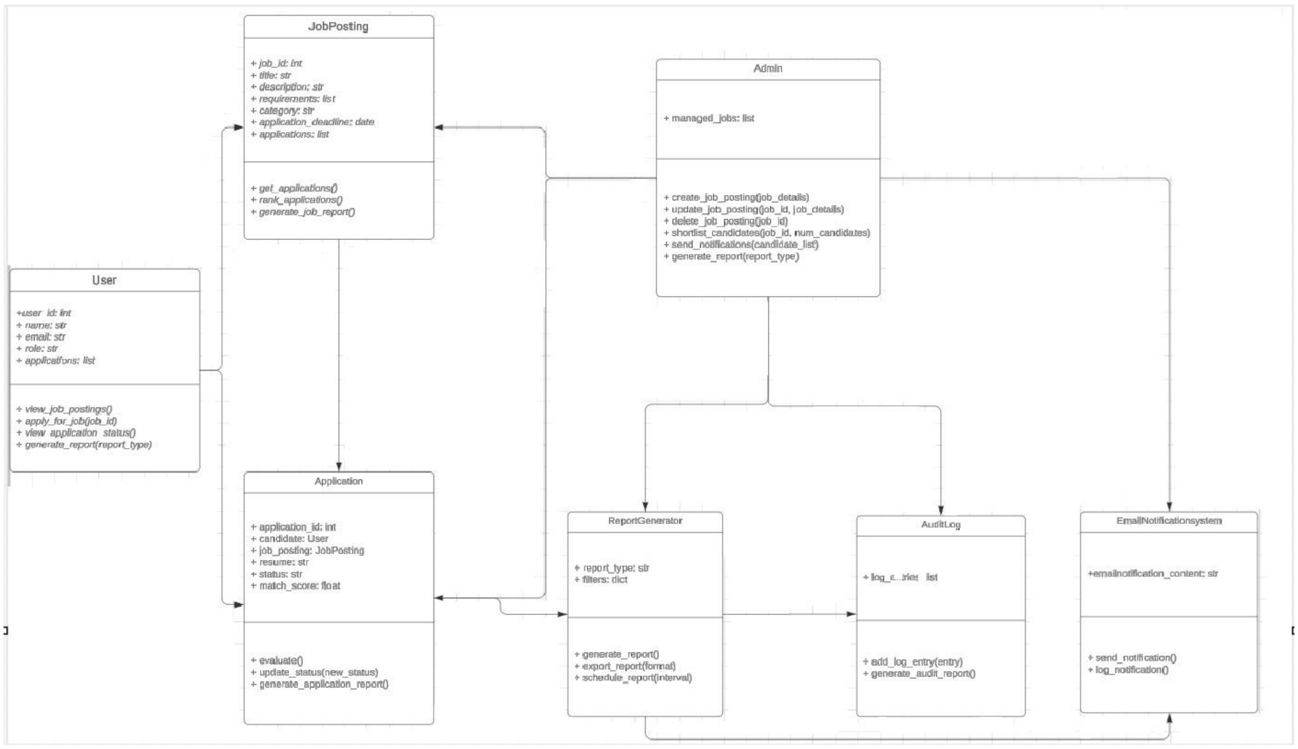


Figure 12. Class Diagram

# CHAPTER 9
# TESTING

## 9.1 UNIT TESTING

### Unit Testing Description

In our Django application, we have developed unit tests to verify the core functionality and reliability of critical components, specifically focusing on file upload handling, resume-job description analysis, and structured storage of uploaded files. Each test case is structured to evaluate a specific aspect of the application and confirm that expected behaviors are met under controlled conditions. Below is a detailed description of each test case and its purpose.

### Test Case 1: Uploaded Files Model Test

**Objective:** To validate that files uploaded to the UploadedFiles model are saved with the expected filenames, specifically ensuring the resume file name starts with the intended prefix.

**Description:** This test creates mock files (resume2.pdf for resume and job_description.pdf for the job description) using SimpleUploadedFile. It then instantiates the UploadedFiles model with these files and verifies if the stored resume file name starts with "resumes/resume2", ensuring files are saved with a structured naming convention.

**Outcome:** If successful, the test confirms the model correctly saves and names uploaded files.

### Test Case 2: Resume Analysis Function Test

**Objective:** To check if the analyze_resume_job_description function provides an analysis result that includes the expected output format (like "percentage" match) when given text from a resume and job description.

**Description:** This unit test directly tests the analysis function, using a mock resume text containing technical skills and a job description text with similar requirements.

The test checks if the function returns a response containing a "percentage" key, indicating the match score between resume and job description.

**Outcome:** This verifies that the function processes inputs correctly and returns results in the anticipated format.

**Test Case 3: File Upload View Test**

**Objective:** To ensure the file upload view (upload_resume) functions as expected, including handling file uploads and job description text submission.

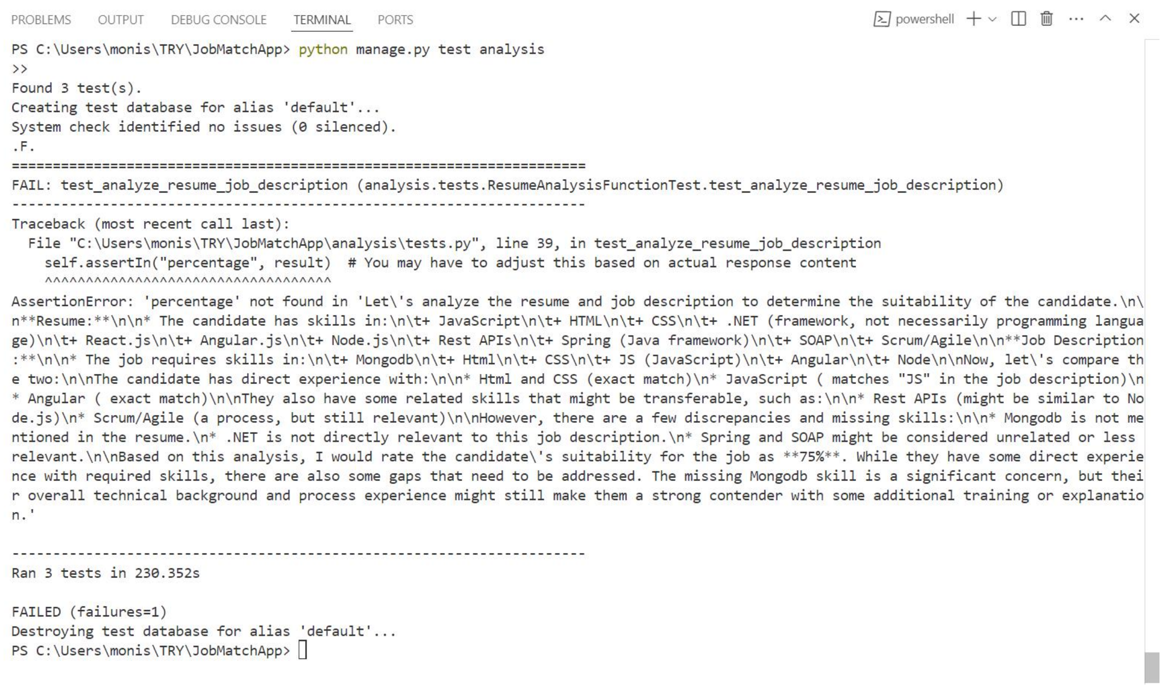
Description: This test simulates a file upload and text submission through the Django test client. A sample PDF file (resume2.pdf) and job description text are provided. The test then posts this data to the upload_resume view, validating that the response status is 200 and contains the expected keyword ("result") to confirm successful processing.

**Outcome:** This confirms that the file upload and form submission are working and that the application responds appropriately to valid inputs.

**Report Summary Example:**

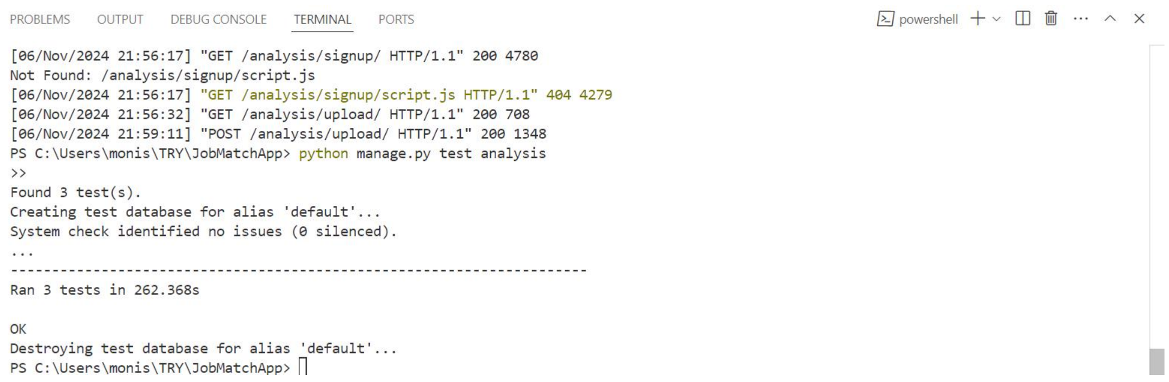
In our Django application, unit tests cover key components to ensure the application's core functionality works as expected. The first test case verifies file naming conventions in the model, the second checks the logic of the resume-job description matching function, and the third tests the file upload view to confirm it processes inputs and returns responses successfully. The tests passed, confirming the system handles uploads, model saving, and analysis functionality as designed.

## 9.2 TESTING SCREENSHOTS:



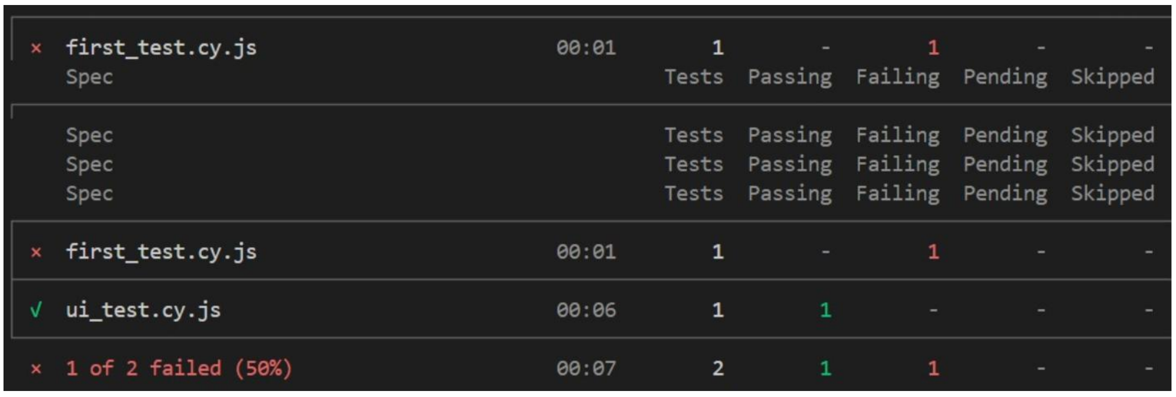
```
PS C:\Users\monis\TRY\JobMatchApp> python manage.py test analysis
>>
Found 3 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.F.
======================================================================
FAIL: test_analyze_resume_job_description (analysis.tests.ResumeAnalysisFunctionTest.test_analyze_resume_job_description)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\monis\TRY\JobMatchApp\analysis\tests.py", line 39, in test_analyze_resume_job_description
    self.assertIn("percentage", result)  # You may have to adjust this based on actual response content
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: 'percentage' not found in 'Let\'s analyze the resume and job description to determine the suitability of the candidate.\n\
n**Resume:**\n\n* The candidate has skills in:\n\t+ JavaScript\n\t+ HTML\n\t+ CSS\n\t+ .NET (framework, not necessarily programming langua
ge)\n\t+ React.js\n\t+ Angular.js\n\t+ Node.js\n\t+ Rest APIs\n\t+ Spring (Java framework)\n\t+ SOAP\n\t+ Scrum/Agile\n\n**Job Description
:**\n\n* The job requires skills in:\n\t+ Mongodb\n\t+ Html\n\t+ CSS\n\t+ JS (JavaScript)\n\t+ Angular\n\t+ Node\n\nNow, let\'s compare th
e two:\n\nThe candidate has direct experience with:\n\n* Html and CSS (exact match)\n* JavaScript ( matches "JS" in the job description)\n
* Angular ( exact match)\n\nThey also have some related skills that might be transferable, such as:\n\n* Rest APIs (might be similar to No
de.js)\n* Scrum/Agile (a process, but still relevant)\n\nHowever, there are a few discrepancies and missing skills:\n\n* Mongodb is not me
ntioned in the resume.\n* .NET is not directly relevant to this job description.\n* Spring and SOAP might be considered unrelated or less
relevant.\n\nBased on this analysis, I would rate the candidate\'s suitability for the job as **75%**. While they have some direct experie
nce with required skills, there are also some gaps that need to be addressed. The missing Mongodb skill is a significant concern, but thei
r overall technical background and process experience might still make them a strong contender with some additional training or explanatio
n.'

----------------------------------------------------------------------
Ran 3 tests in 230.352s

FAILED (failures=1)
Destroying test database for alias 'default'...
PS C:\Users\monis\TRY\JobMatchApp>
```

Figure 13. Testing of Functions



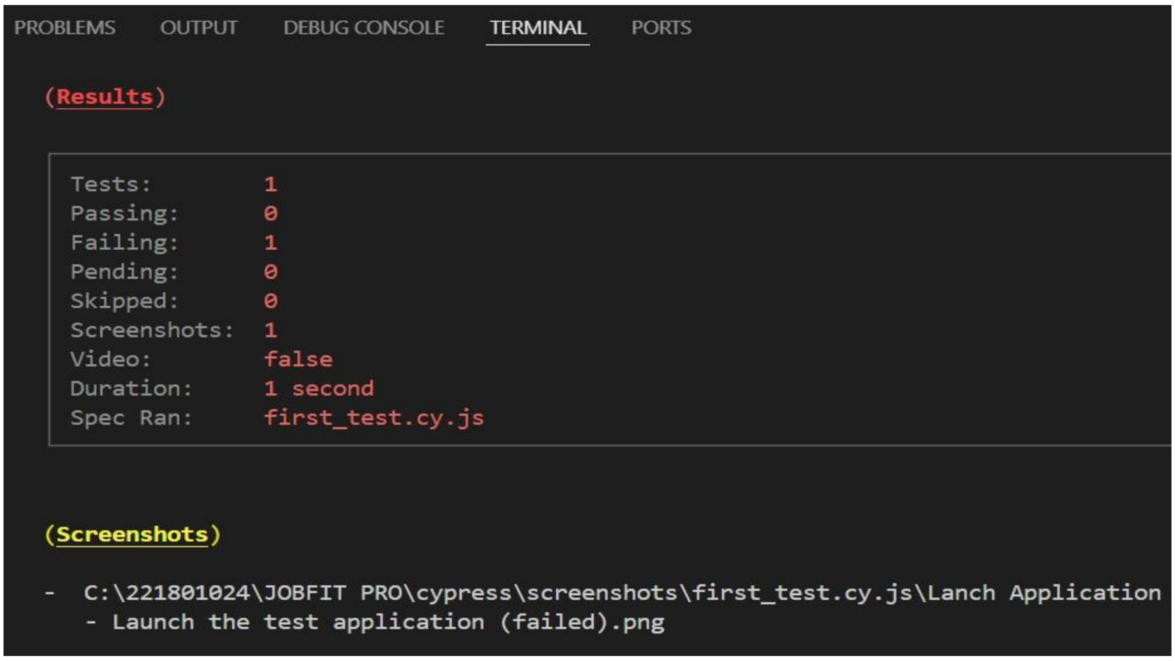
```
[06/Nov/2024 21:56:17] "GET /analysis/signup/ HTTP/1.1" 200 4780
Not Found: /analysis/signup/script.js
[06/Nov/2024 21:56:17] "GET /analysis/signup/script.js HTTP/1.1" 404 4279
[06/Nov/2024 21:56:32] "GET /analysis/upload/ HTTP/1.1" 200 708
[06/Nov/2024 21:59:11] "POST /analysis/upload/ HTTP/1.1" 200 1348
PS C:\Users\monis\TRY\JobMatchApp> python manage.py test analysis
>>
Found 3 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...
----------------------------------------------------------------------
Ran 3 tests in 262.368s

OK
Destroying test database for alias 'default'...
PS C:\Users\monis\TRY\JobMatchApp>
```

Figure 14. Test cases Passed



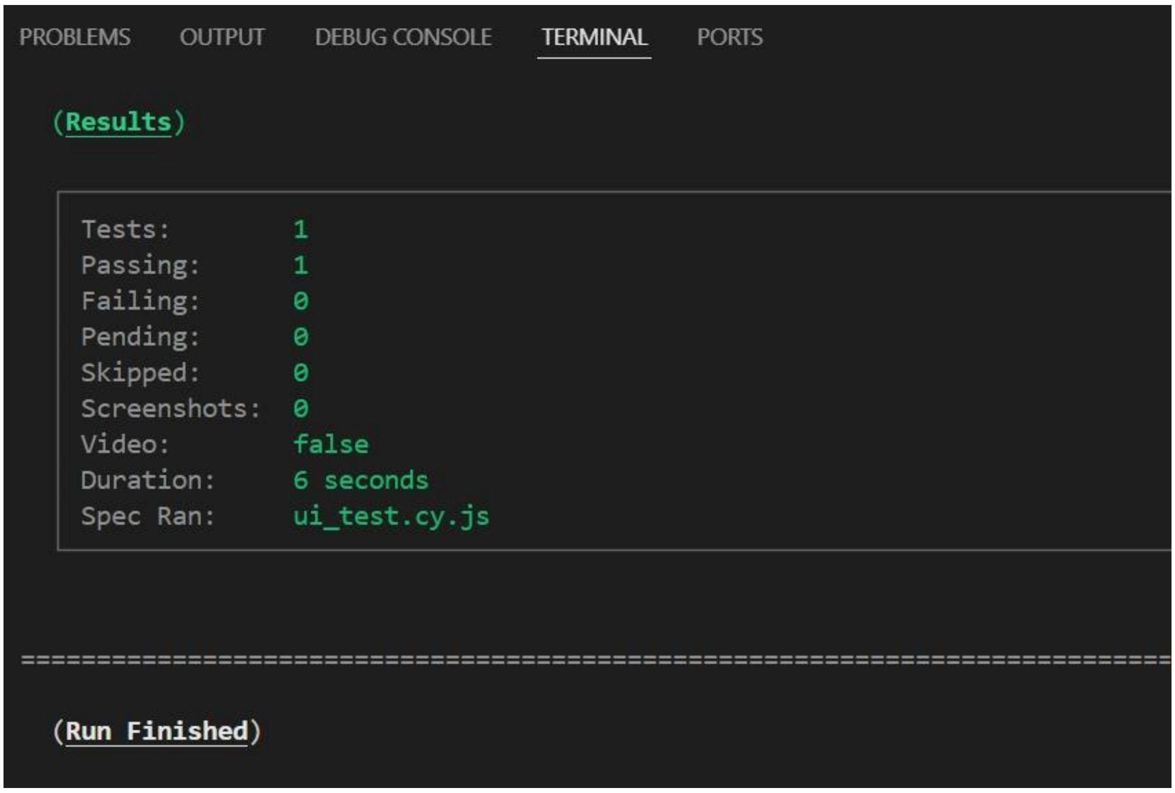
```
  ×  first_test.cy.js                    00:01      1        -        1        -        -
     Spec                                        Tests  Passing  Failing  Pending  Skipped

     Spec                                        Tests  Passing  Failing  Pending  Skipped
     Spec                                        Tests  Passing  Failing  Pending  Skipped
     Spec                                        Tests  Passing  Failing  Pending  Skipped

  ×  first_test.cy.js                    00:01      1        -        1        -        -

  √  ui_test.cy.js                       00:06      1        1        -        -        -

  ×  1 of 2 failed (50%)                 00:07      2        1        1        -        -
```

Figure 15. UI Test



```
(Results)

  Tests:        1
  Passing:      0
  Failing:      1
  Pending:      0
  Skipped:      0
  Screenshots:  1
  Video:        false
  Duration:     1 second
  Spec Ran:     first_test.cy.js


(Screenshots)

-  C:\221801024\JOBFIT PRO\cypress\screenshots\first_test.cy.js\Lanch Application
   - Launch the test application (failed).png
```

Figure 16. UI Test case Failed



```
(Results)

  Tests:        1
  Passing:      1
  Failing:      0
  Pending:      0
  Skipped:      0
  Screenshots:  0
  Video:        false
  Duration:     6 seconds
  Spec Ran:     ui_test.cy.js


==========================================================================

(Run Finished)
```

Figure 17. UI Test case passed

# CHAPTER 10
# RESULTS AND DISCUSSION

## 10.1 RESULTS

The Job Fit Pro project, built with Django and integrated with the Meta Llama 3.1 8b instruct model, has successfully delivered several key functionalities essential for automating the recruitment process. The system's automated resume filtering and ranking feature efficiently processes submitted resumes, identifying the top 10 candidates based on job-specific requirements, which significantly streamlines hiring for employers. It provides a smooth and intuitive user experience, including features like seamless registration, easy login, and the ability to search and apply for suitable roles with straightforward resume submission. The HTML/CSS-based frontend ensures responsiveness, enhancing accessibility across devices. On the backend, Django effectively handles user authentication, job management, and integration with the LLM model. Initial unit testing has verified core functionalities such as user registration, job application, resume ranking, and role-based access, all of which contribute to a reliable and user-friendly system that minimizes hiring bias.

## 10.2 DISCUSSION

While Job Fit Pro demonstrates strong functionality, some areas could be improved for enhanced performance and user experience. Additional integration and system testing are recommended to ensure stability under higher traffic and larger datasets as the system scales. Gathering user feedback to refine the resume ranking algorithm could increase matching accuracy and further reduce biases. Optimizing backend processing and the LLM integration would also improve scalability and response times. Lastly, enhancements to the user interface—such as clearer application status updates and smoother navigation—would make the platform more user-friendly for job seekers and employers alike. Overall, with these refinements, Job Fit Pro could offer an even more efficient and scalable recruitment solution.

# CHAPTER 11
# CONCLUSION AND FUTURE ENHANCEMENT

## 11.1 CONCLUSION

The Job Fit Pro project has successfully developed a platform for automating the resume filtering process, leveraging the Meta Llama 3.1 8b instruct model to rank candidates based on how well their resumes match job-specific criteria. The system allows job seekers to apply for roles by submitting their personal details and resumes before the deadline, while providing employers with an efficient way to review and rank the top 10 candidates. The backend, powered by Django, handles critical operations such as user authentication, job management, and interaction with the locally run Meta Llama model, while the frontend, designed with HTML/CSS, ensures a smooth and responsive user experience. Additionally, email alerts have been incorporated to notify candidates of application statuses and deadlines, further enhancing the platform's usability.

## 11.2 FUTURE ENHANCEMENT

For future enhancements, the project could benefit from more comprehensive integration and system testing to ensure scalability and performance under varying conditions, especially with larger datasets or higher user traffic. Further optimization of the Meta Llama model, including industry-specific fine-tuning, could improve the accuracy of resume ranking, making the system even more relevant to various job sectors. Incorporating user feedback will be crucial in refining the ranking algorithm and enhancing the user interface to improve both candidate experience and admin workflow. Moreover, expanding the system’s functionality to support a wider range of job roles, industries, and additional features like real-time notifications and performance analytics could further strengthen the platform. Finally, exploring cloud-based hosting for model deployment would improve scalability and ensure the system can handle future growth more efficiently.

## REFERENCES

- K. Nakamura, S. Tanaka, and Y. Nagano, "Using Natural Language Processing for Resume Screening and Job Matching," IEEE Transactions on Engineering Management, vol. 67, no. 3, pp. 840-851, Sept. 2020.
- L. Tunstall, L. von Werra, and T. Wolf, Natural Language Processing with Transformers: Building Machine Learning Solutions Using BERT and GPT-3, O'Reilly Media, 2022.
- N. Gupta and R. Jain, "A Comparative Study of Techniques for Matching Job-Description and Resumes Using NLP," International Journal of Computer Applications, vol. 176, no. 41, pp. 20-27, Aug. 2020.
- M. Chang and P. Kumar, "Evaluating Resume Parsing Techniques Using Machine Learning for HR Applications," IEEE Access, vol. 8, pp. 72345-72354, 2020.
- L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: unbiased boosting with categorical features," arXiv preprint arXiv:1810.11363, 2018.
- P. Aggarwal and R. Sharma, "An NLP-Based Approach for Automated Resume Screening and Matching," IEEE International Conference on Advances in Computing, Communication, & Automation (ICACCA), pp. 1-6, Oct. 2021.
- A. Păduraru, L. Oprea, and D. Moldoveanu, "Efficient Resume Screening Using Text Processing and Machine Learning," IEEE 23rd International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, pp. 1016-1021, 2019.
- D. Jurafsky and J. H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 3rd ed., Pearson, 2023.