AI-Powered Resume Screening and Ranking System

A Project Report

submitted in partial fulfillment of the requirements

of

AICTE Internship on AI: Trans-formative Learning
with
TechSaksham – A joint CSR initiative of Microsoft & SAP

by

Mr.Sonawane Dinesh Shantilal

dineshsonawanew2004@gmail.com

Under the Guidance of

Saomya Sir & Pawan Sir

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to **Edunet Foundation** for providing me with the opportunity to complete my internship. This internship has been an invaluable learning experience that enabled me to work on the project **"AI-Powered Resume Screening and Ranking System."**

I extend my deepest thanks to my mentors and the entire team at Edunet Foundation for their continuous guidance, support, and encouragement throughout the internship. Their expertise and insights played a pivotal role in shaping the project and enhancing my technical knowledge.

The experience of developing an AI-powered system that automates and ranks resumes has helped me gain a deeper understanding of AI, machine learning, and real-world problem-solving. This journey has been an enriching and fulfilling part of my professional growth.

Once again, I am deeply thankful for this incredible opportunity and for the trust placed in me during the course of this internship.

ABSTRACT

In today's competitive job market, recruiters are inundated with resumes for each job posting, making the hiring process time-consuming and prone to human biases. This project aims to develop an AI-powered Resume Screening and Ranking System that leverages Natural Language Processing (NLP) and Machine Learning (ML) to automate resume parsing, job description matching, and candidate ranking. By incorporating semantic similarity techniques and bias detection mechanisms, the system will enhance hiring efficiency and fairness. This report details the system's design, methodology, implementation, and results, providing a comprehensive overview of its capabilities. This solution aims to improve recruitment accuracy, fairness, and scalability.

Keywords : *AI-powered Resume Screening , Natural Language Processing (NLP) , Machine Learning (ML) , Deep Learning (BERT , Transformers) , Job Description Matching , Bias Mitigation in Hiring , Optical Character Recognition (OCR) , Applicant Tracking System (ATS).*

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

Organizations always seek to hire employees who perfectly suit the job. Improper selection decisions for a new employee often have costly impacts on the work. Hence, Persons who stand behind the selection decision face an arduous task of selecting the most appropriate person from several applicants.

Recruitment is the process of searching, attracting, and hiring qualified applicants for employment in an organization . Figure 1 presents an overview of the key steps of the recruitment process.

A recruitment process starts with the advertising of an available job position. This is carried out using diverse advertising channels such as websites, newspapers, and others. Job seekers who are interested in that job will apply for the job by creating their profiles using a designated online form or uploading their resumes through the organization's website. Received applications are then screened to find out the suitable candidates to interview.

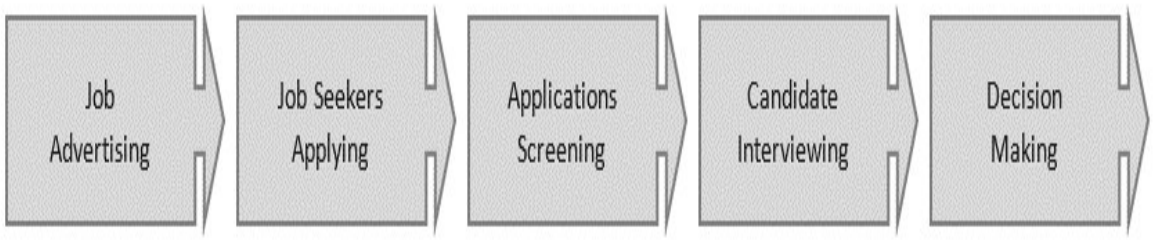

Figure 1: Overview of the recruitment process.

Screeners firstly should understand the requirement for the job. After that, they look through each of the submitted applications and reject applicants who do not meet the requirements. Finally, they find the best applicant who matches the job by comparing resumes with the job profile. The top few candidates listed during the screening stage will go along advanced stages in the process of evaluation, like interviews, written tests, and group discussions. The feedback received from the evaluation processes is used to make the final hiring decision. The candidate who passes the interview stage will be offered the position

.

1.1 Problem Statement:

The recruitment process is often inefficient due to the large volume of resumes recruiters must manually screen. Manual resume review is prone to human error, inconsistency, and unconscious bias, which can lead to the wrong candidates being shortlisted. Additionally, traditional systems often fail to identify candidates with transferable or implicitly stated skills. This necessitates an AI-powered solution that can automate resume screening and rank candidates based on their suitability for job descriptions.

1.2 Motivation:

With the growing volume of job applications and the demand for better hiring practices, there is a strong need for automation in resume screening. Automating this process can help reduce the workload on recruiters, minimize biases, and improve hiring accuracy. By leveraging NLP and AI, it is possible to not only match keywords but also infer transferable skills and identify suitable candidates more effectively. This project aims to streamline the hiring process and ensure fairness in candidate selection.

1.3 Objective:

The primary objectives of the AI-powered Resume Screening and Ranking System are:

1. **Automated Resume Parsing:** Extract relevant information from resumes (e.g., personal details, skills, education, experience).
2. **Job Description Matching:** Compare resumes with job descriptions using AI algorithms.
3. **Candidate Ranking:** Rank candidates based on relevance to job requirements.
4. **Skill Gap Analysis:** Identify missing or weak skills in candidates.
5. **Bias Detection:** Mitigate biases in the screening process.
6. **User-Friendly Interface:** Provide an intuitive dashboard for recruiters to view and manage candidate rankings

1.4 Scope of the Project:

The scope of this project includes developing a system that automates resume parsing and candidate ranking. The system will be able to:

- Parse resumes in multiple formats (PDF, Word, plain text).
- Match resumes with job descriptions using NLP and ML algorithms.

- Provide ranking scores and detailed analysis.
- Detect and mitigate biases to ensure fair hiring practices.

CHAPTER 2

Literature Survey

2.1 Review relevant literature or previous work in this domain.

Resume screening is a crucial step in the recruitment process, enabling recruiters to filter and shortlist candidates efficiently. Traditionally, this process was manual, requiring HR professionals to spend extensive time reviewing resumes. However, with advancements in Artificial Intelligence (AI) and Natural Language Processing (NLP), automated resume screening has gained traction. Several studies have focused on utilizing AI-based models to enhance the accuracy and efficiency of recruitment.

A study by **Zhang et al. (2021)** explored the use of NLP and machine learning techniques to extract essential skills, experience, and qualifications from resumes, significantly reducing the time required for candidate evaluation. Similarly, **Lee & Chen (2020)** demonstrated the effectiveness of deep learning models in understanding resume context and matching candidates to job descriptions with higher precision.

Furthermore, **Patel et al. (2019)** highlighted the benefits of using AI-driven applicant tracking systems (ATS) that automate resume parsing and ranking based on predefined criteria such as experience, skills, and education. Their findings indicate that AI-based screening improves hiring accuracy and minimizes human bias.

2.2 Existing models, techniques, or methodologies :

2.2.1 Models

1. Rule-Based Models

1.1 Keyword Matching Models

- Uses predefined keywords and Boolean search to filter resumes.
- Compares job descriptions with resumes for exact word matches.
- **Example:**
    - **ATS** (Applicant Tracking Systems) used by companies to scan resumes for specific keywords.

1.2 Resume Parsing Models

- Uses NLP and regular expressions to extract structured information (name, education, skills, experience).

- **Example Tools:**
  - Rchilli Resume Parser
  - DaXtra Resume Parsing

2. Machine Learning (ML)-Based Models

2.1 Traditional ML Models

- Train models to classify resumes based on features like skills, experience, and education.
- Common ML algorithms used:
  - **Logistic Regression**: Predicts the probability of a resume matching a job.
  - **Support Vector Machine (SVM)**: Classifies resumes as "relevant" or "not relevant".
  - **Random Forest**: Uses multiple decision trees for better accuracy.
- **Example:**
  - JobMatcher (2018) used TF-IDF + SVM to rank resumes.

2.2 TF-IDF and Word Embeddings-Based Models

- Uses TF-IDF (Term Frequency-Inverse Document Frequency) to identify important words in resumes.
- Converts text into numerical vectors using:
  - **Word2Vec** – Captures word relationships.
  - **FastText** – Improves word embeddings with subword information.
  - **GloVe** – Learns word associations based on word co-occurrence.
- **Example:**
  - Resume2Vec (2019) used Word2Vec to compare resumes with job descriptions.

3. Deep Learning (DL)-Based Models

3.1 LSTM (Long Short-Term Memory) Models

- LSTM (a type of Recurrent Neural Network - **RNN**) processes text sequences.
- Used to rank resumes based on job descriptions.
- **Example:**
  - JobFitLSTM (2020) used LSTMs to extract resume features.

3.2 Transformer-Based Models (BERT, GPT, etc.)

- Uses BERT (Bidirectional Encoder Representations from Transformers**)** for resume-job matching.

- GPT models generate personalized job recommendations based on resume data.
- **Example:**
  - ResumeBERT (2021) fine-tuned BERT for resume parsing and ranking.
  - SkillMatch (2022) used BERT embeddings for job matching.

4. Hybrid Models (Combining ML + Deep Learning + Rule-Based Approaches)

- Combines the best of rule-based, ML, and deep learning techniques.
- **Example:**
  - SmartRecruit (2023): Uses TF-IDF, Word2Vec, and BERT for accurate ranking.
  - AIResumeRanker (2024): Resume Parsing + ML + Deep Learning for more robust filtering.

2.2.2 Techniques

1. Text Extraction Techniques

- **Optical Character Recognition (OCR)**: Extracts text from images or scanned resumes.
- **PyPDF2 and python-docs**: Libraries for extracting structured text from PDF and Word resumes.
- **Natural Language Processing (NLP)**: Used for parsing resume sections like education, experience, and skills.

2. Feature Extraction and Vectorization

- **TF-IDF (Term Frequency-Inverse Document Frequency)**: Weighs words based on importance in a document.
- **Word2Vec and FastText**: Captures contextual meaning of words in resumes.
- **BERT and Sentence Transformers**: Used for deeper semantic understanding.

3. Resume Ranking Techniques

- **Cosine Similarity**: Measures text similarity between resumes and job descriptions.
- **Logistic Regression, Random Forest**: Used for initial classification of relevant resumes.

- **Neural Networks and Deep Learning**: Provides higher accuracy in ranking but requires significant training data.

4. Bias Mitigation Techniques

- **Fairness-aware AI Models**: Implement bias-detection algorithms to prevent discrimination based on gender, ethnicity, or age.
- **Explainable AI (XAI)**: Provides transparency in ranking decisions

2.2.2 Methodologies

Several methodologies have been employed for AI-based resume screening, including:

1. Natural Language Processing (NLP):

- Used to extract and analyze textual information from resumes.
- Techniques like Named Entity Recognition (NER) and Part-of-Speech (POS) tagging help identify key information (e.g., names, skills, job titles).
- Tools such as Space and NLTK are commonly used for NLP tasks in resume screening.

2. Machine Learning-Based Ranking Models:

- Algorithms such as Support Vector Machines (SVM), Decision Trees, and Random Forests are applied to classify resumes based on candidate suitability.
- Li & Wang (2018) showed that ensemble models perform better in ranking resumes compared to single classifiers.

3. Deep Learning Approaches:

- Recurrent Neural Networks (RNN) and Transformer-based models like BERT are employed to understand the semantic meaning of resume content.
- Huang et al. (2022) proposed using BERT embeddings to match candidate resumes with job descriptions effectively.

4. Cosine Similarity & TF-IDF Vectorization:

- Used for matching job descriptions with resumes based on textual similarity.
- The TF-IDF (Term Frequency-Inverse Document Frequency) technique helps in feature extraction, improving the ranking process.

2.3 Gaps in Existing Solutions and How Your Project Addresses Them

1. Lack of Contextual Understanding:

- Many existing models fail to capture the nuanced meaning of job descriptions and candidate resumes. Simple keyword-based matching leads to inaccurate rankings.
- Our project addresses this by implementing BERT-based embeddings, ensuring deeper contextual understanding.

2. Bias in AI Models:

- AI models often inherit biases from training datasets, leading to discrimination in hiring.
- We aim to mitigate bias using fairness-aware algorithms and diverse training datasets.

3. Limited Support for Non-Standard Resume Formats:

- Most systems struggle with resumes in non-standard formats (e.g., image-based resumes, PDFs with complex layouts).
- Our approach integrates OCR (Optical Character Recognition) techniques to extract content from varied formats.

4. Scalability Issues:

- We employ cloud-based deployment (using Streamlit Cloud) for enhanced scalability and accessibility.

Existing Gaps	Proposed Solutions in Your Project
Bias in AI models	Implement **Fairness-aware AI** for unbiased ranking.
Limited Contextual Understanding	Use **BERT-based NLP models** for better resume-job matching.
Difficulty Handling Multiple File Formats	Support **PDF, Word, and Image-based resumes** using **OCR, PyPDF2, and python-docx**.
Lack of Explainability	Use **Explainable AI (XAI)** techniques
Scalability Issues	Optimize pipelines and deploy using **Streamlit Cloud**

Table 1 : Gaps in Existing Solutions

CHAPTER 3

Proposed Methodology

3.1 System Design

3.1.1 SystemArchitectureDiagram:

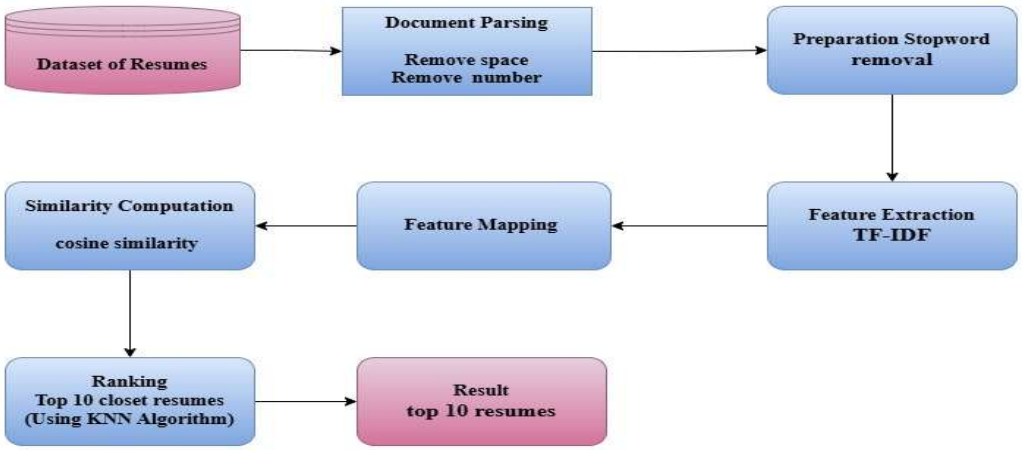

Figure 2 : SystemArchitectureDiagram

This System Architecture Diagram illustrates the end-to-end process of how resumes are screened and ranked using Natural Language Processing (NLP) and Machine Learning techniques. Below is a step-by-step breakdown of each component:

1. Dataset of Resumes

- The system starts with a dataset of resumes, which can be in various formats (PDF, DOCX, TXT, etc.).
- These resumes contain candidate details such as name, education, skills, experience, and achievements.

2. Document Parsing

- This step involves preprocessing the resume text to clean and standardize the content.
- It includes:
  - **Removing spaces** to ensure uniform formatting.
  - **Removing numbers** that may not be relevant for text-based comparisons.

3. Preparation (Stopword Removal)

- Stopwords (e.g., "the", "is", "and", "a") do not contribute much to meaning and are removed.
- This step helps in improving the accuracy of feature extraction by reducing noise in the data.

4. Feature Extraction (TF-IDF)

- The TF-IDF (Term Frequency-Inverse Document Frequency) technique is used to extract important words and phrases from resumes.
- This helps in understanding the relevance of each term in the resume concerning the job description.
- **Why TF-IDF?**
    - Assigns higher weights to words that appear frequently in a resume but are less common across all resumes.
    - Helps in matching resumes with job descriptions based on important terms.

5. Feature Mapping

- The extracted features (TF-IDF scores) are converted into a structured representation (vector format).
- This allows mathematical operations to be performed for similarity comparison.

6. Similarity Computation (Cosine Similarity)

- Cosine Similarity is a metric used to measure the similarity between the job description and each resume.
- It calculates the cosine of the angle between two text vectors (resume and job description).
- Higher cosine similarity → Better match between resume and job description.

7. Ranking (Using KNN Algorithm)

- The K-Nearest Neighbors (KNN) algorithm is applied to rank resumes based on similarity scores.
- The top 10 closest resumes are selected based on the similarity score.
- KNN finds the most relevant resumes by considering the closest neighbors (resumes) in the feature space.

8. Result (Top 10 Resumes)

- The system provides the final ranked list of top 10 resumes that best match the job description.
- These ranked resumes can be displayed to recruiters for final selection.

3.1.2 Data Flow Diagram Level Zero

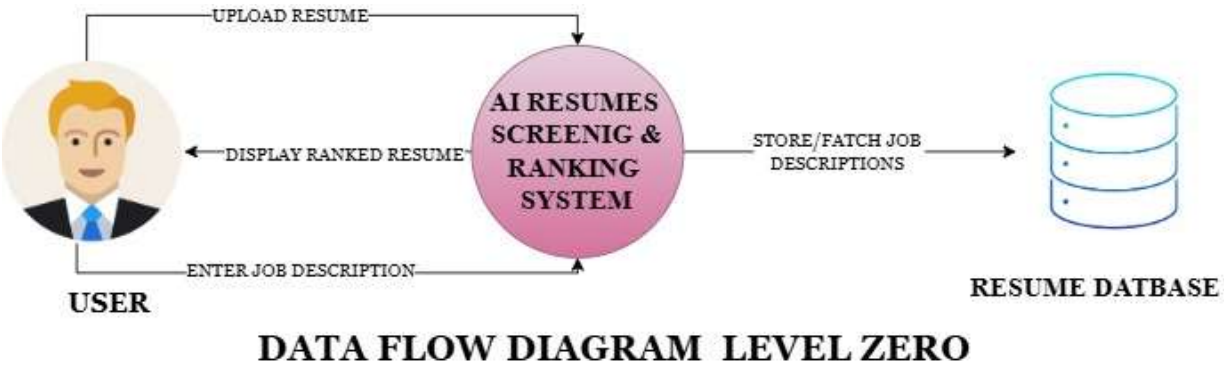

Figure 3 : Data Flow Diagram Level Zero

This Level 0 Data Flow Diagram (DFD) represents the high-level overview of how data flows through the AI Resume Screening & Ranking System. It includes the main entities, processes, and data stores involved in the system.

Entities and Data Flow

1. User (Recruiter or HR Professional)

- The user interacts with the system by providing input and receiving output.
- **Inputs:**
  - **Upload Resume:** The user uploads one or multiple resumes into the system.
  - **Enter Job Description:** The user enters the job description against which the resumes need to be screened and ranked.
- **Output:**
  - **Display Ranked Resume:** The system processes the resumes and ranks them based on relevance to the job description. The ranked list is then displayed to the user.

2. AI Resume Screening & Ranking System (Main Process)

- This is the core component of the system, responsible for processing resumes and ranking them based on their relevance to the job description.
- **Key Functions:**
  - Extract relevant information from resumes using Natural Language Processing
  - Match resumes with the job description using semantic similarity techniques (TF-IDF, BERT, Cosine Similarity, etc.).
  - Rank resumes based on factors like skills, experience, education, and job fit.
  - Send results back to the user for review.

3. Resume Database (Data Store)

- A **central repository** where job descriptions and resume data are stored and retrieved as needed.
- **Functions:**
  - Store job descriptions entered by the user.
  - Retrieve stored resumes to compare with new job descriptions.
  - Update the database when new resumes or job descriptions are added

3.1.3 Use Case Diagram

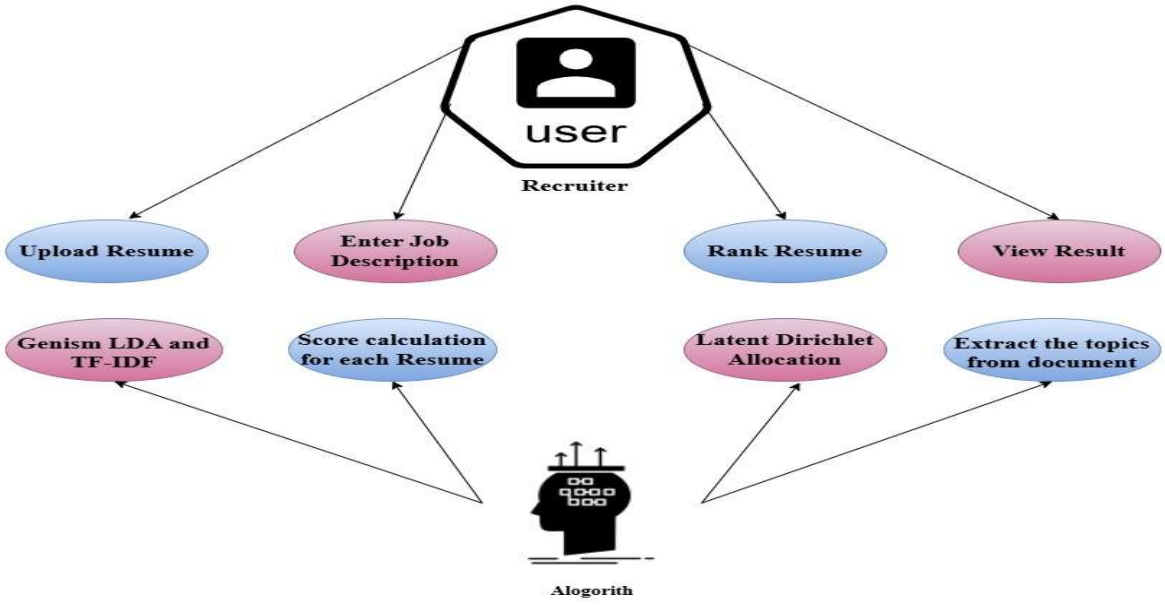

Figure 4 : Use Case Diagram

A Use Case Diagram is used to represent the interactions between users (actors) and the system. This Use Case Diagram showcases the key functionalities of the AI Resume Screening and Ranking System, detailing how recruiters interact with the system and how different processes contribute to resume ranking.

❖ Actors in the Diagram

1. User (Recruiter)

- The recruiter is the primary actor who interacts with the system.
- The recruiter performs actions such as uploading resumes, entering job descriptions, ranking resumes, and viewing results.

2. Algorithm (AI Model/Backend System)

- The algorithm is responsible for processing resumes, computing scores, and ranking them using NLP and Machine Learning techniques.

❖ Use Cases (Functionalities) and Their Descriptions

1. Upload Resume

- The recruiter uploads multiple resumes to the system.
- The system processes the resumes using Genism LDA (Latent Dirichlet Allocation) and TF-IDF for feature extraction.
- Genism LDA is used to identify hidden topics in resumes, while TF-IDF helps in extracting important words and terms.

2. **Enter Job Description**

- The recruiter provides a job description that the system will use to compare against the uploaded resumes.
- The system performs score calculations for each resume to determine its relevance to the job description.

3. **Rank Resume**

- The system ranks resumes based on similarity to the job description.
- Latent Dirichlet Allocation (LDA) is used to understand the context of resumes by identifying the dominant topics.
- The ranking is determined using ML models like Cosine Similarity, KNN, or Deep Learning models.

4. **View Result**

- The recruiter can view the ranked list of resumes based on the calculated scores.
- The system extracts and displays key topics from documents to justify ranking decisions.

❖ Key Technologies Used in the Diagram

- **Genism LDA (Latent Dirichlet Allocation):** Helps extract topics from resumes and job descriptions.
- **TF-IDF (Term Frequency-Inverse Document Frequency):** Determines the importance of words in resumes for better ranking.
- **Cosine Similarity & KNN (K-Nearest Neighbors):** Used for comparing resumes with job descriptions and ranking them accordingly.

- **Latent Dirichlet Allocation (LDA):** A topic modeling technique to improve relevance scoring.

3.1.4 Sequence Diagram

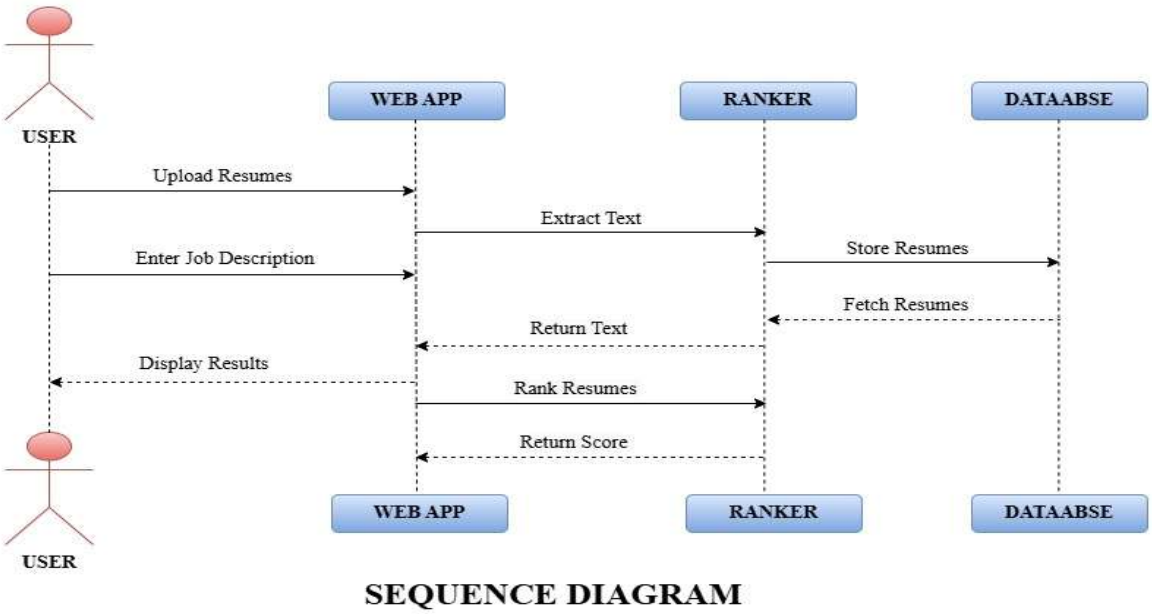

Figure 5 : Sequence Diagram

A Sequence Diagram represents the flow of interactions between different components of the system over time. It showcases the order of execution for various processes, providing a clear picture of how the system functions dynamically.

❖ **Actors and Components in the Diagram**

1. User (Recruiter):

- The recruiter interacts with the system by uploading resumes, entering job descriptions, and viewing results.

2. Web App:

- Acts as the interface where users interact with the system.
- Sends user inputs to the ranking system and retrieves the results.

3. Ranker:

- Processes resumes and job descriptions using NLP and machine learning techniques.
- Extracts text from resumes, ranks them based on relevance, and returns the ranking score.

4. Database:

- Stores resumes and job descriptions.
- Fetches resumes when needed for processing and ranking.

❖ **Step-by-Step Process in the Sequence Diagram**

1. User Uploads Resumes

- The user uploads resumes through the web application.
- The web app forwards the resumes to the Ranker for text extraction.

2. Text Extraction from Resumes

- The Ranker extracts text from the uploaded resumes.
- Extracted text is returned to the web app for processing.

3. Storing Resumes in the Database

- The Ranker stores the processed resumes in the database for future use.

4. User Enters Job Description

- The recruiter inputs a job description into the web app.
- The web app forwards the job description to the Ranker.

5. Fetching Resumes for Matching

- The Ranker fetches stored resumes from the database for comparison.

6. Resume Ranking Process

- The Ranker compares resumes with the job description using NLP and machine learning models (TF-IDF, Cosine Similarity, or BERT embeddings).
- Computes a ranking score for each resume.

7. Returning Ranked Resumes

- The Ranker returns the ranking scores to the web app.
- The web app displays the ranked resumes to the recruiter.

8. User Views Results

- The recruiter sees the ranked resumes on the web app and can proceed with hiring decisions.

3.1.5 Activity Diagram

This Activity Diagram provides a clear visual representation of the resume screening and ranking process, helping developers and stakeholders understand the system's workflow efficiently.

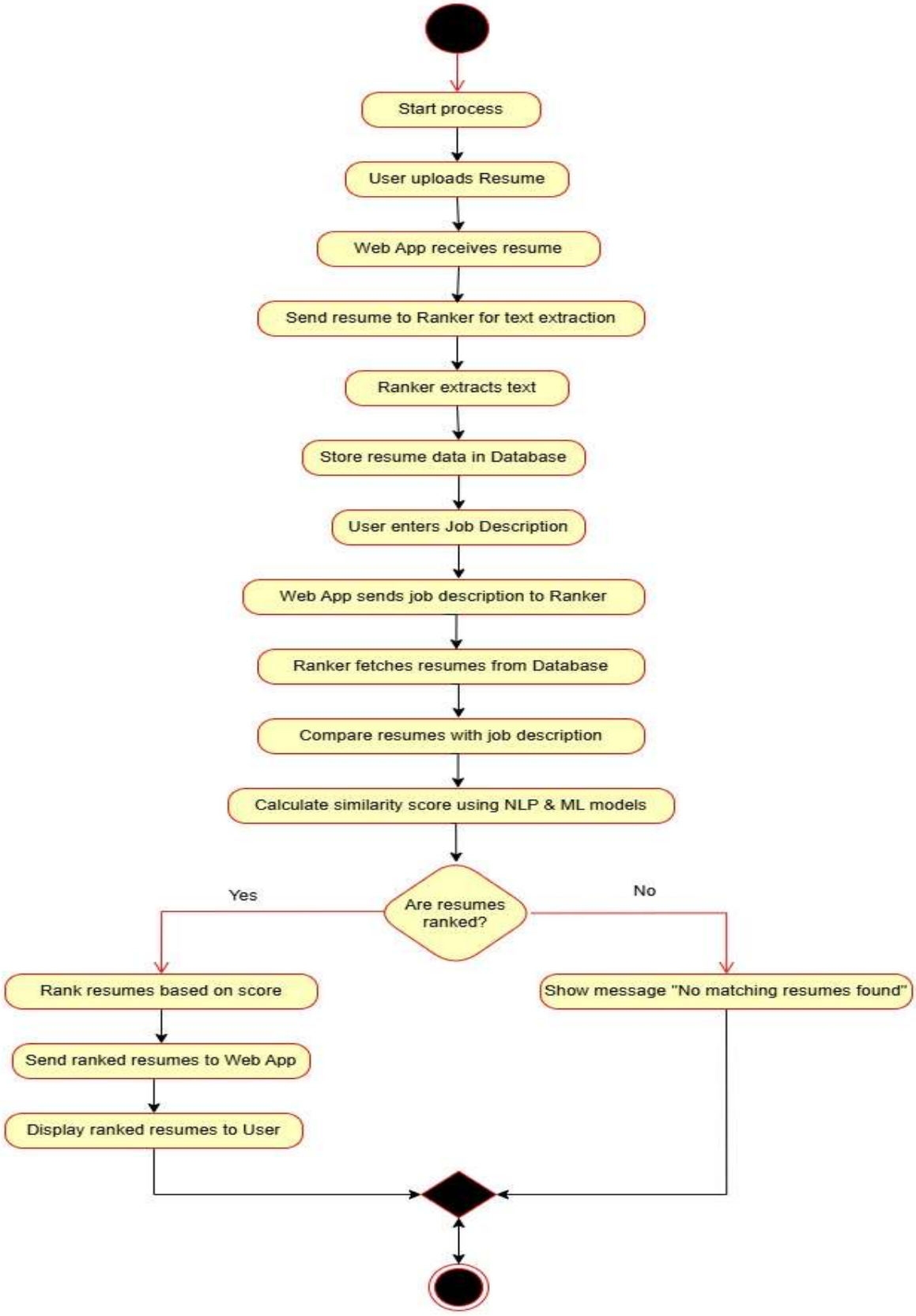

Figure 6 : Activity Diagram

1. Start Process:

- The process begins when the user initiates the system.

2. User uploads Resume:

- The user (recruiter or applicant) uploads a resume to the web application.

3. Web App receives resume:

- The web application receives the uploaded resume.

4. Send resume to Ranker for text extraction:

- The resume is sent to the ranker module, which extracts relevant text data.

5. Ranker extracts text:

- The system extracts the text from the resume using NLP techniques.

6. Store resume data in Database:

- The extracted text and relevant information from the resume are stored in the database for further processing.

7. User enters Job Description:

- The user enters a job description against which resumes will be ranked.

8. Web App sends job description to Ranker:

- The web application sends the job description to the ranking module.

9. Ranker fetches resumes from Database:

- The ranker retrieves stored resumes from the database.

10. Compare resumes with job description:

- The system compares the retrieved resumes with the given job description.

11. Calculate similarity score using NLP & ML models:

- The system calculates a similarity score using Natural Language Processing (NLP) and Machine Learning (ML) models (e.g., TF-IDF, Cosine Similarity, or LDA).

12. Decision: Are resumes ranked?

- The system checks whether resumes were successfully ranked based on the job description.

Two Possible Outcomes:

(Yes: Resumes are ranked)

13. Rank resumes based on score:

- The system ranks the resumes based on similarity scores.

14. Send ranked resumes to Web App:

- The ranked resumes are sent back to the web application.

15. Display ranked resumes to User:

- The user sees the ranked list of resumes.

(No: No Matching Resumes Found)

16. Show message "No matching resumes found":

- If no resumes match the job description, the system displays an appropriate message.

3.1.6 Class Diagram

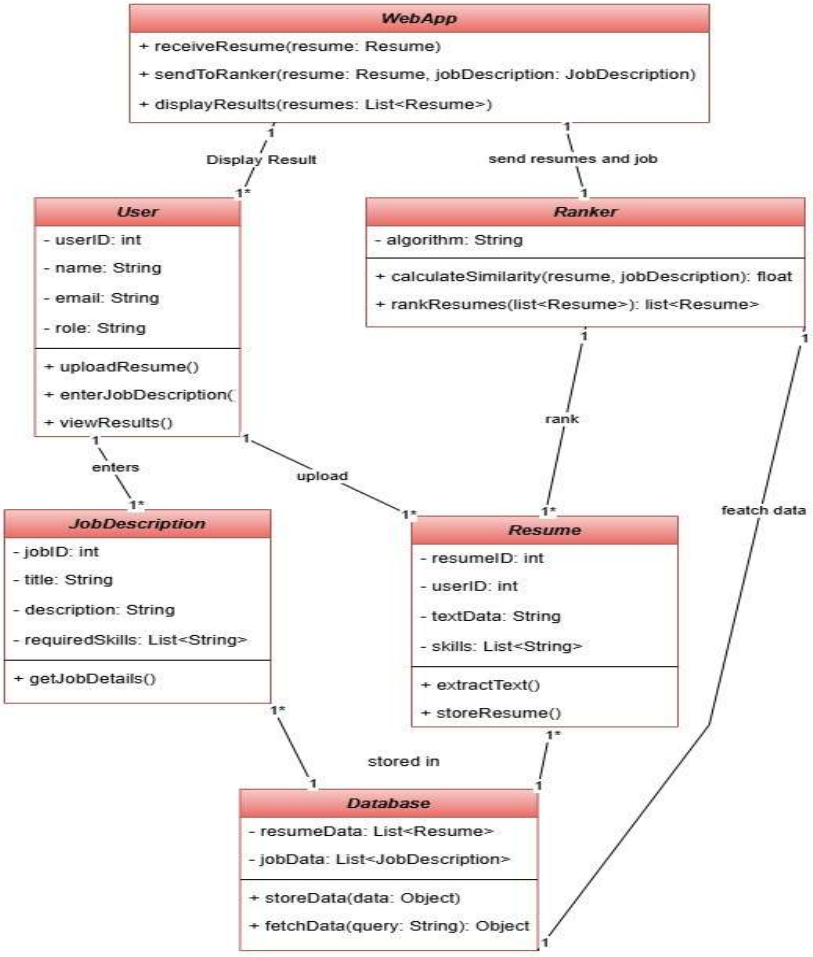

Figure 7 : Class Diagram

This Class Diagram models a Resume Screening & Ranking System, showing:

User uploads a resume & enters a job description.

WebApp sends the data to the Ranker.

Ranker processes resumes using NLP & ML, fetches from Database.

Resumes are ranked & displayed to the user.

Database stores all resume & job description data.

3.1.7 Deployment Diagram

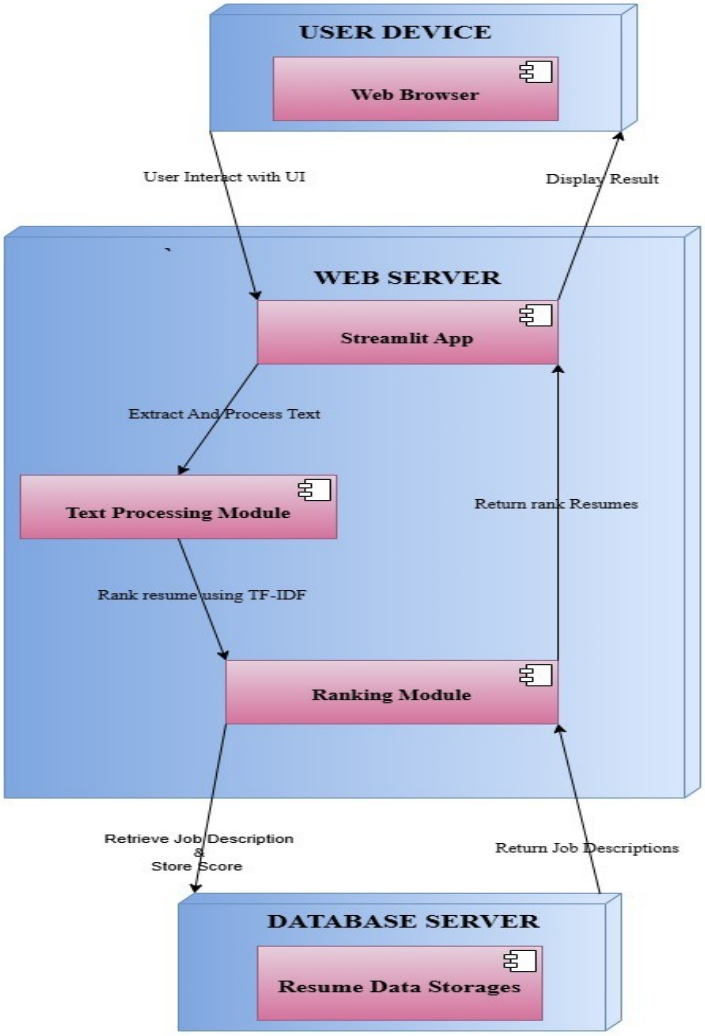

Figure 8 : Deployment Diagram

This Deployment Diagram represents the architecture of an AI-powered Resume Screening and Ranking System. It illustrates the hardware (nodes) and software (components) involved in the system, along with their interactions.

Main Components in the Diagram

The system is divided into three main layers (nodes):

1. User Device

2. Web Server

3. Database Server

Each node contains different components (software modules), and arrows indicate the interactions between them.

1. User Device (Client Side)

- **Component: Web Browser**
  - The user interacts with the system via a web browser.
  - The browser is responsible for sending user inputs (resume & job description) to the Web Server and displaying the ranked results.
- **Interactions**
  - "User Interact with UI" → The user uploads a resume & enters job details through the Streamlit web interface.
  - "Display Result" → The Web Server sends ranked resumes back to the user's browser.

2. Web Server (Processing Layer)

- **Component: Streamlit App**
  - The main web application runs on a Streamlit-based server.
  - It serves as the frontend + backend, handling user requests and interactions.
  - It communicates with different processing modules to process resumes and rank them.
- **Component: Text Processing Module**
  - Extracts and processes text from resumes.
  - May use NLP techniques such as TF-IDF, Word2Vec, or BERT to analyze resume content.
  - Sends processed resume data to the Ranking Module.
- **Component: Ranking Module**
  - Uses TF-IDF (Term Frequency-Inverse Document Frequency) to score and rank resumes.
  - Matches resume text with job description keywords and assigns a similarity score.
  - Sends the ranked resumes back to the Streamlit App for display.
- **Interactions**
  - "Extract And Process Text" → The Streamlit App sends uploaded resumes to the Text Processing Module for extraction & NLP-based processing.
  - "Rank resumes using TF-IDF" → The Text Processing Module sends the extracted text to the Ranking Module for ranking.
  - "Return ranked Resumes" → The Ranking Module sends the processed & ranked resumes back to the Streamlit App.

3. Database Server (Storage Layer)

➢ **Component: Resume Data Storages**

- Stores all resumes, job descriptions, and ranking scores.
- Can be implemented using a SQL or NoSQL database (e.g., PostgreSQL, MongoDB, Firebase).

➢ **Interactions**

- **"Retrieve Job Description"** → The Ranking Module fetches job descriptions from the database to compare with resumes.
- **"Store Score"** → The Ranking Module stores the similarity scores of resumes in the database.
- **"Return Job Descriptions"** → The database sends job descriptions back to the Web Server for ranking.

3.2 Requirement Specification

3.2.1 Hardware Requirements:

- **Processor:** Intel Core i5
- **RAM:** Minimum 16GB recommended for heavy NLP processing
- **Storage:** Minimum 50GB SSD
- **Operating System:** Windows Server
- **Cloud Hosting (Optional):** Streamlit Cloud

3.2.2 Software Requirements:

- **Operating System:** Windows 11
- **Programming Languages:** Python 3.8+ (Backend, AI Model, NLP Processing)
- **Frameworks & Libraries**
  - **Frontend : Streamlit** (For building the web interface)
  - **Backend (Processing & Ranking):**
    - **NLTK / SpaCy** (For Natural Language Processing)
    - **Scikit-learn** (For TF-IDF and similarity ranking)
    - **Pandas / NumPy** (For data handling and processing)
  - **Database & Storage: PostgreSQL**
  - **Machine Learning & AI (Optional)**
    - **TF-IDF Vectorization** (For ranking resumes)

- **Cloud & Deployment Services**
  - **GitHub** (For code version control)
  - **Streamlit Cloud** (For hosting the web application)
- **Additional Tools:** VS Code (For development)

CHAPTER 4

Implementation and Result

4.1 Snap Shots of Result:

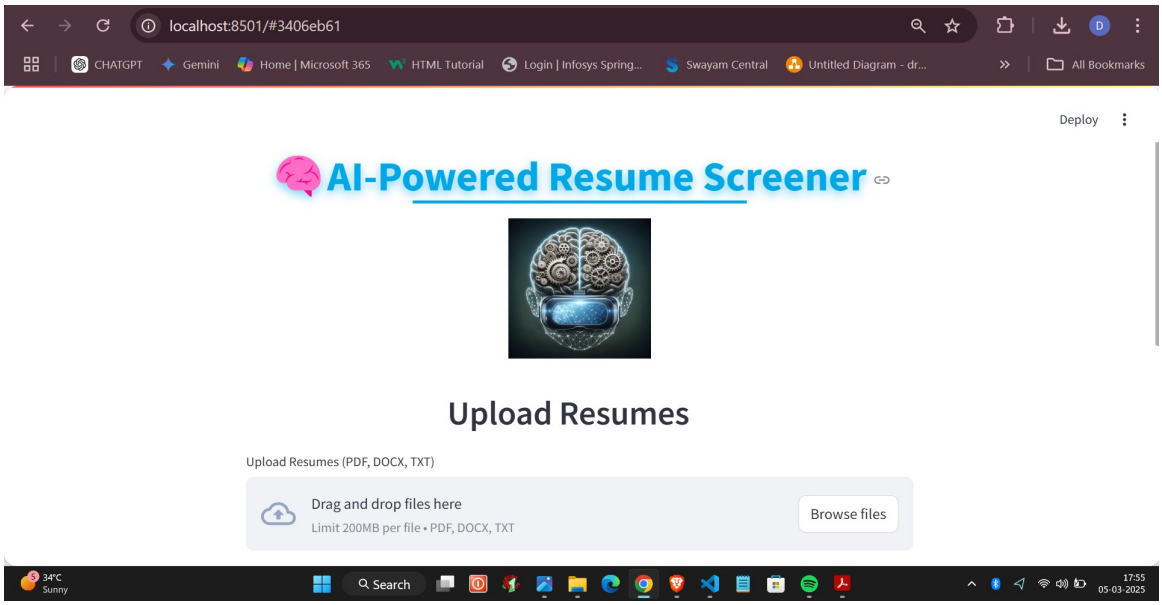

Figure 9 : Snap Shots

The snapshot represents the User Interface (UI) of an AI-Powered Resume Screener, which is a Streamlit web application running on localhost (port 8501).

1. Application Title:

- The title "AI-Powered Resume Screener" is displayed in bold and blue, indicating that the application is designed for automated resume screening.
- The brain image suggests that the application uses Artificial Intelligence (AI) for processing resumes.

2. Functionality - Resume Upload Section:

- The interface allows users to upload resumes in different formats, including PDF, DOCX, and TXT.
- It provides a drag-and-drop area and a Browse files button for selecting files manually.
- The file size limit is 200MB per file.

3. Deployment Status:

- The application is running on localhost (8501), meaning it is currently being tested locally before being deployed to a cloud service like Streamlit Cloud .

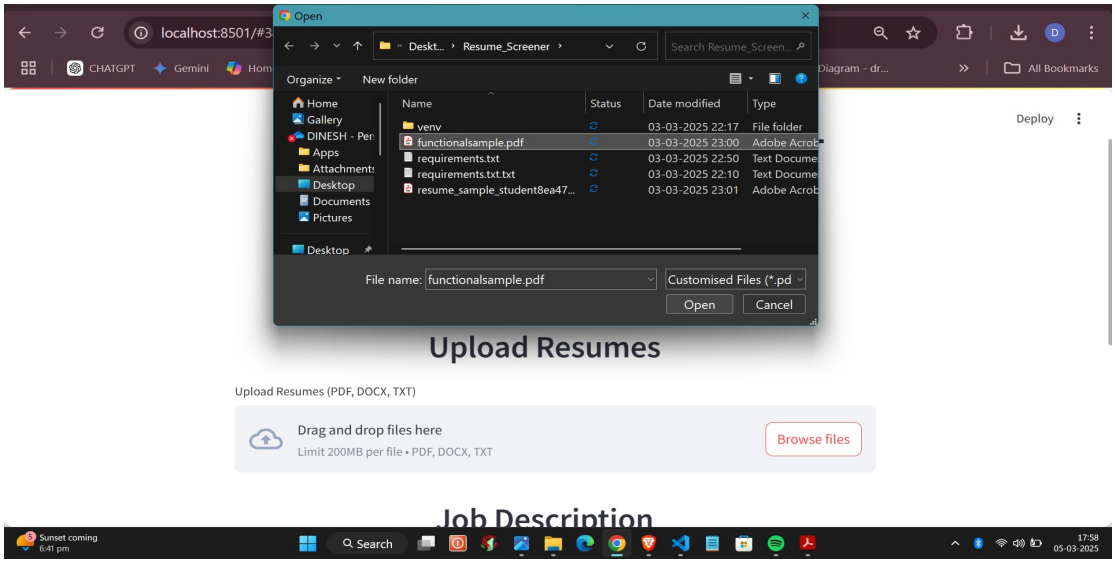

Figure 10 : Snap Shots

The snapshot represents a user interaction with the AI-Powered Resume Screener, specifically the resume upload process. The user is selecting a file to upload from their local system.

File Selection Dialog Box (Windows File Explorer)

- The user is browsing for a file inside the Resume_Screener folder on their desktop.
- The selected file is "functionalsample.pdf", a PDF file that is about to be uploaded to the application.
- The user is in the process of uploading a resume to be screened and analyzed by the application.
- Once uploaded, the application will likely extract text, skills, and relevant information from the resume using Natural Language Processing (NLP).
- The resume may then be compared against job descriptions, and the system might rank candidates based on relevance.

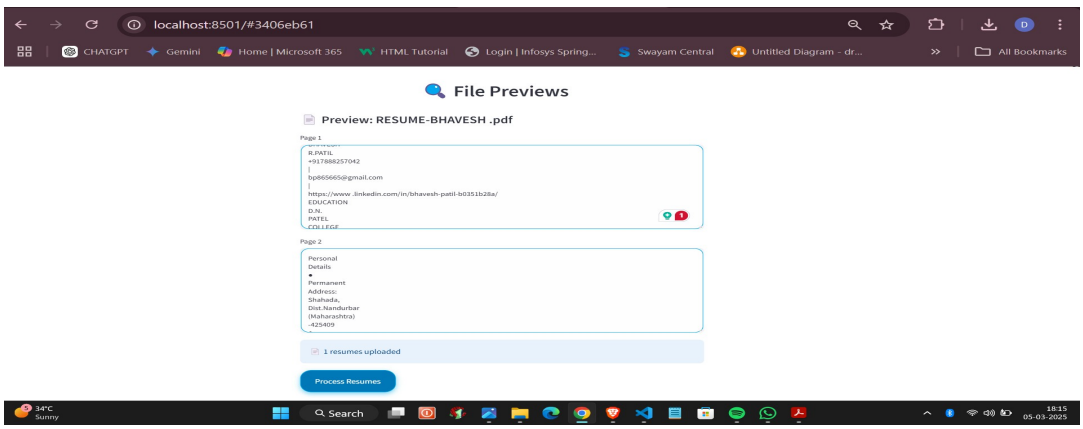

Figure 11 : Snap Shots

This snapshot represents the **"File Preview"** stage of the AI-Powered Resume Screener web application. The system has successfully extracted text from an uploaded resume (RESUME-BHAVESH.pdf) and is displaying its content before processing.

- The resume has been successfully uploaded and parsed, and the extracted text is displayed for review.
- The user can now proceed with processing the resume, which likely involves text analysis, skill matching, and ranking based on job descriptions.
- The application is functioning as expected and is at the resume processing stage before generating results.

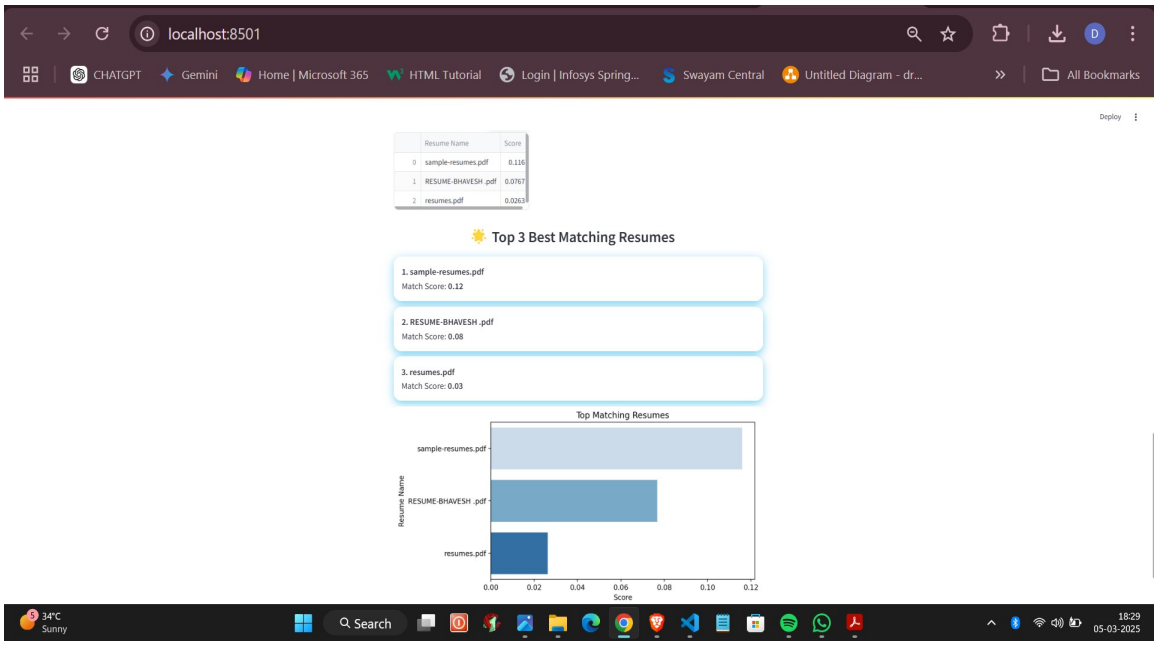

Figure 12 : Snap Shots

This snapshot represents the "Resume Ranking and Matching Results" stage of the AI-Powered Resume Screener application. It shows how different resumes have been analyzed and ranked based on their relevance to a given job description.

Top 3 Matching Resumes Section

- The application has ranked the uploaded resumes based on their similarity to the job description.
- Each resume is assigned a match score, which indicates how well the resume aligns with the job criteria.
- The top 3 resumes are displayed with their respective match scores:
    1. sample-resumes.pdf → **0.12**

2. RESUME-BHAVESH.pdf → **0.08**
3. resumes.pdf → **0.03**

· **Match Score Table (Top Left Corner)**

- · A table provides a structured view of the ranked resumes with their respective similarity scores.
- The highest-scoring resume (sample-resumes.pdf with **0.116**) is the most relevant.

· **Bar Chart Representation**

- · A **bar chart** visualizes the ranking, making it easy to compare scores.
- The x-axis represents **match scores**, and the y-axis represents **resume names**.
- **Darker bars indicate higher scores**, showing how each resume performs against the job criteria.

4.2 GitHub Link for Code:

https://github.com/sonawanewdinesh18/AI-Resume-Screening

4.3Sreamlite App Link

https://ai-resume-screening-sfdhusg7bdptmf7mwazwgz18.streamlit.app/

CHAPTER 5

Discussion and Conclusion

5.1 Future Work:

1. Fine-tune the model with real recruiter feedback.
2. Enhance resume parsing accuracy using advanced NLP models like BERT or spaCy.
3. Implement fuzzy matching techniques to improve keyword recognition.
4. Add support for more file types like CSV, JSON, and LinkedIn profiles.
5. Integrate a feedback loop for recruiters to refine ranking results.
6. Improve UI/UX with interactive filters and visual analytics.
7. Store resumes in a database for efficient candidate tracking.
8. Deploy as a scalable web app with authentication and security.
9. Expand to multilingual support for global job markets.
10. Ensure fairness and reduce bias in resume screening.
11. Optimize the ranking algorithm using machine learning models.
12. Improve real-time processing speed for handling large datasets.

5.2 Conclusion:

The AI-Powered Resume Screening and Ranking System effectively streamlines the hiring process by automating resume screening using Natural Language Processing (NLP) and machine learning. The system efficiently extracts relevant information from resumes, compares them with job descriptions, and ranks candidates based on suitability.

By implementing this solution, recruiters can save significant time and effort, ensuring a more objective and data-driven approach to candidate selection. The project demonstrates the potential of AI in recruitment, reducing manual workload and improving decision-making accuracy.

Future enhancements, such as integrating real recruiter feedback, expanding industry-specific optimizations, and improving real-time processing capabilities, can further refine the system. Overall, this project serves as a strong foundation for AI-driven recruitment solutions, making hiring more efficient and effective.

REFERENCES

[1]. https://www.drawio.com/ "*for drawing the uml diagrams".*

Jessica Simko , ***"****How Hiring Managers Make Decisions"*

[2]. http://www.careerealism.com/hiring-managers-decisions/

Vinayak Joglekar , "*Ranking Resumes using MachineLearning****"***

[3]. https://vinayakjoglekar.wordpress.com/2014/06/24/ranking-resumes-using-machine

Peter Gold , *"Artificial Intelligence Recruiting"*

[4]. https://www.linkedin.com/pulse/artificial-intelligence-recruiting-peter-gold

Turbo Ricruit ,*"Automated Application Processing","Better candidate experience","Matching Job Descriptions to Resumes"*

[5]. http://www.turborecruit.com.au/benefits-of-artificial-inteligence-for-recruitment/

[6]. https://en.wikipedia.org/wiki/R%C3%A9sum%C3%A9_parsing?utm_source=chatgpt.com

[7]. https://en.wikipedia.org/wiki/Artificial_intelligence_in_hiring?utm_source=chatgpt.com

[8]. https://en.wikipedia.org/wiki/Applications_of_artificial_intelligence?utm_source=chatgpt.com

[9]. https://ai.googleblog.com