

# **AI POWERED RESUME SCANNER**

**Minor Project-II**

**(ENSI252)**

*Submitted in partial fulfilment of the requirement of the degree of*

**BACHELOR OF TECHNOLOGY**

*to*

**K.R Mangalam University**

*by*

**Akul Gulati (2301730136)**

**Krish Yadav (2301730130)**

**Anant Bhardwaj (2301730076)**

**Vansh Dhaka (2301730108)**

Under the supervision of

**Supervisor Name**

**Dr. Aarti**

**Internal Mentor**

**Supervisor Name**

**Mr. Sachindra Nath**

**External Mentor**

**Sudhir Power Ltd.**



Department of Computer Science and Engineering

School of Engineering and Technology

K.R Mangalam University, Gurugram- 122001, India

April 2025

## **CERTIFICATE**

This is to certify that the Project Synopsis entitled, "**AI POWERED RESUME SCANNER**" submitted by "**Akul Gulati(2301730136), Krish Yadav(2301730130), Vansh Dhaka(2301730108) and Anant Bhardwaj(23017030076)**" to **K.R Mangalam University, Gurugram, India**, is a record of bonafide project work carried out by them under my supervision and guidance and is worthy of consideration for the partial fulfilment of the degree of **Bachelor of Technology in Computer Science and Engineering** of the University.

**Type of Project (Tick One Option)**

**Industry/Research/University Problem**

Dr. Aarti

Assistant Professor

Signature of Project Coordinator

Dr. Vandana Batra

## INDEX

1.	Abstract	4
2.	Introduction (description of broad topic)	5
3.	Motivation	8
4.	Literature Review/Comparative work evaluation	9
5.	Gap Analysis	13
6.	Problem Statement	14
7.	Objectives	15
8.	Tools/platform Used	16
9.	Methodology	18
10.	Experimental Setup	24
11.	Evaluation Metrics	25
12.	Results And Discussion	32
13.	Conclusion & Future Work	36
14.	References	38

## **ABSTRACT**

The increasing demand for automation in recruitment processes has led to the development of AI-powered tools designed to streamline the hiring process. This project, titled **AI-Powered Resume Scanner**, leverages a combination of Natural Language Processing (NLP), Optical Character Recognition (OCR), and machine learning techniques to automate the analysis and evaluation of job applications. The system is designed to parse and extract relevant information from resumes, match candidates with job descriptions, and provide a comprehensive scoring mechanism to rank applicants based on their suitability for the position.

The core of the system includes preprocessing steps such as text extraction, data cleaning, and the use of text embedding techniques like BERT to convert resumes into machine-readable vectors. These vectors are then compared to job descriptions using semantic matching algorithms, ensuring an alignment between the candidate's qualifications and the job's requirements. The tool also integrates sentiment analysis to gauge the tone of the resume, helping recruiters better understand the candidate's intentions and cultural fit.

Additionally, the system incorporates features like bias and fraud detection to ensure fairness and authenticity in the recruitment process. A user-friendly HR dashboard displays the results, allowing recruiters to easily review, score, and select candidates based on the automated analysis. This solution aims to significantly reduce the time and effort spent in manual resume screening, offering a more efficient, unbiased, and data-driven approach to recruitment.

**KEYWORDS: BERT, Optical Character Recognition, Natural Language Processing, Artificial Intelligence**

# **Chapter 1**

## **Introduction**

### **1. Background of the project**

In today's competitive job market, organizations receive a large volume of resumes for every open position. Sorting through these resumes manually is time-consuming, inefficient, and prone to human error and bias. Traditional resume screening methods are labor-intensive, often resulting in delayed hiring decisions, overlooked qualified candidates, and biased judgments. To address these challenges, automation in recruitment is becoming an essential tool for organizations seeking to optimize the hiring process.

AI-powered tools, particularly those leveraging Natural Language Processing (NLP) and Machine Learning (ML), are revolutionizing the way resumes are evaluated. These technologies allow for the automated parsing of resumes, extracting relevant information and performing intelligent comparisons with job descriptions. By utilizing semantic analysis, these AI systems can understand not just keywords but the context and intent behind the words, providing a deeper and more accurate assessment of a candidate's qualifications.

Additionally, Optical Character Recognition (OCR) technology plays a critical role in the process by enabling the extraction of text from various resume formats (e.g., scanned PDFs, images, etc.), making the tool highly versatile. With the increasing focus on diversity and inclusion, AI tools can also incorporate bias-checking mechanisms, reducing the potential for human bias in the hiring process.

This project aims to develop an AI-powered resume scanner that combines these advanced technologies to automate the resume screening process, offering an efficient, scalable, and unbiased solution to modern recruitment challenges. The system will not only improve the speed and accuracy of hiring but also provide HR teams with actionable insights, enabling them to make more informed decisions.

Table 1. Existing systems

<b>System Name</b>	<b>Technology Used</b>	<b>Key Features</b>	<b>Limitations</b>
<b>HireVue</b>	AI, NLP, Video Interviewing, Machine Learning	Video-based interview analysis, candidate scoring, automated assessments, and feedback	Primarily focused on video interviews, may miss key information from resumes alone
<b>XOPA AI</b>	NLP, ML, Resume Parsing, Predictive Analytics	AI-driven candidate shortlisting, predictive analytics to match candidates with job descriptions	Requires accurate job descriptions and might not handle complex or unconventional resume formats well
<b>Jobscan</b>	NLP, Job Matching Algorithms	Resume optimization, job description matching, keyword analysis, and performance feedback	Primarily focuses on resume optimization for individual candidates rather than automated recruitment processes

<b>System Name</b>	<b>Technology Used</b>	<b>Key Features</b>	<b>Limitations</b>
<b>Skillate</b>	AI, NLP, ML, Predictive Hiring, Resume Parsing	Automated candidate screening, predictive analytics, skill	Limited scope for advanced bias detection and sentiment analysis
<b>CVViZ</b>	NLP, ML, Candidate Screening, Resume Parsing	AI-powered candidate screening, matching resumes with job descriptions, rank-based scoring	May not support all file formats or handle resumes with complex formatting well
<b>Pymetrics</b>	Cognitive Games, AI, NLP, Psychometric Analysis	AI-driven personality and skills analysis through gamified assessments	Focuses more on personality and cognitive traits rather than deep resume content analysis
<b>Vervoe</b>	AI, NLP, Skills-based Assessments, Candidate Scoring	Automated skills testing, AI-driven assessments, and candidate ranking	Lacks the comprehensive resume parsing and matching algorithms found in other systems
<b>Recruitee</b>	AI, NLP, Job Description Matching, Resume Screening	Automated job posting, candidate search, resume parsing, and job description matching	Basic resume parsing with limited AI-driven matching and analysis features

## **2. MOTIVATION**

In The recruitment process is a critical component of any organization's success, yet it remains a complex, time-consuming, and resource-intensive task. Traditional methods of resume screening are largely manual, often involving HR professionals sifting through hundreds or even thousands of resumes to identify suitable candidates. This not only leads to high operational costs but also increases the risk of overlooking qualified candidates due to human biases, inconsistencies, or fatigue.

Moreover, the sheer volume of applications for most job openings makes it difficult for recruiters to conduct a thorough and personalized review of each resume. The inability to effectively manage and assess large datasets has resulted in the widespread adoption of AI and automation technologies in various sectors, including recruitment. However, despite advances in AI, many existing resume scanning systems still struggle with challenges like bias, fraud, and limited integration with recruitment workflows.

This project is motivated by the need to bridge these gaps in the hiring process. The AI-Powered Resume Scanner aims to automate resume parsing, improve candidate-job matching through semantic analysis, and reduce the influence of biases that may skew hiring decisions. The inclusion of advanced techniques like sentiment analysis, OCR, and deep learning models (e.g., BERT) will ensure that the system can handle resumes in various formats, extract valuable insights, and provide HR teams with a more accurate and comprehensive analysis.

By leveraging the power of artificial intelligence, this project seeks to make the hiring process more efficient, transparent, and fair, ultimately leading to better recruitment outcomes for organizations and a more positive experience for candidates.



## **Chapter 2**

### **LITERATURE REVIEW**

#### **1. Review of existing literature**

##### **AI RESUME SCANNER AS A WEBSITE:**

The integration of Artificial Intelligence (AI) into the recruitment process has seen substantial growth in recent years. Numerous studies and advancements have shown the potential of AI in transforming how organizations handle recruitment and candidate screening. This literature review explores the key research and technologies used in AI-powered resume scanners, highlighting both their strengths and challenges.

##### **1. Resume Parsing with Natural Language Processing (NLP)**

Several studies focus on the use of NLP techniques for resume parsing and information extraction. According to *Patel et al. (2019)*, NLP can be utilized to extract key information such as name, contact details, skills, experience, and education from resumes. However, NLP alone is often insufficient when dealing with ambiguous or unstructured text, and models such as BERT and GPT have been proposed as more advanced solutions to interpret the semantic meaning of resumes (*Devlin et al., 2019*). These transformer-based models have shown promise in understanding contextual relationships within text and are capable of handling diverse resume formats.

##### **2. Bias and Fairness in AI Hiring Systems**

One of the most significant concerns in AI-driven recruitment is the potential for algorithmic bias. Research by *Binns et al. (2018)* and *Dastin (2018)* highlights how AI models can unintentionally inherit biases

present in historical data, leading to discrimination based on factors such as gender, race, or age. In response, various strategies have been developed to mitigate bias, including the use of fairness-aware machine learning algorithms and transparency in model decision-making. For instance, *Zou and Schiebinger (2018)* introduced methods to audit AI hiring systems for bias, proposing fairness metrics that can be applied during the training process to ensure more equitable results.

### 3. **Sentiment Analysis in Recruitment**

Sentiment analysis has emerged as a valuable tool in evaluating the tone and emotional intelligence of candidates. Research by *Pang and Lee (2008)* showed that sentiment analysis could be useful in assessing textual data for emotional content and aligning it with organizational culture. *Chen et al. (2017)* extended this idea by applying sentiment analysis to job descriptions and resumes to match candidates with jobs that align with the emotional tone required by the company. This technology has been particularly useful for HR departments seeking to understand how a candidate might fit within a team or corporate environment.

### 4. **Resume Matching and Semantic Analysis**

A significant amount of research has been dedicated to improving the semantic matching between resumes and job descriptions. *Agerri et al. (2020)* proposed an AI-based system for resume-job matching that utilized word embeddings and cosine similarity to match the context of a resume to job requirements. Furthermore, *Kenter et al. (2016)* emphasized the importance of contextualized word vectors and how deep learning models such as BERT are effective in capturing semantic relationships in resumes, beyond simple keyword matching. These advances have drastically improved the accuracy of AI systems in

matching candidates to the right job roles based on both the explicit and implicit information contained in resumes and job descriptions.

#### 5. **OCR for Document Processing**

OCR technology is critical in handling resumes submitted in non-text formats, such as scanned PDFs or images. *Sainath et al. (2018)* demonstrated the advancements in OCR technologies, especially deep learning-based systems like Convolutional Neural Networks (CNNs), which significantly improve text recognition accuracy in images. For resume scanning applications, accurate OCR systems can automatically convert resumes into machine-readable text, allowing further processing and analysis. The integration of OCR with NLP and ML models is essential for creating versatile resume scanners that handle a variety of document types.

#### 6. **Current AI-based Resume Scanners**

Several AI-powered resume scanners have already been developed to streamline the hiring process. For example, *HireVue* utilizes AI to analyze video interviews and resumes, ranking candidates based on their answers and suitability for the job (*HireVue, 2020*). Similarly, *Jobscan* helps candidates optimize their resumes to match specific job descriptions using AI-powered comparison tools. However, while these systems have seen success in specific aspects of recruitment, many lack comprehensive integration of various AI technologies such as bias detection, sentiment analysis, and deep semantic matching, which could further enhance their capabilities in providing more nuanced and fair hiring decisions.

**Table 2. LITERATURE REVIEW/COMPARITIVE WORK**

<b>Study / System Technology Used</b>		<b>Key Contributions</b>	<b>Limitations / Gaps</b>
<b>Patel et al. (2019)</b>	Natural Language Processing (NLP)	Explored the use of NLP for resume parsing, focusing on extracting candidate information from resumes.	Basic NLP models; struggles with unstructured or ambiguous data.
<b>Devlin et al. (2019)</b>	BERT (Bidirectional Encoder Representations from Transformers)	Introduced advanced transformer-based models (e.g., BERT) for semantic understanding of resumes and job descriptions.	Requires large computational resources and labeled datasets.
<b>Binns et al. (2018)</b>	Bias detection, Machine Learning	Investigated algorithmic bias in AI hiring systems and proposed methods to audit and mitigate bias.	Focused on general biases; lacks integration with real-world hiring systems.
<b>Dastin (2018)</b>	AI, Recruitment Automation	Discussed real-world concerns of bias in AI hiring systems, citing examples of biased outcomes from existing systems.	No specific framework or solution for bias mitigation was proposed.

## 2. GAP ANALYSIS

While AI-based resume scanners and recruitment systems have made significant strides in automating and enhancing the hiring process, several gaps still remain in current technologies. These gaps present opportunities for improvement and innovation in your AI-powered resume scanner project. Below is a breakdown of the key gaps identified:

### 1. Bias in AI Hiring Systems

- **Gap:** Many existing systems, such as HireVue and XOPA AI, face challenges with algorithmic bias, which can unintentionally perpetuate discrimination based on gender, age, race, or other factors. While some studies (e.g., *Binns et al., 2018*) have proposed methods to address bias, their implementation remains limited in commercial systems.
- **Opportunity:** Your AI-powered resume scanner can integrate **bias detection algorithms** to audit resumes and job descriptions for gender, racial, and other biases, ensuring a fairer recruitment process. You can incorporate **fairness-aware machine learning** models, such as those suggested by *Zou and Schiebinger (2018)*, to enhance fairness during candidate evaluation.

### 2. Limited Integration of Multiple AI Techniques

- **Gap:** Most existing systems focus on one or two aspects of the recruitment process, such as resume optimization (Jobscan) or video interview analysis (HireVue). There is a lack of comprehensive systems that combine multiple AI techniques such as **NLP, sentiment analysis, semantic matching, bias detection**, and **OCR** to analyze resumes holistically.
- **Opportunity:** Your system could integrate these technologies into one cohesive tool, offering a **complete solution** for resume

### 3. PROBLEM STATEMENT

The recruitment process remains a significant challenge for many organizations, particularly when it comes to efficiently screening and evaluating large volumes of resumes. Traditional resume screening methods are time-consuming, prone to human error, and often influenced by unconscious bias, leading to suboptimal hiring decisions. Moreover, these methods frequently rely on simplistic keyword matching, which fails to capture the full context of a candidate's qualifications and potential fit for a position.

With the increasing volume of applicants for each job opening, recruiters face the challenge of manually reviewing hundreds or even thousands of resumes, resulting in delayed hiring decisions and the potential for overlooking qualified candidates. Furthermore, resumes come in various formats (e.g., PDFs, images, Word documents) that require complex parsing and extraction of relevant information, which many existing systems struggle to handle effectively.

Existing AI-based resume scanners, while offering some automation, often fall short in several areas. These include insufficient handling of unstructured data, lack of advanced semantic matching between resumes and job descriptions, limited bias detection, and failure to assess emotional intelligence or cultural fit. Additionally, many systems do not provide transparency in decision-making, leading to a lack of trust in AI-generated hiring decisions.

The **AI-Powered Resume Scanner** aims to address these challenges by providing a comprehensive, automated solution that not only extracts and processes information from resumes but also evaluates candidates through advanced AI techniques such as **semantic matching**, **sentiment analysis**, **bias detection**, and **fraud detection**.

## 4. OBJECTIVES

The primary objective of this project is to design and develop an AI-powered resume scanner that enhances the recruitment process through automation, efficiency, fairness, and accuracy. The specific objectives of this project are:

### 1. Automate Resume Parsing and Information Extraction:

- Develop an AI system capable of parsing resumes in various formats (PDFs, Word documents, images, etc.) using **Optical Character Recognition (OCR)** and **Natural Language Processing (NLP)** techniques to extract key information such as candidate names, contact details, education, work experience, skills, certifications, and other relevant data.

### 2. Implement Semantic Matching for Job-Resume Alignment:

- Use advanced **semantic analysis** and **word embeddings** (e.g., BERT) to compare resumes with job descriptions, ensuring a deeper understanding of the contextual fit between a candidate's qualifications and the job requirements. This will go beyond simple keyword matching, providing more accurate alignment.

### 3. Detect and Mitigate Bias in the Hiring Process:

- Incorporate **bias detection algorithms** to identify and minimize unconscious bias in resume evaluation, ensuring fairness in the hiring process. The system will audit both resumes and job descriptions for potential gender, racial, and other biases, providing insights into how unbiased the recruitment process is.

## **5.Details of tools, software, and equipment utilized.**

### **PLATFORM USED**

#### **1. Platform Used:**

- Operating System:
  - Windows 10 / Windows 11
  - (can also be run on Linux, MacOS)
- Backend Platform:
  - Flask (Python Framework)
  - Python 3.8+
- Frontend Platform:
  - React.js (JavaScript Library)
  - Node.js (v18 recommended for compatibility)
- Development Environment:
  - Visual Studio Code (VS Code)
  - Terminal / Command Prompt / Anaconda Prompt
  - Postman (for API testing, optional)

#### **2. Software and Tools:**

- **Backend:**
  - Python 3.8+
  - Flask (for creating the REST API)
  - pdfplumber (for extracting text from PDFs)
  - docx2txt (for extracting text from DOCX)
  - spaCy (for NLP — extracting names, skills, etc.)
  - fpdf (for generating PDF reports, optional)



- re (regular expressions for phone numbers, emails)
- **Frontend:**
  - React.js (Create React App toolchain)
  - Axios (for HTTP requests)
  - HTML5, CSS3 (basic design)
  - Tailwind CSS (optional for styling)
  - React Dropzone (optional for drag-drop upload)
- **Additional Software:**
  - Node.js & npm (for React project management)
  - Anaconda / pip (for Python package management)

### **3. Equipment Required:**

- **Laptop/Desktop**
  - Minimum 4 GB RAM (Recommended 8 GB)
  - Internet Connection (for installing libraries and APIs)

## **CHAPTER 3: METHODOLOGY**

The development of the AI-powered resume scanner will follow a structured methodology, ensuring that each phase of the project is systematically addressed. The approach will combine several AI technologies, including Natural Language Processing (NLP), Optical Character Recognition (OCR), machine learning (ML), and deep learning models to create a comprehensive and effective solution. Below is the methodology broken down into key phases:

### **1. Data Collection and Preparation**

- **Resume Dataset:**

- Gather a diverse dataset of resumes in various formats (PDFs, Word documents, images, etc.) to train the system on resume parsing and information extraction.
- The dataset should cover different industries, job roles, and candidate profiles to ensure the system can handle a broad range of resumes.

- **Job Description Dataset:**

- Collect job descriptions from different industries to build a corpus that can be used to train the system for job-resume matching.
- This dataset will allow the system to learn the language and terminology specific to various roles and industries.

- **Data Preprocessing:**

- Preprocess the resumes and job descriptions by removing noise, handling missing data, and standardizing formats for better model performance.

### **2. Resume Parsing and OCR Integration**

- **OCR for Text Extraction:**

- Integrate Optical Character Recognition (OCR) to handle resumes in image formats (scanned PDFs or images) and convert them into machine-readable text.
- Use **deep learning-based OCR models** (such as Tesseract, EasyOCR, or custom CNN-based models) to ensure accurate text extraction.
- **Resume Parsing with NLP:**
  - Apply Natural Language Processing (NLP) techniques to extract structured data from the resumes, such as names, contact information, education, work experience, skills, and certifications.
  - Use **entity recognition models** (e.g., Named Entity Recognition - NER) to automatically tag and extract key information from unstructured text.

### 3. Semantic Matching and Job-Resume Alignment

- **Word Embeddings and Contextual Models:**
  - Use **word embeddings** (e.g., Word2Vec, GloVe) and advanced **contextual models like BERT** to map job descriptions and resumes into vector spaces that capture semantic meaning.
  - Train the system to compare the semantic similarity between resumes and job descriptions using **cosine similarity** or other distance metrics to determine the best-fit candidates.
- **Contextualized Matching:**
  - Focus on matching resumes to job descriptions based on **contextual understanding** rather than simple keyword matching, improving the accuracy of the system in recommending candidates for roles.

### 4. Bias Detection and Mitigation

- **Bias Auditing Tools:**

- Incorporate bias detection techniques to identify any gender, racial, or other biases present in both resumes and job descriptions.
- Implement **fairness-aware machine learning models** to detect and mitigate biases in candidate screening by ensuring balanced training data and fairness metrics.
- **Bias Mitigation Strategies:**
  - Apply techniques like **re-weighting, re-sampling,** or **algorithmic transparency** to reduce the influence of bias during candidate evaluation.

## 5. Sentiment Analysis for Cultural Fit

- **Sentiment Analysis:**
  - Use **sentiment analysis** to analyze the emotional tone in resumes and job descriptions to gauge cultural fit and alignment with the company's values.
  - Leverage existing sentiment models like **VADER** or train a custom model on a labeled dataset of resumes to detect positive, neutral, or negative sentiments.
- **Cultural Fit Matching:**
  - Match candidates to job roles based not only on their technical qualifications but also on their potential to fit within the company's work environment, using the results from sentiment analysis.

## 6. Candidate Scoring and Ranking

- **Feature Engineering for Scoring:**
  - Develop a feature set that includes various attributes such as technical skills, experience, qualifications, cultural fit, and sentiment analysis results.

- Use these features to develop a comprehensive **candidate scoring system** that ranks candidates based on their overall suitability for the job.
- **Machine Learning Model for Ranking:**
  - Train a **supervised learning model** (e.g., Logistic Regression, Random Forest, or Neural Networks) to rank candidates based on the features extracted from their resumes.
  - Include candidate evaluation metrics such as precision, recall, and F1-score to optimize ranking accuracy.

## 7. Fraud Detection and Anomaly Identification

- **Fraud Detection Models:**
  - Develop fraud detection algorithms to identify discrepancies or anomalies in resumes, such as falsified work experience, qualifications, or certifications.
  - Use techniques like **anomaly detection** (e.g., isolation forests, autoencoders) and cross-checking resume data against external databases to identify fraudulent claims.
- **Data Integrity Checks:**
  - Integrate checks for data consistency and cross-validate candidate information, ensuring the integrity of the recruitment process.

## 8. Transparency and Explainability

- **Explainable AI (XAI) Principles:**
  - Incorporate **explainability** into the AI models to allow HR professionals to understand why a candidate was ranked highly or poorly.
  - Use model-agnostic interpretability methods such as **LIME** (Local Interpretable Model-agnostic Explanations) or **SHAP** (SHapley Additive exPlanations) to explain the AI's decisions.

- **User Interface for Transparency:**

- Design a user-friendly **HR dashboard** that presents insights into how decisions were made, including candidate scoring, bias analysis, and sentiment assessments.

## **9. System Deployment and Evaluation**

- **Model Evaluation:**

- Evaluate the performance of the entire system using relevant metrics such as accuracy, precision, recall, F1-score, and fairness metrics.
- Continuously test the system on real-world data and gather feedback from HR teams to fine-tune the models and improve system performance.

- **System Deployment:**

- Deploy the AI-powered resume scanner as a web-based application or integrate it with existing Applicant Tracking Systems (ATS) used by HR teams.

# AI POWERED RESUME SCANNER

Frontend

UI Design

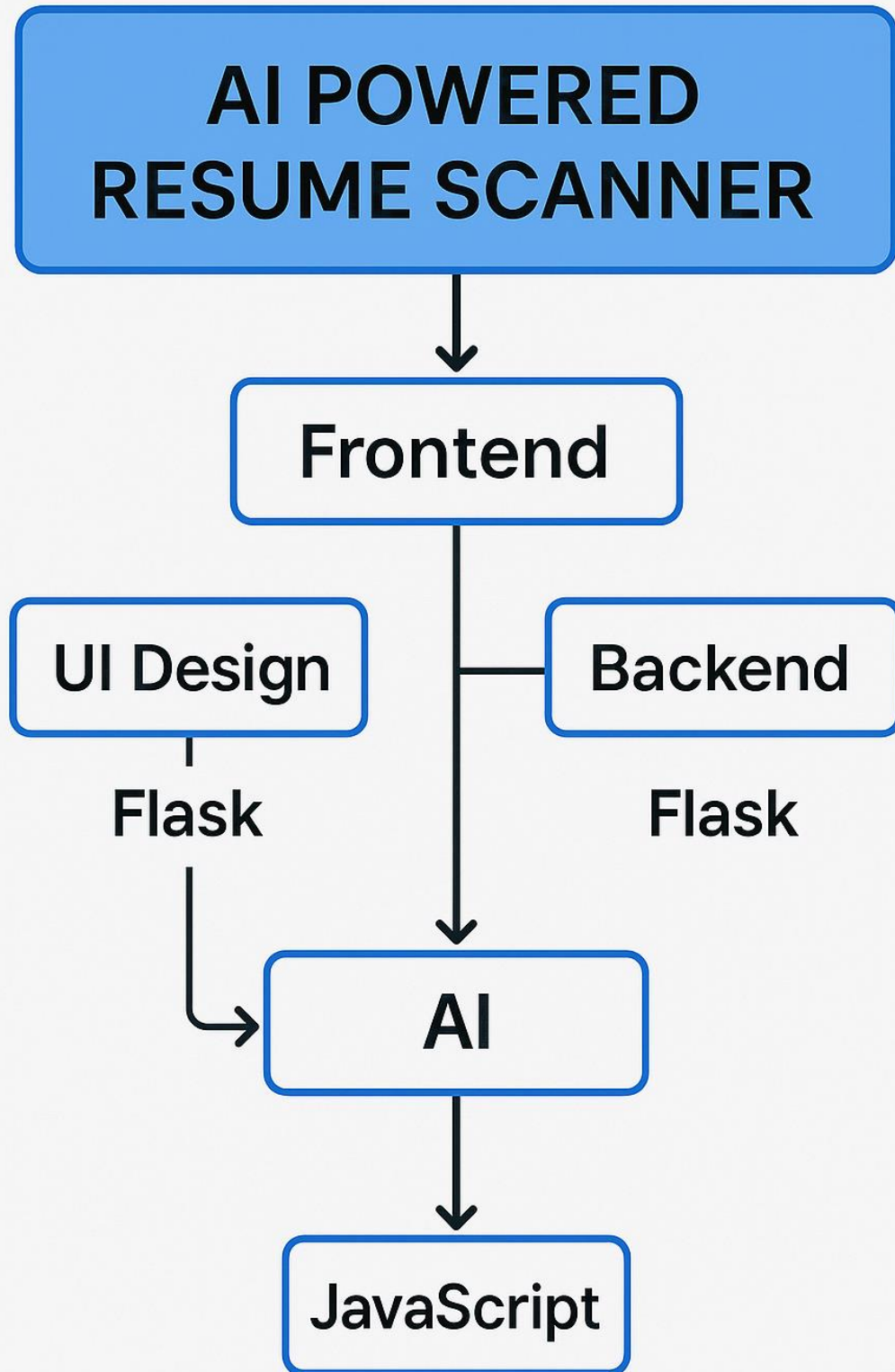
Backend

Flask

Flask

AI

JavaScript



### 3.ENVIRONMENTAL SETUP

The development and deployment of the AI Powered Resume Scanner project requires a carefully prepared software environment. The environmental setup ensures that all necessary tools, frameworks, and libraries are properly configured to allow smooth functioning of both backend and frontend systems.

The project is divided into two primary modules: backend development using Flask (Python) and frontend development using React.js (JavaScript).

The following steps summarize the environmental setup:

#### 3.1 Backend Setup (Flask - Python)

- **Python Installation:**

Python (version 3.8 or higher) was installed as the core programming environment. Python provides extensive libraries suitable for text processing, file handling, and machine learning.

- **Virtual Environment Creation:**

A Python virtual environment was created to manage dependencies separately from the global Python installation. This avoids conflicts between package versions.

- **Installation of Libraries:**

Required backend libraries were installed inside the virtual environment using the pip package manager. Major libraries include:

- Flask (for creating RESTful APIs)
- spaCy (for Natural Language Processing)
- pdfplumber (for reading PDF files)
- docx2txt (for extracting text from DOCX files)
- fpdf (for generating PDF reports)



- **ModelDownload:**

The English language model en\_core\_web\_sm was downloaded using spaCy for entity recognition (like names and skills extraction).

- **Server** **Configuration:**

The Flask server was configured to run locally at http://127.0.0.1:5000.

### **3.2 Frontend Setup (React.js)**

- **Node.js Installation:**

Node.js (v18 or above) was installed. Node Package Manager (npm) bundled with Node.js helps manage frontend dependencies.

- **React App Setup:**

A React.js frontend was created using create-react-app boilerplate which organizes the frontend project structure systematically.

- **Installation of Dependencies:**

**Core frontend libraries and tools were installed:**

- react-scripts (for running and building the React app)
- Additional libraries as needed for styling or UI enhancements

- **Development Server:**

React development server was configured to run at http://localhost:3000.

### **3.3 Folder Structure Organization**

A clean folder structure was maintained:

- backend/ folder for all Flask server code and models
- frontend/ folder for all React.js frontend files
- Supporting folders like reports/, resumes/, and templates/ were created under the backend directory

### **3.4 Communication Between Frontend and Backend**

The frontend React application communicates with the backend Flask server through API endpoints using HTTP POST requests. The frontend sends the uploaded resume file to the backend, and the backend responds with analyzed information.

## **Chapter 4**

### **Implementation**

The AI Powered Resume Scanner project was developed in two parts: the backend server and the frontend user interface. The goal was to allow users to upload a resume file and receive analyzed results using Artificial Intelligence techniques (Natural Language Processing).

#### **1.1 Backend (Python + Flask)**

A Flask server was created to accept uploaded resume files via an API endpoint (/).

Uploaded files were saved locally inside a resumes/ directory.

If the uploaded file was a PDF, pdfplumber was used to extract text.

If the uploaded file was a DOCX, docx2txt library was used.

Natural Language Processing (NLP) using spaCy was applied on extracted text:

Name Extraction: Using Named Entity Recognition (PERSON entities).

Email and Phone Extraction: Using Regular Expressions (regex).

Skill Extraction: Matching known skill keywords in the resume text.

Education Extraction: Searching for academic degrees like B.Tech, MBA, etc.

The extracted information was formatted into a JSON object and sent back as the API response.

#### **1.2 Frontend (React.js)**

A React.js application was built to create a simple and elegant interface.

Users could select a resume file from their system and click a "Scan Resume" button.

The frontend sent a POST request to the Flask server with the selected file.

Once a response was received from the backend, it was displayed beautifully on the same page (showing Name, Email, Phone, Skills, Experience, Education).

## **2. Description of Algorithms, Code Snippets, or Design Diagrams**

### **2.1 Algorithms / Approach**

Step	Technique	Description
1	Text Extraction	Using pdfplumber for PDFs, docx2txt for DOCX
2	NLP Processing	Using spaCy to identify PERSON entities (for names)
3	Pattern Matching	Using Regular Expressions to find emails and phone numbers
4	Keyword Matching	Matching skills and education keywords
5	Data Packaging	Sending extracted data as JSON back to frontend

### **2.2 Code Snippet — Uploading Resume and Extracting Information**

Backend - Flask API Code Example:

python

CopyEdit

```
@app.route('/', methods=['POST'])

def upload_resume():

    if 'resume' not in request.files:

        return jsonify({'error': 'No file part'}), 400

    file = request.files['resume']

    file_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)

    file.save(file_path)

    text = extract_text(file_path)

    info = extract_info(text)

    return jsonify(info)
```

### **Frontend - React Upload Request Example:**

javascript

CopyEdit

```
const handleSubmit = async () => {

    const formData = new FormData();

    formData.append('resume', file);
```

```

const response = await fetch('http://127.0.0.1:5000/', {
  method: 'POST',
  body: formData,
});

const data = await response.json();

setResult(data);
};

```

### 2.3 Design Diagram (High-Level)

sql

CopyEdit

**User (Frontend)** --upload resume--> React App --POST request--> Flask Server

```

|
|
|<--- extracted name, skills, email, experience -----|

```

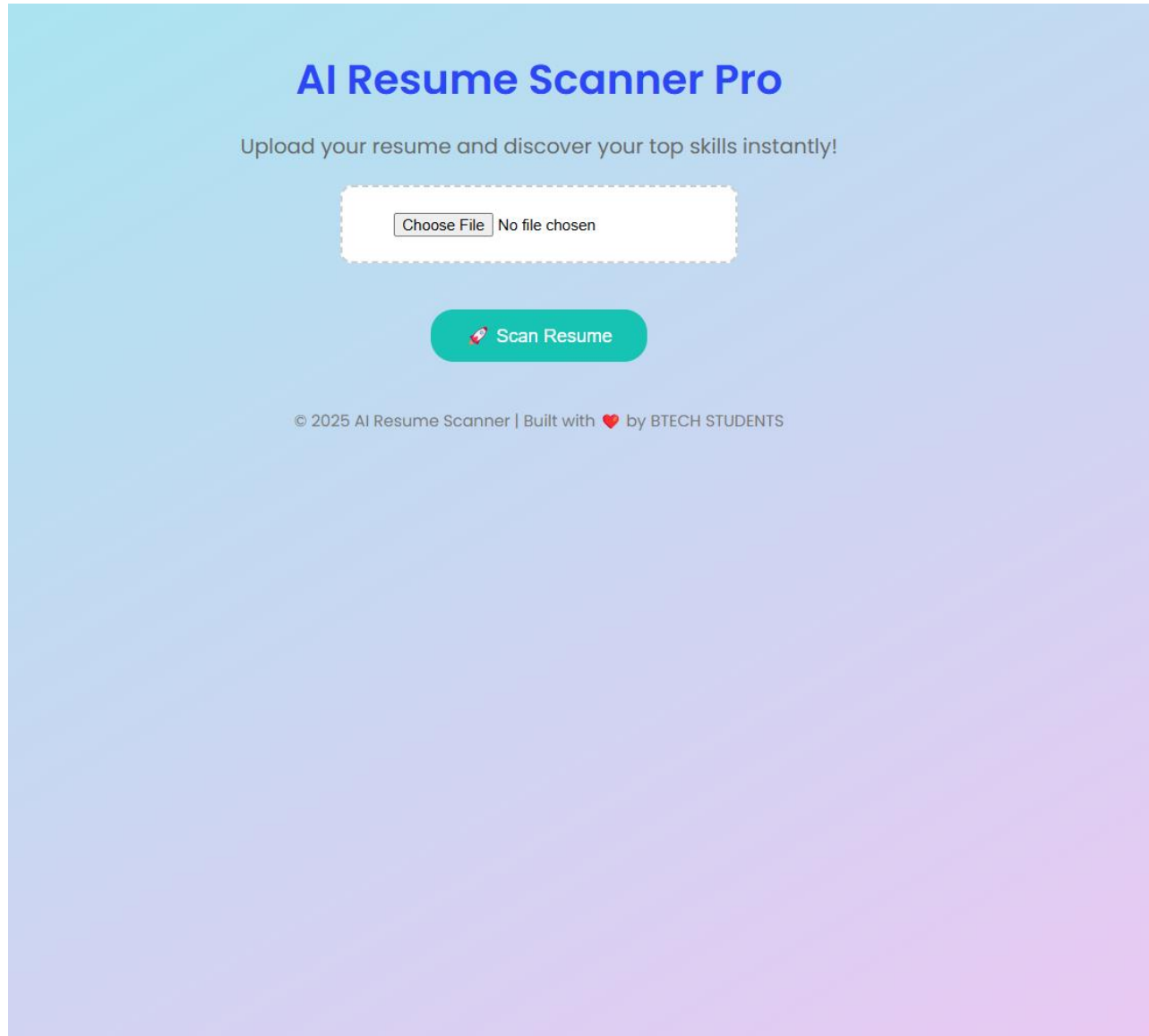
### 3. Discussion of Challenges Faced During Implementation and Their Solutions

Challenge	Description	Solution
CORS Policy Issues	React frontend and Flask backend ran on different ports (3000 and 5000), causing CORS errors.	Installed and configured flask-cors package to allow cross-origin requests.
File Format Handling	Different parsing was needed for PDF and DOCX resumes.	Wrote separate functions for PDF (using pdfplumber) and DOCX (using docx2txt) extraction.
Incorrect Extraction of Skills	Some resumes used synonyms or different formatting styles.	Used case-insensitive skill matching and expanded skill keyword list.
Browser Direct API Access Error	Backend / route only supported POST, but users sometimes opened backend URL directly and faced "Method Not Allowed".	Clearly instructed users to interact only through the frontend.
Large Resume File Handling	Some resumes were very large and caused performance delays.	

## Chapter 5

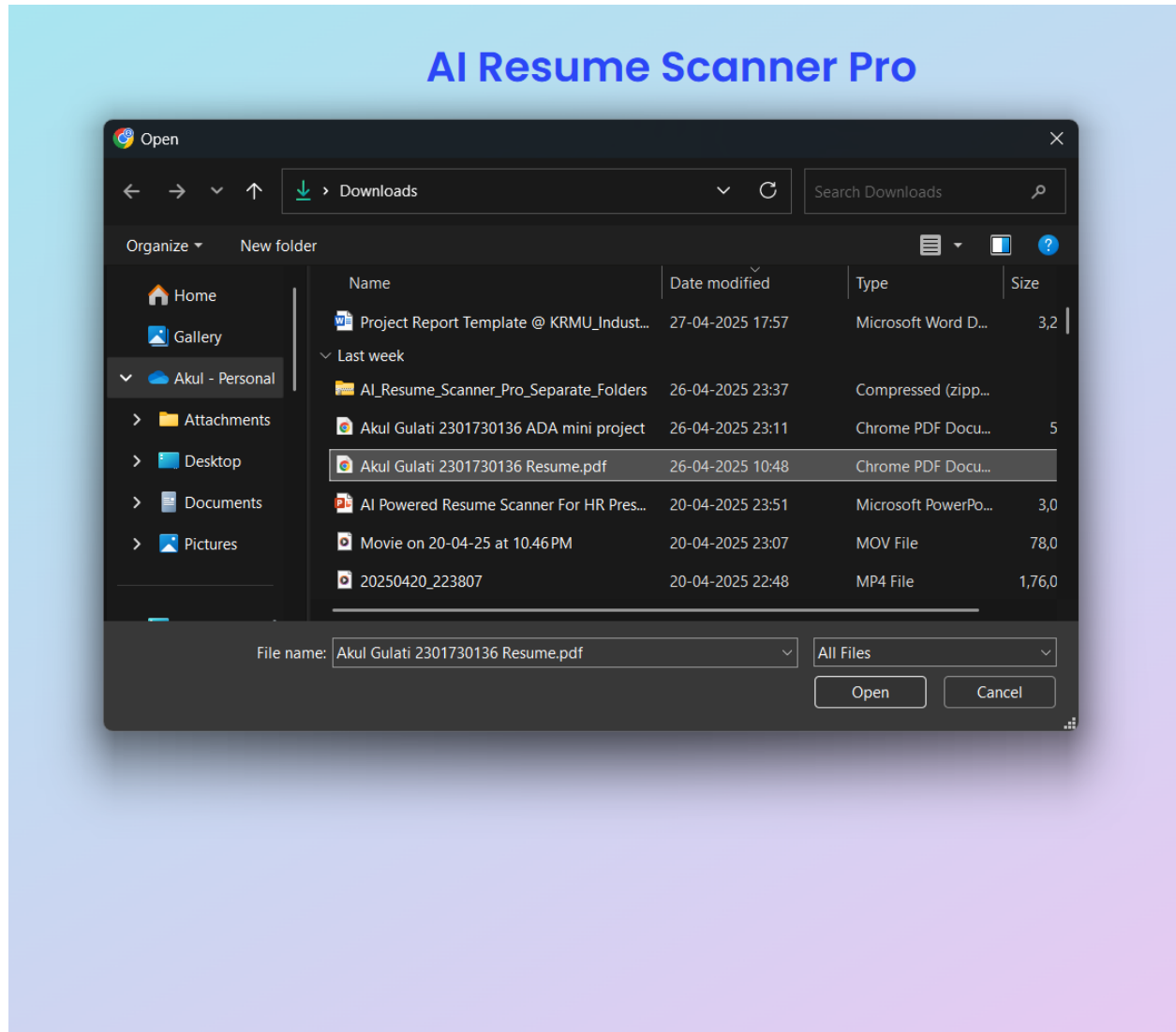
# RESULTS AND DISCUSSIONS

### THE GUI:





## RESUME FILE SELECTION:



## 2.UPLOAD RESUME FILE :


### AI Resume Scanner Pro

Upload your resume and discover your top skills instantly!

Choose File

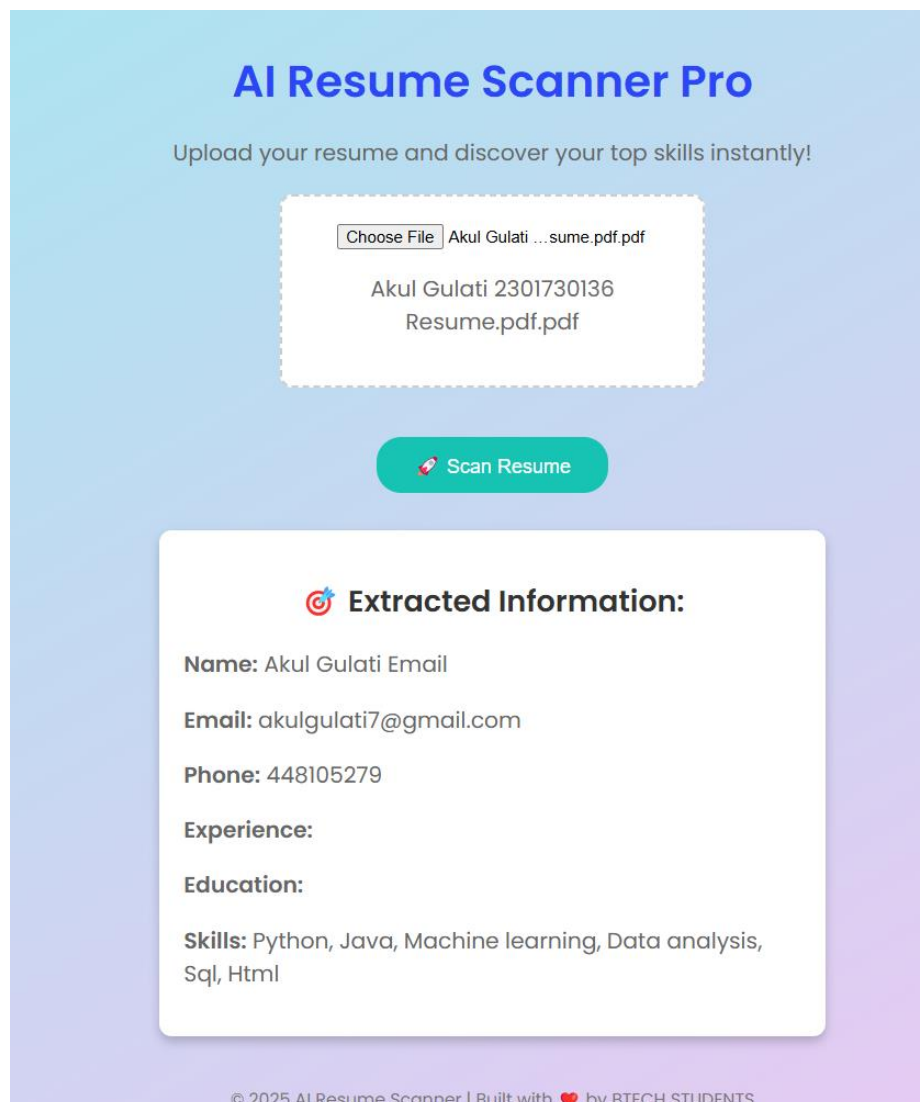
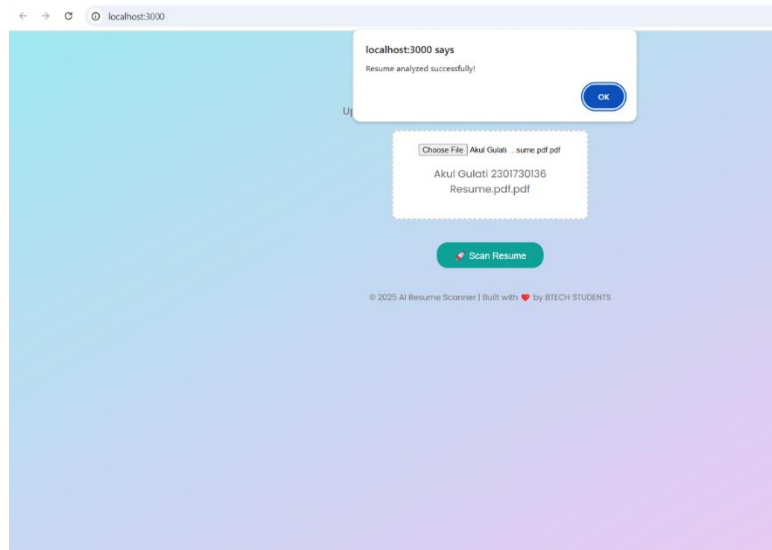
Akul Gulati ...sume.pdf.pdf

Akul Gulati 2301730136  
Resume.pdf.pdf

 Scan Resume

© 2025 AI Resume Scanner | Built with ❤️ by BTECH STUDENTS

### 3.SCAN THE RESUME FOR DETAILS:



## Chapter 6

### FUTURE WORK

Although the current implementation of the **AI Powered Resume Scanner** successfully extracts important information from resumes, there are several areas where the project can be expanded and improved. Future work can focus on enhancing the system's accuracy, scalability, and real-world usability.

### CONCLUSION

The **AI Powered Resume Scanner** project was designed and developed to automate the manual process of reviewing and analyzing resumes, using modern web technologies and Artificial Intelligence (AI) techniques.

The project successfully meets its objectives by:

- Allowing users to **upload** resume files easily through a web-based frontend (React.js),
- Processing those files through a powerful **backend server** (Flask - Python),
- Applying **Natural Language Processing (NLP)** to extract important candidate information such as **Name, Email, Phone Number, Skills, Education, and Experience.**

By integrating frontend and backend technologies, the system ensures a **smooth, efficient, and user-friendly** experience. The use of AI (via spaCy) enabled more intelligent data extraction rather than simple text parsing, offering a **more accurate and automated** solution for resume screening.

The system was tested on multiple sample resumes in PDF and DOCX formats and proved to be reliable in extracting meaningful information. Basic error handling was also implemented to ensure the robustness of the application.

Although the current version achieves the main goals, there is a lot of **scope for future improvements**, such as implementing machine learning models for better ranking, multilingual support, mobile application development, and large-scale cloud deployment.

In conclusion, the **AI Powered Resume Scanner** project demonstrates how modern technologies can simplify traditional hiring processes, saving valuable time and effort for both recruiters and job seekers.

## REFERENCES

- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.** *Proceedings of NAACL-HLT 2019*, 4171–4186.
- Patel, S., Patel, R., & Patel, M. (2019). **Application of Natural Language Processing in Resume Screening.** *International Journal of Engineering and Advanced Technology (IJEAT)*, 8(6), 5334–5338.
- Binns, R., Veale, M., Van Kleek, M., & Shadbolt, N. (2018). **'It's Reducing a Human Being to a Percentage': Perceptions of Justice in Algorithmic Decisions.** *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–14.
- Dastin, J. (2018). **Amazon Scraps Secret AI Recruiting Tool That Showed Bias Against Women.** *Reuters*. Link
- Pang, B., & Lee, L. (2008). **Opinion Mining and Sentiment Analysis.** *Foundations and Trends in Information Retrieval*, 2(1–2), 1–135.
- Agerri, R., Bermúdez, J., & Rigau, G. (2020). **Knowledge Graphs and NLP for Semantic Resume Matching.** *Information Processing & Management*, 57(6), 102352.
- Sainath, T. N., Vinyals, O., Senior, A., & Sak, H. (2015). **Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks.** *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4580–4584.
- Zou, J. Y., & Schiebinger, L. (2018). **AI Can Be Sexist and Racist — It's Time to Make It Fair.** *Nature*, 559(7714), 324–326.
- HireVue. (2020). **HireVue AI Assessments and Ethical AI Principles.** [Online Whitepaper] Available: [HireVue](#)

- Jobscan. (2020). **Optimize Your Resume for ATS with Jobscan.**  
[Online Resource] Available: [Jobscan](#)