**IQRA National University, Peshawar**

****

**Smart Resume Analyzer Using Python & NLP Techniques**

***By***

**Muhammad Suleman Khan INU/FALL21-BSCS-18862/PSH**

**Shah Fahad Nawaz INU/FALL21-BSCS-19054/PSH**

***Supervisor*Salman Ali Khan**

***Bachelor of Science in Computer Science (2021-2025)***

**The candidate confirms that the work submitted is their own and appropriate  
 credit has been given where reference has been made to the work of others**.

****

**IQRA National University, Peshawar**

**Smart Resume Analyzer Using Python & NLP Techniques**

**A project presented to**

**IQRA National University, Peshawar**

**In partial fulfillment**

**of the requirement for the degree of**

***Bachelor of Science in Computer Science (2021-2025)***

**By**

**Muhammad Suleman Khan INU/FALL21-BSCS-18862**

**Shah Fahad Nawaz INU/FALL21-BSCS-19054**

**DECLARATION**

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be the reproduction of some other. We will stand by the consequences. No portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Muhammad Suleman Khan Shah Fahad Nawaz

--------------------------- ---------------------------

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS-CS “**Smart Resume Analyzer Using Python & NLP Techniques**” was developed by **MUHAMMAD SULEMAN KHAN (INU/FALL-2021-BS-CS-18862)** and **SHAH FAHAD NAWAZ (INU/FALL-2021-BS-CS-19054)** under the supervision of “**SALMAN ALI KHAN**” and that in his opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Computer Science or Software Engineering.

**Viva Voce Examination on Project Report of Mr. Muhammad Suleman Khan & Shah Fahad Nawaz was conducted on Tuesday 20th August 2024.**

**Approved By:**

**Supervisor**

Salman Ali Khan

Lecturer Department of Computer Science

IQRA National University, Peshawar

**Internal Examiner**

Huma Mumtaz

Lecturer Department of Computer Science

IQRA National University, Peshawar

**External Examiner**

Dr. Mohammad Mansoor Qadir

Assistant Professor

CECOS University of IT & Emerging Sciences.

**Head of Department**

Prof. Dr. Atif Ishtiaq

Department of Computer Science

IQRA National University, Peshawar

**Executive Summary**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed viverra, nunc a convallis faucibus, risus ipsum commodo mauris, vel dapibus augue est sed elit. Integer fermentum purus a purus scelerisque, vitae gravida neque ultricies. Donec sed tellus vitae sapien fermentum feugiat. Curabitur efficitur, orci at blandit tincidunt, ex turpis malesuada sapien, id fringilla nisl ex vitae enim. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Etiam luctus varius erat, sit amet scelerisque nisl iaculis at. Cras porta, erat nec imperdiet facilisis, nisi velit luctus ipsum, a eleifend justo nibh sed purus. Aliquam erat volutpat. Phasellus bibendum arcu eu risus placerat, a elementum nisl congue. Sed dapibus tristique lacus, a tempus nulla viverra in. Nam feugiat felis eget eros posuere suscipit. Morbi consequat ex at justo imperdiet egestas. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed viverra, nunc a convallis faucibus, risus ipsum commodo mauris, vel dapibus augue est sed elit. Integer fermentum purus a purus scelerisque, vitae gravida neque ultricies. Donec sed tellus vitae sapien fermentum feugiat. Curabitur efficitur, orci at blandit tincidunt, ex turpis malesuada sapien, id fringilla nisl ex vitae enim. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Etiam luctus varius erat, sit amet scelerisque nisl iaculis at. Cras porta, erat nec imperdiet facilisis, nisi velit luctus ipsum, a eleifend justo nibh sed purus. Aliquam erat volutpat. Phasellus bibendum arcu eu risus placerat, a elementum nisl congue. Sed dapibus tristique lacus, a tempus nulla viverra in. Nam feugiat felis eget eros posuere suscipit. Morbi consequat ex at justo imperdiet egestas.

**Acknowledgement**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor “Salman Ali Khan”. Without his personal supervision, advice, and valuable guidance, the completion of this project would have been doubtful. We are deeply indebted to him for his encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

Muhammad Suleman Khan Shah Fahad Nawaz

--------------------------- ---------------------------

**Abbreviations**

|  |  |
| --- | --- |
| **WWW** | World Wide Web |
| **YOLO** | You only look once |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Table of Contents**

[**1 Introduction**](#_Toc520773734) 1

[1.1 Brief 2](#_Toc520773735)

[1.2 Relevance to Course Modules 7](#_Toc520773736)

[1.3 Project Background](#_Toc520773737) 8

[1.4 Literature Review 10](#_Toc520773738)

[1.5 Analysis of Literature Review 17](#_Toc520773739)

[1.6 Problem Statement 18](#_Toc520773738)

[1.7 Proposed Solution 19](#_Toc520773738)

[1.8 Methodology and Software Lifecycle for This Project](#_Toc520773740) 19

1.8.1 [Rationale behind Selected Methodology](#_Toc520773752) 19

[**2** **Requirement Analysis** 2](#_Toc520773748)1

[2.1 System Requirement](#_Toc520773751) 21

[2.2 Requirement Elicitation 22](#_Toc520773751)

2.2.1 [Functional Requirements](#_Toc520773752) 22

[2.2.2 Non-Functional Requirements 23](#_Toc520773753)

[2.3 Tools and Technology 24](#_Toc520773751)

[2.3.1 Hardware Required 24](#_Toc520773753)

[2.3.2 Software Required](#_Toc520773753) 24

[2.3.3 Documentation & Reporting Tools](#_Toc520773753) 24

[2.4 Usecase Diagram 34](#_Toc520773751)

[2.4.1 Usecase Description 34](#_Toc520773751)

[**3 Design and Architecture**](#_Toc520773754) 36

[3.1 System Architecture 36](#_Toc520773759)

[3.2 Input / Output Design](#_Toc520773760) 40

[3.3 Design Models 42](#_Toc520773761)

3.3.1 Flow Chart [43](#_Toc520773752)

[3.3.2 Sequence Diagram 44](#_Toc520773753)

[3.3.3 Activity Diagram 45](#_Toc520773753)

[3.3.4 Data Flow Diagram 46](#_Toc520773753)

[3.3.5 DFD Level 0 Diagram 46](#_Toc520773753)

[3.3.5 DFD Level 1 Diagram 47](#_Toc520773753)

[**4** **Implementation** 48](#_Toc520773763)

[4.1 Algorithm Used 48](#_Toc520773765)

[4.2 Modules 48](#_Toc520773766)

[4.3 Modules Description 49](#_Toc520773767)

4.3.1 Preparing Custom Dataset [49](#_Toc520773752)

4.3.2 Splitting Dataset [50](#_Toc520773752)

4.3.3 Model Training [51](#_Toc520773752)

[4.4 Confusion Matrix 54](#_Toc520773767)

[4.5 F1-Confidence Curve 55](#_Toc520773767)

[4.6 Precision-Confidence Curve 56](#_Toc520773767)

[4.7 Recall-Confidence Curve 57](#_Toc520773767)

[4.8 Precision-Recall Curve 58](#_Toc520773767)

[4.9 Training Results 59](#_Toc520773767)

[4.10 Model Validation 60](#_Toc520773767)

4.10.1 Integration with Computer Vision [60](#_Toc520773752)

[**5 Testing and Evaluation** 62](#_Toc520773768)

[5.1 Testing Dataset Evaluation 62](#_Toc520773771)

[5.2 Real-time Detection Performance](#_Toc520773776) 66

[**6** **Conclusion and Future Work** 70](#_Toc520773778)

[6.1 Conclusion 70](#_Toc520773780)

[6.2 Future Work 71](#_Toc520773781)

6.2.1 Scalability and Deployment [71](#_Toc520773752)

6.2.2 User Interface Enhancement [71](#_Toc520773752)

6.2.3 Continuous Model Improvement and Adaptation  [72](#_Toc520773752)

[**References** 73](#_Toc520773782)

**LIST OF FIGURES**

Fig 1.1: Depicting BRT Crowd Analysis System 3

Fig 1.2: Overview of Ultralytics YOLO v8 4

Fig 1.3: The essential mechanics of YOLO object detection model 5

Fig 1.4: Structure of YOLOv8 detection model 6

Fig 1.5: Agile Diagram of Crowd Analysis System 20

Fig 2.1: Venn Diagram of AI, ML, DL & CV Relationships 29

Fig 2.2: Neural Network with Hidden Layers 30

Fig 2.3: Use Case Diagram 34

Fig 3.1: System Architecture 36

Fig 3.2: YOLOv8 Model Architecture 39

Fig 3.3: Input Design of System 41

Fig 3.4: Output Window GUI 42

Fig 3.5: Flow Chart of System 43

Fig 3.6: System Sequence Diagram 44

Fig 3.7: Activity Diagram of System 45

Fig 3.8: Level 0 DFD 46

Fig 3.9: Level 1 DFD 47

Fig 4.1: Dataset Annotation with ImgLabel 49

Fig 4.2: Splitting of Dataset 50

Fig 4.3: Train Batch\_0 of Images (first) 51

Fig 4.4: Train Batch\_632 of Images (last) 52

Fig 4.5: Training Parameters of YOLOv8 Model 53

Fig 4.6: Confusion Matrix 54

Fig 4.7: F1- Confidence Curve 56

Fig 4.8: Precision-Confidence Curve 57

Fig 4.9: Recall-Confidence Curve 58

Fig 4.10: Precision-Recall Curve 59

Fig 4.11: Training Results of YOLOv8 Model 59

Fig 4.12: Validation Batch\_1 of Images 60

Fig 4.13: Graphical Overlays with OpenCV 61

Fig 5.1: Testing Image\_1 62

Fig 5.2: Testing Image\_2 63

Fig 5.3: Testing Image\_3 63

Fig 5.4: Testing Image\_4 64

Fig 5.5: Testing Image\_5 64

Fig 5.6: Testing image\_6 65

Fig 5.7: Real-time Video\_Feed\_1 66

Fig 5.8: Real-time Video\_Feed\_2 67

Fig 5.9: Real-time Video\_Feed\_3 68

Fig 5.10: Real-time Video\_Feed\_4 68

Fig 5.11: Real-time Video\_Feed\_5 69

**LIST OF TABLES**

Table 1.1 Comprehensive Features of YOLOv8 Model 4

Table 1.2 Applications of Crowd Analysis 6

Table 2.1 Comparison of Deep Learning Architectures 32

Table 4.1 Confusion Matrix Table 55

**Abstract**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed viverra, nunc a convallis faucibus, risus ipsum commodo mauris, vel dapibus augue est sed elit. Integer fermentum purus a purus scelerisque, vitae gravida neque ultricies. Donec sed tellus vitae sapien fermentum feugiat. Curabitur efficitur, orci at blandit tincidunt, ex turpis malesuada sapien, id fringilla nisl ex vitae enim. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Etiam luctus varius erat, sit amet scelerisque nisl iaculis at. Cras porta, erat nec imperdiet facilisis, nisi velit luctus ipsum, a eleifend justo nibh sed purus. Aliquam erat volutpat. Phasellus bibendum arcu eu risus placerat, a elementum nisl congue. Sed dapibus tristique lacus, a tempus nulla viverra in. Nam feugiat felis eget eros posuere suscipit. Morbi consequat ex at justo imperdiet egestas.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed viverra, nunc a convallis faucibus, risus ipsum commodo mauris, vel dapibus augue est sed elit. Integer fermentum purus a purus scelerisque, vitae gravida neque ultricies. Donec sed tellus vitae sapien fermentum feugiat. Curabitur efficitur, orci at blandit tincidunt, ex turpis malesuada sapien, id fringilla nisl ex vitae enim. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Etiam luctus varius erat, sit amet scelerisque nisl iaculis at. Cras porta, erat nec imperdiet facilisis, nisi velit luctus ipsum, a eleifend justo nibh sed purus. Aliquam erat volutpat. Phasellus bibendum arcu eu risus placerat, a elementum nisl congue. Sed dapibus tristique lacus, a tempus nulla viverra in. Nam feugiat felis eget eros posuere suscipit. Morbi consequat ex at justo imperdiet egestas.

**Keywords:** to be added

**CHAPTER 1**

**INTRODUCTION**

**Introduction**

The Smart Resume Analyzer is an advanced web application designed to streamline and enhance the recruitment process by automatically analyzing and matching resumes to job descriptions. In today's competitive job market, recruiters and hiring managers face the challenge of efficiently screening large volumes of resumes to identify candidates with the most relevant skills and qualifications. This application addresses this challenge by leveraging Natural Language Processing (NLP) techniques and machine learning algorithms to automate the initial screening process, reducing time and effort while improving the quality of candidate selection. It ensures a more objective, data-driven approach to recruitment by identifying the most relevant candidates, minimizing human error, and allowing recruiters to focus on strategic decision-making tasks.

The system provides an intuitive web interface where users can upload multiple resume PDF files and input a job description. It then processes these documents to extract key information such as skills, education, experience, and other relevant details. Using sophisticated text analysis techniques, the application calculates a match score between each resume and the job description, identifying required and preferred skills, missing qualifications, and overall semantic relevance.

What sets this application apart is its intelligent weighting system that allows users to customize the importance of different matching criteria, such as required skills, preferred skills, and overall contextual similarity. The system presents results through an interactive dashboard with visualizations including radar charts for skill categories and bar charts for skill gaps, enabling quick identification of top candidates.

This project not only addresses the limitations of traditional keyword-based filters but also contributes to reducing unconscious bias in early-stage recruitment. By providing standardized and transparent evaluation metrics, the Smart Resume Analyzer promotes fairness and inclusivity in candidate assessment.

**1.1 Brief**

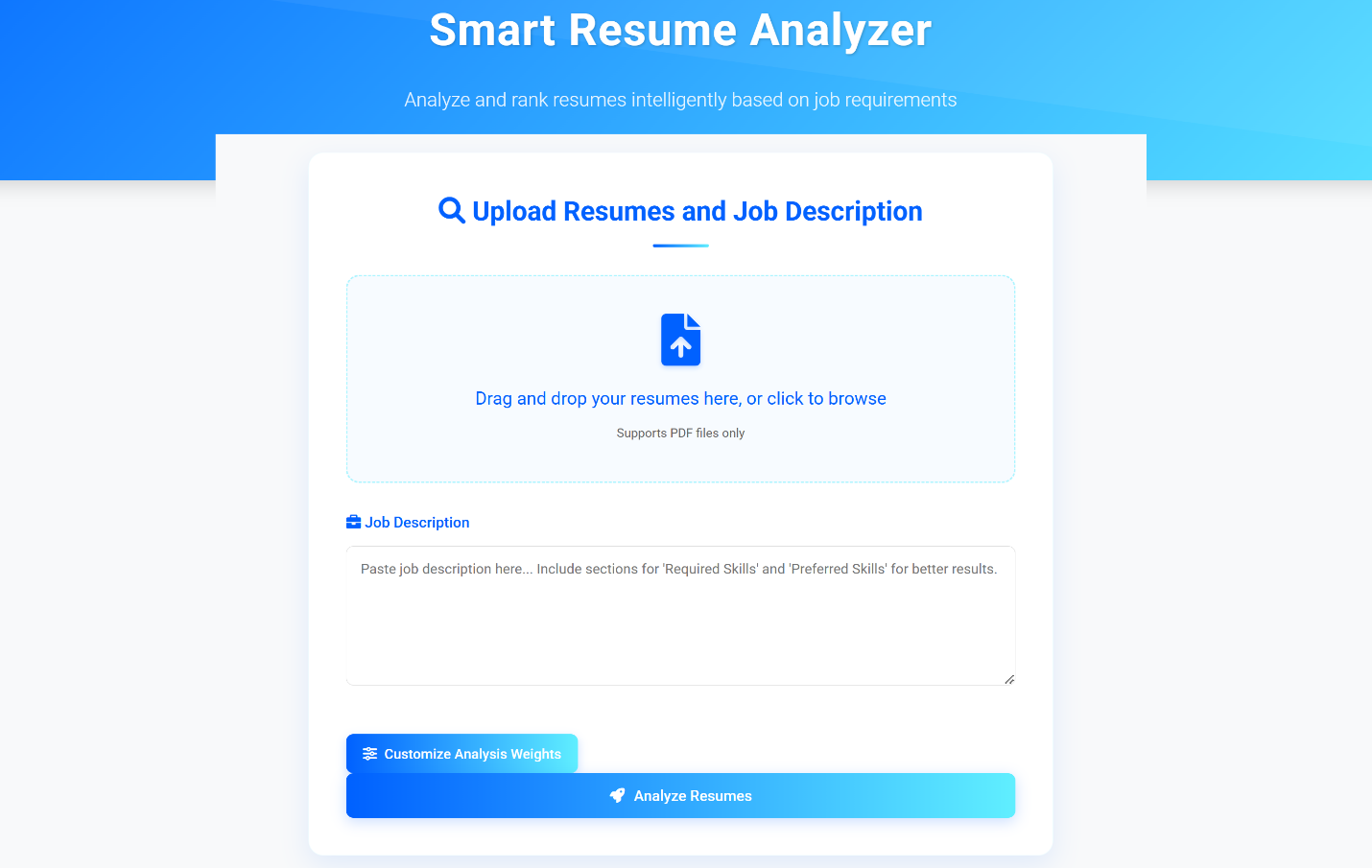
The Smart Resume Analyzer is a comprehensive web-based solution designed to modernize the early stages of recruitment by automating resume analysis and matching. Built using Python’s Flask framework for the backend and enhanced with advanced Natural Language Processing (NLP) techniques, the system offers recruiters a scalable and intelligent platform to process resumes more efficiently and with greater accuracy. Traditional resume screening practices are often time-consuming, inconsistent, and prone to subjective bias. The Smart Resume Analyzer aims to mitigate these challenges by introducing a structured, algorithmic approach to candidate evaluation.

At its core, the system allows users to upload one or more resume files in PDF format and input a job description via text input or file upload. Once submitted, the application initiates a multi-stage processing pipeline. This includes document parsing, section recognition (e.g., education, skills, work experience), and key information extraction using a combination of rule-based patterns and machine learning techniques. The extracted data is then compared to the job description, and a weighted matching algorithm computes a compatibility score for each resume.

One of the standout features of the system is its customizable scoring algorithm, which gives recruiters control over how much weight is assigned to various factors such as required skills, preferred skills, and semantic similarity. This personalization enhances the accuracy of the candidate-job fit based on organizational priorities. Additionally, the application incorporates semantic similarity analysis using word embeddings from spaCy, which allows it to capture contextual meaning beyond simple keyword matches.

To present results in a meaningful way, the system generates visual dashboards that include radar charts and bar graphs, highlighting skill coverage and gaps for each candidate. Recruiters can quickly identify the most qualified applicants and view detailed resume insights.

By streamlining the resume screening process and offering insightful analytics, the Smart Resume Analyzer provides significant value in reducing hiring cycle time, improving selection quality, and promoting fairness. It stands as a practical demonstration of how artificial intelligence and NLP can transform recruitment practices in modern organizations.



**Figure 1.1: Smart Resume Analyzer Web Application**

In this project, Python serves as the core programming language due to its simplicity, versatility, and extensive ecosystem of libraries. Python's powerful libraries such as NLTK, spaCy, scikit-learn, and pandas are leveraged for processing, analyzing, and manipulating textual data efficiently.

Natural Language Processing (NLP) plays a central role in the Smart Resume Analyzer by enabling the extraction of meaningful information from resumes and job descriptions. Using NLP techniques, the system tokenizes text, removes stop words, and performs lemmatization to normalize the data. Named Entity Recognition (NER) and part-of-speech tagging help identify key skills, qualifications, and experiences.

The processed text is then vectorized using TF-IDF or similar techniques to compute similarity scores. These techniques enable accurate and intelligent matching between candidate resumes and job requirements, enhancing decision-making in recruitment.

**Table 1.1: Features of Smart Resumes Analyzer**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Resume Upload (PDF) | |  | | --- | | Allows users to upload multiple resume PDF files for batch processing and analysis. |  |  | | --- | |  | |
| Job Description Input | |  | | --- | | Provides a field to enter or upload the job description against which resumes are to be matched. |  |  | | --- | |  | |
| NLP-Based Text Extraction | |  | | --- | | Utilizes NLP techniques to extract skills, education, experience, and other relevant information from text. |  |  | | --- | |  | |
| Match Scoring System | |  | | --- | | Calculates a score based on how closely each resume aligns with the job description criteria. |  |  | | --- | |  | |
| Customizable Weighting Criteria | |  | | --- | | Enables users to assign different importance levels to various matching criteria (skills, experience, etc.). |  |  | | --- | |  | |
| Skill Gap Identification | |  | | --- | | Highlights missing or underrepresented skills in each resume relative to the job description. |  |  | | --- | |  | |
| Interactive Dashboard | |  | | --- | | Displays results visually using bar charts, radar charts, and other interactive elements. |  |  | | --- | |  | |

The Smart Resume Analyzer is equipped with a range of intelligent features that enhance the efficiency and accuracy of the recruitment process. Users can upload multiple resumes in PDF format, and input a job description for comparative analysis. The system employs Natural Language Processing (NLP) techniques to extract key information from resumes, such as skills, education, and work experience. One of the core functionalities is the match scoring system, which calculates how closely each resume aligns with the job description based on various criteria. Users can also customize the importance of these criteria through an intuitive weighting system. Additionally, the application identifies skill gaps, highlighting missing qualifications for each candidate. Results are presented in an interactive dashboard using radar and bar charts for easy visualization. By focusing on objective, data-driven metrics, the system reduces unconscious bias.

**Table 1.2: Applications of Resume Analyzer**

|  |  |
| --- | --- |
| **Application** | **Description** |
| Automated Resume Screening | |  | | --- | | Speeds up the hiring process by automatically filtering and ranking resumes based on job relevance. |  |  | | --- | |  | |
| Skill Gap Analysis | |  | | --- | | Identifies missing skills in candidate profiles compared to job requirements for better evaluation. |  |  | | --- | |  | |
| Candidate Shortlisting | |  | | --- | | Helps recruiters quickly shortlist top candidates by providing match scores and insights through dashboards. |  |  | | --- | |  | |
| Bias Reduction in Hiring | |  | | --- | | Promotes fair hiring by using standardized, data-driven evaluations, reducing human bias in initial screening. |  |  | | --- | |  | |
| Customizable Match Criteria | |  | | --- | | Allows recruiters to prioritize different criteria (skills, experience, education) according to job needs. |  |  | | --- | |  | |
| |  | | --- | | Visual Resume Insights |  |  | | --- | |  | | Generates visual analytics (charts, graphs) to give recruiters an overview of candidate strengths and gaps. |

**1.2 Relevance to Course Modules**

**Programming Fundamentals:**

The development of the Smart Resume Analyzer extensively applies core programming fundamentals taught in the course. Concepts such as control structures, data types, functions, file handling, and modular programming are utilized to build the application's backend logic. Python, a high-level language covered in the curriculum, is used to manage file uploads, extract resume data, and structure the application's flow. Error handling, user input validation, and object-oriented design principles are incorporated to ensure a robust and maintainable codebase. These foundational skills are critical in creating a functional and efficient application that can process and evaluate multiple resumes seamlessly.

**Natural Language Processing (NLP):**

Natural Language Processing is at the heart of this project, enabling the system to analyze unstructured text in resumes and job descriptions. Course modules on NLP provided essential knowledge on techniques such as tokenization, lemmatization, named entity recognition, and part-of-speech tagging, all of which were applied during resume parsing. Libraries like spaCy and NLTK were used to extract relevant entities such as skills, experience, and education. Semantic similarity and keyword matching further enhance the resume-job matching process. By using NLP, the application interprets human language meaningfully, making the analysis intelligent and context-aware.

**Machine Learning:**

Machine learning plays a significant role in improving the accuracy and intelligence of the Smart Resume Analyzer. The project incorporates supervised learning techniques to enhance resume matching by training models on labeled data, identifying patterns between resumes and ideal job profiles. Techniques like feature extraction, vectorization (using TF-IDF), and classification algorithms help the system rank resumes more effectively. Course modules on ML covered model evaluation, data preprocessing, and performance metrics—all directly applied in the project. This integration ensures that the system improves over time and adapts to changing hiring criteria based on historical data and feedback.

**Artificial Intelligence:**

The Smart Resume Analyzer reflects core AI concepts by simulating human decision-making in candidate screening. It applies AI techniques to evaluate resumes beyond basic keyword filtering, taking context, semantics, and relevance into account. Knowledge from the AI course modules, such as decision trees, search algorithms, and intelligent agents, shaped the logic that underpins the scoring and recommendation system. The tool's ability to mimic intelligent behavior—such as selecting the most suitable candidate based on multiple weighted parameters—demonstrates the practical application of AI theories in solving real-world problems in recruitment and human resources.

**1.3 Project Background**

In today’s competitive and digitally driven job market, organizations receive hundreds, sometimes thousands, of resumes for a single job opening. Traditionally, human recruiters manually sift through these resumes to identify candidates who best match the job requirements. This manual process is not only time-consuming and inefficient but also prone to human error, subjectivity, and unconscious bias. Moreover, due to the sheer volume of applications, qualified candidates may be overlooked if their resumes are not formatted in a way that easily highlights their relevant skills and experience. This growing challenge in the recruitment landscape calls for intelligent, automated solutions that can improve the accuracy and fairness of candidate selection.

The idea for the Smart Resume Analyzer project was born from this gap in the recruitment process. The goal was to develop a web-based tool that can automatically analyze resumes and match them with a given job description using advanced technologies like Natural Language Processing (NLP) and machine learning. By converting unstructured resume data into structured, analyzable content, the application can identify the most suitable candidates quickly and objectively. This significantly reduces the workload on HR professionals and recruiters while enhancing the quality of the hiring process.

The project integrates multiple modules and technologies that are directly relevant to the academic coursework of computer science, including programming fundamentals, data structures, algorithms, machine learning, and artificial intelligence. The system parses resumes, extracts key details such as skills, experience, and education, and compares them with job description keywords. It then assigns a match score based on various parameters such as required skills, preferred skills, and semantic relevance, providing a fair and consistent evaluation of all candidates.

Additionally, the Smart Resume Analyzer provides visual insights through charts and graphs, enabling quick decision-making. It also reduces the reliance on rigid keyword-based filters by incorporating context-aware analysis, making it more robust and intelligent. The use of customizable weights in the scoring system adds flexibility, allowing recruiters to prioritize certain criteria based on the nature of the job.

Ultimately, this project not only reflects a practical application of academic knowledge but also addresses a real-world problem. It stands as a testament to how technology can be harnessed to enhance human resource functions, making the recruitment process faster, more transparent, and more effective.

**1.4 Literature Review**

Several techniques have been explored to automate the resume parsing process, and a thorough review of these methods is presented in [1]. The authors outline the progression from rule-based approaches to more sophisticated methods involving machine learning and Natural Language Processing (NLP). Techniques like Named Entity Recognition (NER), syntactic parsing, and semantic role labeling are emphasized for their role in extracting structured data from unstructured resumes. The study discusses common challenges such as inconsistent formatting, ambiguity in language, and variability in resume structures. It also outlines future directions that include the development of domain-adaptive models and improved evaluation benchmarks. The work stresses the need for robust solutions that can handle diverse resume types and deliver accurate, context-aware parsing, making it a foundational reference for systems aiming to automate recruitment processes.

In [2], the authors propose a dual-purpose system that not only analyzes resumes but also recommends skill enhancement resources based on the analysis. The framework utilizes NLP to extract essential information from resumes and match them against job descriptions. By identifying missing skills or qualifications, the system provides targeted learning suggestions through integrated educational platforms. The model includes stages like keyword extraction, semantic similarity computation, and skill mapping. A unique feature of the system is its ability to guide users toward professional development rather than merely filtering candidates. The paper emphasizes the importance of using AI not only for shortlisting but also for empowering users to improve employability. This approach aligns with modern trends in AI-based recruitment systems that integrate skill development and career guidance.

A novel approach to resume-job description matching is discussed in [3], where the authors propose a semantically enriched Bag-of-Words (BoW) model. Traditional BoW methods are criticized for ignoring the context and meaning of words, which this study addresses by incorporating word embeddings and semantic similarity metrics. The model accounts for synonyms, domain-specific terms, and contextual relevance using pre-trained language models, allowing for more accurate candidate-job matches. It also includes normalization techniques to reduce redundancy and enhance interpretability. Experimental results show improved matching accuracy and relevance compared to basic keyword search. This study underscores the necessity of semantic understanding in recruitment tools and serves as a guide for developing systems that analyze not just words but the intent and contextual suitability of a candidate’s profile.

In [4], the authors explore the effectiveness of K-Means clustering and Random Forest classification for resume classification by experimenting with various cluster sizes and dataset combinations. The study highlights how altering cluster sizes impacts the accuracy and relevance of resume categorization. K-Means is employed to group similar resumes, while Random Forest is used to classify them into predefined job categories. The performance of both algorithms is evaluated based on classification accuracy and computational efficiency. The authors demonstrate that optimal cluster size and balanced datasets significantly improve model performance. This research offers insights into combining unsupervised and supervised learning methods for better classification outcomes in resume screening applications, illustrating how clustering can aid in preprocessing and improving recruitment workflows when paired with intelligent classification strategies.

In [5], the authors present a resume analysis tool built using Natural Language Processing (NLP) to extract relevant candidate information and assess suitability against job descriptions. The system uses techniques such as tokenization, part-of-speech tagging, and named entity recognition to retrieve key attributes like education, experience, and technical skills from unstructured resumes. The model also incorporates text similarity measures to compare extracted features with job requirement keywords. This study underscores the power of NLP in automating the tedious task of manual resume screening and aims to reduce human bias by standardizing candidate evaluation. The proposed system is particularly beneficial in large-scale hiring scenarios where manual screening becomes impractical. By demonstrating the feasibility of NLP-based extraction and comparison, this research supports the foundation for advanced automated recruitment platforms.

[6] Thangaramya et al. (2024) introduced an automated resume parsing and ranking system that harnesses the power of Natural Language Processing (NLP). Their approach systematically processes resumes to extract structured data such as skills, experience, and qualifications, converting unstructured text into a usable format for automated analysis. The parsed information is then evaluated and ranked against job descriptions using semantic similarity measures. A significant contribution of this study is the integration of advanced NLP techniques like dependency parsing and named entity recognition to enhance information retrieval accuracy. The proposed framework reduces human effort and increases the consistency of resume screening, making it highly applicable for recruitment agencies and large HR departments. This work highlights the growing potential of NLP in transforming manual hiring into a data-driven, automated workflow.

[7] Sharif et al. (2024) developed a smart assistant that leverages NLP and machine learning to select the most suitable resumes from a candidate pool. Their system is designed to assist recruiters by analyzing resumes, ranking them based on relevance to predefined job criteria, and identifying top-performing candidates. It uses a combination of keyword extraction, context-aware embeddings, and classification models to evaluate candidates comprehensively. What sets this study apart is its intelligent assistant capability, which not only parses resumes but also learns from feedback to improve future recommendations. The system is aimed at reducing recruiter bias and increasing hiring accuracy. This research demonstrates how integrating intelligent algorithms with user feedback loops can significantly enhance the resume evaluation process and support decision-making in high-volume recruitment scenarios.

[8] Mohamed et al. (2024) introduced ResuFit, a precision CV matching system that enhances job application processes through intelligent resume-job alignment. The system employs NLP techniques to extract key attributes from resumes and match them with job descriptions using vector space models and similarity algorithms. A major focus is placed on achieving high accuracy in matching by considering not only technical skills but also soft skills and experience depth. The framework supports dynamic updates, ensuring adaptability to changing job requirements and candidate profiles. ResuFit distinguishes itself by implementing a scoring mechanism that enables recruiters to instantly assess candidate fit, streamlining the screening phase. This study offers valuable insight into precision-based recruitment tools that can transform talent acquisition through automation and data-driven analysis.

Several techniques have been explored to automate the resume parsing process, and a comprehensive analysis of NLP-based approaches is presented in [9]. The authors propose a system that utilizes tokenization, stemming, and entity recognition to extract structured data from free-form resumes. Their model efficiently categorizes resumes by matching skills, education, and experience with predefined job criteria. This work emphasizes the adaptability of NLP in handling diverse resume formats and the potential of such systems to streamline recruitment. The study also discusses performance metrics and real-time implementation, showcasing the tool’s accuracy and scalability. By replacing manual screening processes, the system significantly reduces recruiter workload and enhances decision-making, making it a valuable tool in talent acquisition and HR automation.

Several studies have investigated the integration of AI into resume analysis, and [10] presents a system that not only analyzes resumes but also provides skill recommendations. The proposed model employs natural language processing and machine learning algorithms to assess candidate profiles and identify skill gaps. By analyzing job market trends and comparing them with individual resumes, the system offers personalized suggestions to improve employability. This dual approach of parsing and feedback makes the model more than just a filtering tool—it also supports career development. The study underlines how AI can bridge the gap between employer expectations and candidate profiles while enhancing both user experience and recruiter efficiency. Evaluation results show that the system delivers relevant recommendations, thereby making the screening and upskilling process more effective.

Numerous approaches have been suggested for smart recruitment systems, and [11] introduces SATYA, an AI-driven platform for resume screening and recommendation evaluation. This system integrates semantic understanding and deep learning to ensure high-accuracy profile matching. SATYA not only evaluates resumes for relevant qualifications and experience but also provides reasoning for its recommendations using explainable AI techniques. It supports recruiters in shortlisting candidates with greater confidence by highlighting the strengths and shortcomings of each profile. The authors emphasize SATYA’s iterative learning mechanism, which refines the algorithm based on feedback. This model demonstrates how intelligent systems can bring transparency and efficiency to hiring practices, reducing biases and optimizing the candidate selection process.

Efficient hiring processes have become a key priority in the corporate world, and [12] presents a solution focused on streamlining talent acquisition through detailed resume analysis. The authors propose a system that extracts structured insights from resumes, emphasizing the use of analytics to align candidate profiles with job requirements. By leveraging text mining techniques and classification algorithms, their model identifies key attributes such as skills, education, and experience. The paper highlights the importance of data-driven hiring and demonstrates how such systems can improve recruitment strategies by minimizing bias and manual effort.

A practical implementation of machine learning in resume parsing is demonstrated in [13], where the authors developed a skill extractor using Streamlit for the user interface. This work employs traditional ML algorithms to recognize and categorize skills from unstructured resume text. The tool is designed to assist both job seekers and recruiters by visualizing extracted data, making it easier to assess the candidate’s strengths. Unlike deep learning models, this lightweight approach emphasizes accessibility and interpretability, making it well-suited for small to mid-sized recruitment needs or educational environments.

Matching resumes with job descriptions is a critical task in recruitment automation, and [14] introduces a system that addresses this challenge using a combination of NLP and semantic similarity measures. The model parses resumes and compares them with job descriptions to determine the best fit based on skill and keyword alignment. The authors report promising results in ranking resumes accurately, showing how context-aware models can significantly enhance selection precision. This study reinforces the importance of contextual understanding in resume-job matching systems.

The study presented in [15] investigates the development of an AI-driven resume screening system leveraging Natural Language Processing (NLP) to evaluate, classify, and rank applicant profiles. This system employs advanced parsing techniques and entity extraction methods to identify relevant skills, qualifications, and experience from unstructured resume data. A key aspect of the research is the implementation of a profile scoring mechanism that allows recruiters to prioritize candidates more efficiently. The study also emphasizes the creation of a user-friendly interface, enabling HR professionals to interact with and interpret parsed data effectively. Validated using real-world data, the tool demonstrated improved accuracy and speed in the screening process. This reflects the increasing adoption of AI technologies in human resource practices to augment traditional recruitment workflows and reduce manual effort.

Building on similar technological principles, the study in [16] introduces a machine learning-based system designed for both resume parsing and job recommendation. This system extracts essential information from resumes using classification and NLP techniques and then compares the extracted data with job descriptions to find suitable matches. It incorporates similarity algorithms to ensure contextual relevance between a candidate’s profile and job postings. By tailoring job suggestions to the candidate's specific skills and experiences, the model increases applicant engagement and satisfaction. This dual-functionality system not only streamlines the recruitment process but also enhances the applicant experience, addressing both recruiter and candidate needs. The research presents a scalable, intelligent framework that could be integrated into broader HR management systems.

Together, these studies demonstrate the transformative role of AI and NLP in automating and refining recruitment processes. They underline the potential for intelligent systems to not only assist recruiters in decision-making but also to personalize candidate experiences, thereby improving overall hiring outcomes and organizational efficiency.

**1.5 Analysis of Literature Review**

The literature review reveals significant advancements in the field of resume parsing and analysis, highlighting various techniques that utilize Natural Language Processing (NLP) and machine learning (ML) to automate and enhance the recruitment process. Several studies emphasize the importance of automated resume parsing systems, particularly for extracting valuable insights from unstructured resume data. For example, [1] and [6] showcase how NLP techniques, such as Named Entity Recognition (NER) and text classification, are widely applied to detect skills, experiences, and qualifications from resumes, reducing manual labor and bias in recruitment.

The growing trend of integrating machine learning models, such as decision trees and Random Forests, with resume analysis is also evident in multiple studies. [4], [10], and [16] present systems where ML models are used to classify and rank resumes based on their suitability for specific job descriptions. The use of semantic similarity algorithms in matching resumes to job profiles, as discussed in [13] and [14], ensures that the extracted data is relevant, and the system can make context-aware decisions. This is a crucial advancement, as it allows for more precise and efficient matching than simple keyword-based methods.

Another notable trend is the development of user-friendly interfaces, such as Streamlit, to visualize the data extracted from resumes. As shown in [13], this approach not only improves the usability of the system but also facilitates better decision-making for recruiters by making the extracted data easily interpretable. Furthermore, [12] and [15] highlight the growing integration of AI for automating job recommendation systems, indicating a shift towards end-to-end HR solutions that improve candidate-job matching.

In summary, the literature shows a trend toward combining various AI and NLP techniques to streamline recruitment processes. While most systems aim to improve accuracy, efficiency, and fairness in resume screening, future research may focus on enhancing the interpretability and adaptability of these systems to accommodate diverse industries and dynamic job market needs.

**1.6 Problem Statement**

The recruitment process is often overwhelmed by the high volume of resumes submitted for a single job opening. Manually screening resumes is time-consuming, prone to bias, and prone to errors, which leads to suboptimal candidate selection. Traditional keyword-based filtering does not capture the full context of a candidate’s qualifications, and important skills may be missed. There is a need for an automated, objective, and efficient system that can accurately analyze resumes, match them to job descriptions, and improve the overall recruitment process.

**1.7 Proposed Solution**

The Smart Resume Analyzer offers an automated solution to the resume screening process by leveraging Natural Language Processing (NLP) and machine learning techniques. The system analyzes resumes and matches them to job descriptions based on key skills, qualifications, and experience. It utilizes advanced text analysis to evaluate both explicit and implicit information within resumes, offering a more comprehensive and accurate match. The application’s intelligent weighting system allows recruiters to customize the matching criteria, ensuring a flexible, objective, and efficient approach to candidate evaluation, reducing manual effort and bias.

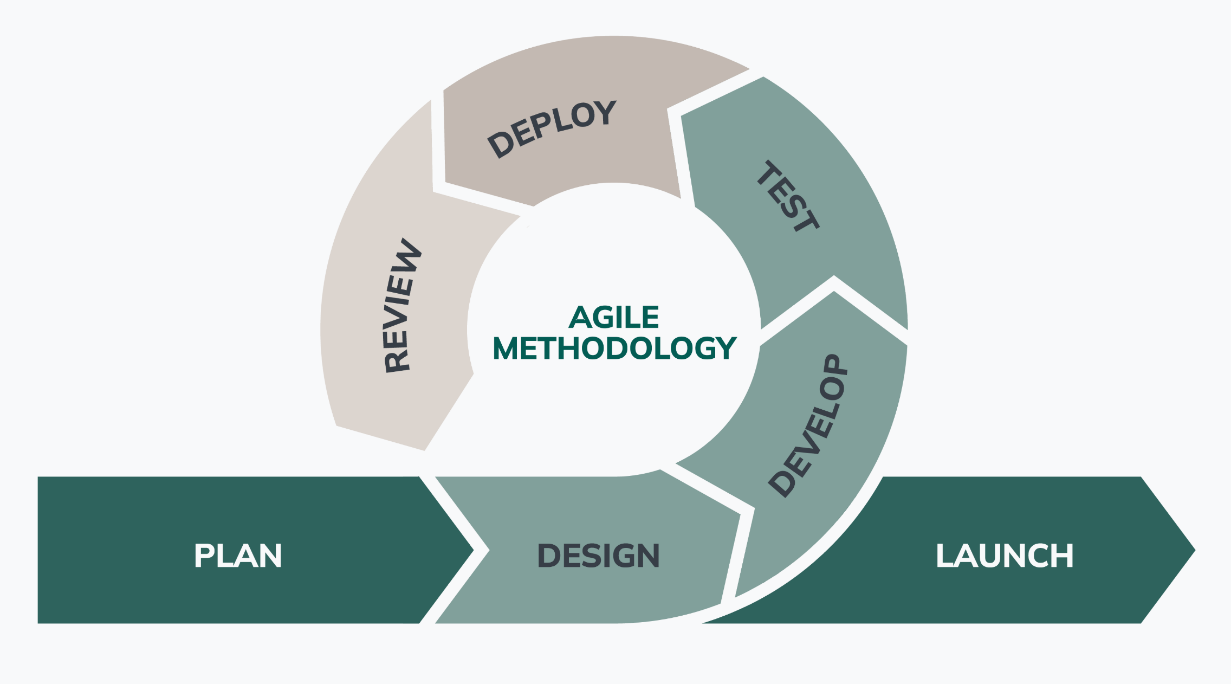
**1.8 Methodology and Software Lifecycle for This Project**

This project follows the Agile methodology, which emphasizes iterative development and continuous feedback. The Agile approach enables flexibility, allowing for regular adjustments and enhancements based on user feedback. Through short development cycles (sprints), the project progresses incrementally, ensuring efficient execution and timely delivery. Regular testing, review, and adaptation at each stage contribute to maintaining high-quality standards and meeting the project’s evolving requirements.

**1.8.1 Rationale Behind Selected Methodology**

The Agile methodology was chosen for this project due to its flexibility, adaptability, and focus on continuous improvement. Given the dynamic nature of resume analysis and the need for regular updates based on user feedback, Agile allows for iterative development, quick adjustments, and early detection of issues. This approach ensures that the project evolves effectively, aligning with the changing requirements and delivering a high-quality product in a timely manner, while also promoting collaboration, transparency, and consistent stakeholder engagement throughout the development lifecycle.

* **Analyze and Plan:** Define project objectives, identify key requirements, and develop a roadmap for implementation. Understand user needs and ensure alignment with project goals.
* **Design:** Create system architecture and UI/UX designs. Focus on ensuring functionality, scalability, and a user-friendly interface, incorporating design feedback from stakeholders.
* **Build:** Develop the application using selected technologies. Implement features incrementally, ensuring code quality, efficiency, and integration with backend systems.
* **Test:** Conduct thorough testing including unit, integration, and user acceptance tests. Ensure the system meets functional and performance
* **Review:** Evaluate project progress, review code and design for quality, and gather user feedback to identify areas for improvement. Adjust project direction as necessary.
* **Launch:** Deploy the application to production, ensuring it is fully functional. Monitor its performance, address any issues, and provide post-launch support for users.



**Figure 1.5: Agile Diagram of Crowd Analysis System**

**CHAPTER 2**

**REQUIREMENT ANALYSIS**

**2.1 System Requirement**

**Feasibility Study:**

A feasibility study evaluates whether the Smart Resume Analyzer system can be successfully developed and deployed. It assesses various dimensions including technical resources, financial viability, and operational capacity. This study ensures that the project is practical, achievable, and beneficial for users, especially recruiters and job seekers, while minimizing potential risks and aligning with available capabilities.

**Technical Feasibility:**

The technical feasibility confirms that the required technology stack, including Natural Language Processing (NLP), machine learning algorithms, and backend support with Python, is available and accessible. All necessary development tools, cloud infrastructure, and libraries are compatible with the system requirements, ensuring that the development team can build and maintain the system efficiently.

**Economic Feasibility:**

The economic feasibility evaluates the cost-effectiveness of the project. Since the system utilizes mostly open-source tools and platforms, development costs remain minimal. Basic hosting, storage, and deployment are affordable, especially during the prototyping phase. The anticipated benefits, such as time and cost savings in recruitment, outweigh the initial expenses, making the project economically viable.

**Operational Feasibility:**

Operational feasibility focuses on whether the system can function effectively within its intended environment. Given that the tool targets HR departments and job seekers with basic technical proficiency, it is designed with a user-friendly interface. The system aligns with current recruitment workflows and can be easily integrated into existing HR processes, ensuring smooth daily operations.

**2.2 Requirement Elicitation**

Requirement elicitation involves gathering needs and expectations from stakeholders to define the system's capabilities. For the Smart Resume Analyzer, inputs were collected through interviews, surveys, and review of existing systems to understand functional goals, performance criteria, and user expectations.

**2.2.1 Functional Requirements**

The functional requirements define the specific actions and operations that the Smart Resume Analyzer system must perform:

1. **Resume Upload**: Users must be able to upload their resumes through a simple and secure interface. The system should accept resumes in commonly used file formats such as PDF and DOCX. The upload mechanism must include basic validation to ensure file type and size limits are met, and it should notify users of successful submission or errors.
2. **Resume Parsing**: Once uploaded, the system should automatically parse the resume to extract structured data. This includes personal details like name, phone number, email, educational qualifications, work experience, technical skills, certifications, and project work. Natural Language Processing (NLP) models should be used to ensure accurate and context-aware extraction, regardless of formatting differences in various resumes.
3. **Skill Matching**: The extracted skills from the resume should be compared against a predefined set of job-related skills or requirements. This process helps in identifying the suitability of the applicant for specific roles. The system should highlight skill gaps or overlaps and present the user with a clear skill comparison.
4. **Resume Scoring**: The system must assign a score to each resume based on parameters such as content richness, skill alignment, keyword presence, and document structure. This scoring metric provides candidates with an insight into the quality of their resumes and allows for a standardized comparison across multiple resumes.
5. **Keyword Analysis**: The system should evaluate the resume for relevant keywords that are commonly used in job descriptions. It should identify missing or underused keywords and suggest improvements to enhance visibility in Applicant Tracking Systems (ATS).

**2.2.2 Non-Functional Requirements**

Non-functional requirements define the quality attributes and performance standards that the system must adhere to, ensuring reliability, usability, and efficiency.

1. **Performance:** The system should be fast and responsive. It must analyze each resume—including parsing, scoring, and keyword checking—within 5 seconds under normal conditions. This helps users get quick results without unnecessary waiting. A fast system also keeps users engaged and increases overall satisfaction.
2. **Scalability:** As more users start using the system and more resumes are uploaded, the system should be able to handle the extra load smoothly. It must grow easily without slowing down or crashing. This ensures that the system continues to perform well, even as the number of users increases over time.
3. **Security:** The system must protect users' personal data, including their resumes and contact information. All sensitive information should be safely stored and shared using secure methods. Only authorized users should be allowed to access the data. This builds trust and ensures user privacy is respected at all times.
4. **Usability:** The application must be easy to use, even for people with little technical experience. The interface should be clean, clear, and simple. Helpful tips, clear buttons, and easy-to-follow instructions should guide users through uploading their resumes and viewing the results. A smooth experience encourages users to return.
5. **Maintainability:** The system should be easy to update and fix. If something goes wrong or needs to be improved, it should not take too much time or effort. A well-organized structure and clear documentation will help future developers make changes quickly and keep the system running well over time.
   1. **Tools and Technology**
      1. **Hardware Required**

* **CPU:** Core i5 7th Gen
* **RAM:** 16.0 GB
* **Storage:** 500GB SSD
* Any Laptop or Desktop with Average Configuration.

**2.3.2 Software Required**

* **Operating System:** Windows 10 or Higher
* **Programming Language:** Python 3.12
* **Software:** Visual Studio Code.
* **Libraries:** Flask, nltk, spaCy, PyPDF2, scikit-learn, Pandas, NumPy, and Matplotlib

**2.3.3 Justification of Selected Technology**

**Front End:**

* **HTML**: Used for structuring web pages.
* **CSS**: For styling the Application
* **Jinja2**: Templating engine used with Flask to render dynamic HTML.

**Back End:**

* **Python**: The main programming language for all backend logic.
* **Flask**: The primary web framework handling routing, server logic, and integration with templates.
* **NLTK, SpaCy**: For natural language processing.
* **PyPDF2, pdfminer.six**: For PDF parsing and text extraction.
* **Matplotlib, seaborn, wordcloud**: For data visualization.
* **Other libraries**: Numpy, pandas, scikit-learn, etc, for data processing and analysis.

**Database:**

* **File-based storage (JSON)**: The project does not use a traditional database. Instead, it uses a JSON file to store and retrieve skill data.

**Python**

Python is the primary programming language used to develop the Smart Resume Analyzer. It’s also known for its simplicity, readability, and versatility, making it an ideal choice for implementing natural language processing (NLP) tasks, machine learning, and web development.

* **Ease of Use**: Python has a simple syntax that reduces development time and makes it accessible for both beginners and experienced developers. This allows for quick prototyping and iterative improvements.
* **Libraries and Frameworks**: Python has a vast ecosystem of libraries and frameworks, enabling efficient development. For this project, we utilized several powerful libraries tailored to NLP, machine learning, and web development.
* **Cross-Platform Compatibility**: Python’s cross-platform capabilities allow the Smart Resume Analyzer to work seamlessly across different operating systems, including Windows, macOS, and Linux.
* **Integration with Web Frameworks**: Python integrates well with Flask, a lightweight web framework used for building the front end of the application, making it easy to create dynamic web pages.

**Libraries Used**

**Flask:**

Flask is a lightweight web framework that enables the development of web applications. It provides tools and libraries to build a dynamic web interface for users to interact with the Smart Resume Analyzer. Flask handles the routing, form validation, and integrates with Python code to manage backend processes. Its simplicity makes it an excellent choice for small to medium-sized web applications like ours.

**NLTK (Natural Language Toolkit):**

NLTK is a powerful library for natural language processing that allows us to work with text data. It provides tools for tokenizing, stemming, lemmatization, and part-of-speech tagging. In this project, NLTK is used to process and analyze the textual data from resumes, helping to identify and extract skills, qualifications, and other relevant information. It enables semantic analysis by analyzing text structures and extracting insights from resumes.

**SpyCy:**

SpaCy is another NLP library that is particularly useful for tasks such as named entity recognition (NER), part-of-speech tagging, and dependency parsing. SpaCy's models are pre-trained on large datasets and are highly efficient, which is critical for the real-time resume parsing in the Smart Resume Analyzer. It helps identify key elements in resumes, such as names, education, and technical skills, to match them against job requirements.

**PyPDF2:**

PyPDF2 is used for parsing PDF files, extracting text, and handling PDF-based resume uploads. Since resumes are commonly submitted in PDF format, PyPDF2 allows the system to extract readable text from PDF files, ensuring compatibility with the majority of resume submissions.

**Scikit-Learn:**

Scikit-learn is a machine learning library that provides simple and efficient tools for data mining and data analysis. In this project, scikit-learn is used to implement machine learning algorithms for resume scoring and classification. It helps create a model that can assess the relevance of a resume to a given job description based on various features such as skills, experience, and qualifications.

**Pandas:**

Pandas is a data manipulation and analysis library that is used to structure and analyze the data from resumes. It allows for the efficient handling of structured data and is used to store and process the extracted features from resumes in a DataFrame, making it easy to perform analysis and comparison with job descriptions.

**NumPy:**

NumPy is a library for numerical computations and is particularly useful for handling arrays and matrices. In this project, NumPy is used for handling numerical data related to resume scoring, allowing the system to perform calculations on large datasets efficiently.

**OpenCV (cv2):**

OpenCV, short for Open-Source Computer Vision Library, is a versatile library designed for computer vision tasks. It offers a comprehensive set of functions for image and video processing, object detection, feature extraction, and more.

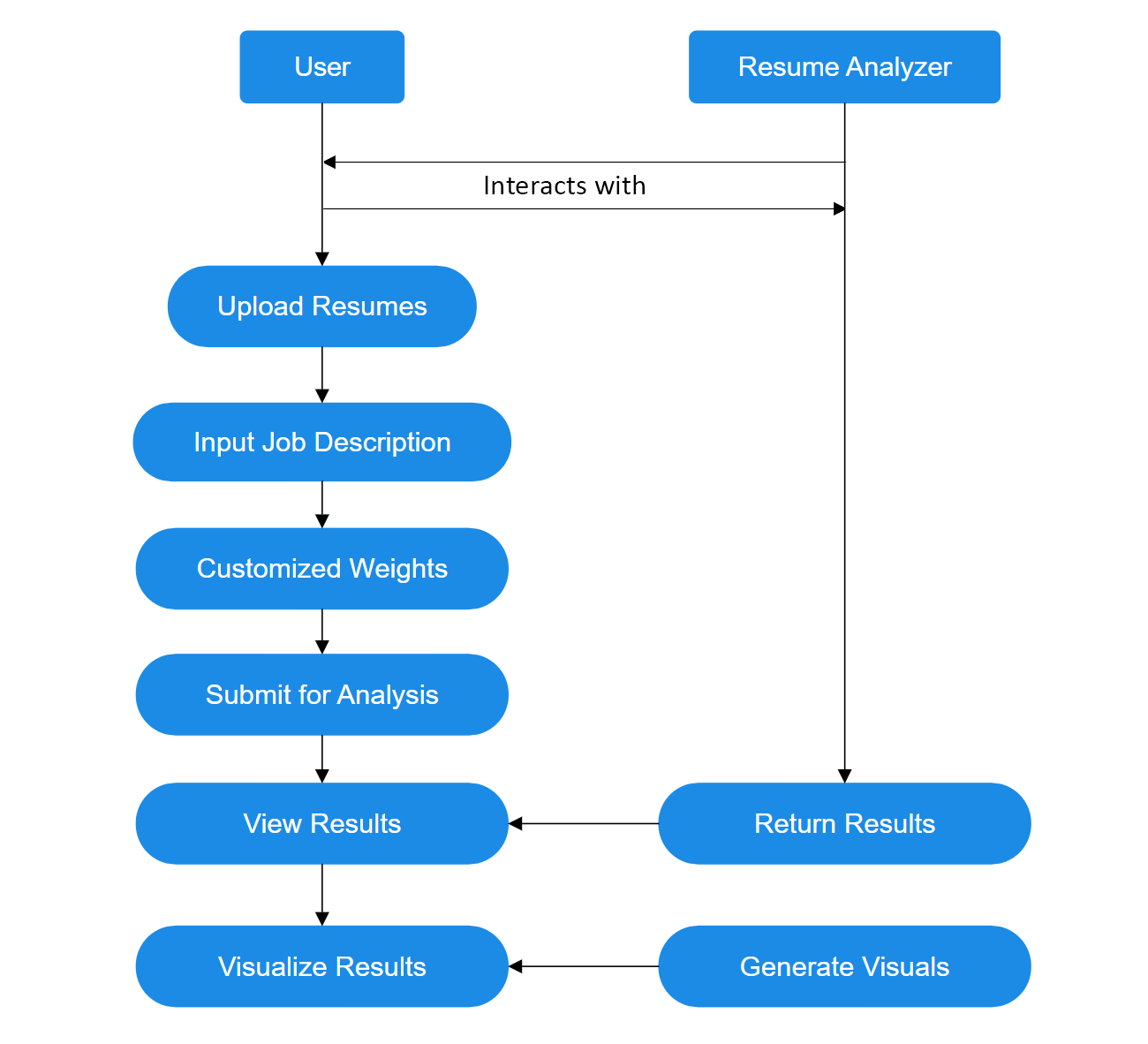
**Matplotlib and Seaborn:**

Matplotlib and Seaborn are data visualization libraries used to generate insightful graphs and charts for the user interface. These visualizations help recruiters and users better understand the analysis results, such as skill gaps, keyword frequency, and resume scoring.

**Word Cloud:**

The word cloud library is used to generate visual representations of frequent terms or keywords in resumes. It helps highlight essential skills or qualifications based on their occurrence in the resumes and provides an easy-to-understand representation of important data.

**2.4 Use Case Diagram**

****

**Figure 2.1: Use Case Diagram**

**2.4.1 Use Case Description**

The use case diagram for the Smart Resume Analyzer project illustrates the interactions between two primary actors: the User and the Resume Analyzer system. This diagram provides a high-level overview of the system’s functionality, clarifying the roles and responsibilities of each actor and the flow of actions within the application.

**User Interaction**

The User represents the recruiter, hiring manager, or any individual seeking to analyze and match resumes against a job description. The user’s journey begins with uploading one or more resume files, typically in PDF format, through the application’s intuitive interface. Next, the user inputs the job description, either by typing it directly or pasting it into the provided text area. To tailor the analysis to specific hiring needs, the user can customize the weighting of different matching criteria—such as required skills, preferred skills, and semantic similarity using interactive sliders, Once the necessary inputs are provided, the user submits the data for analysis. The system then processes the request, and the user waits for the results. Upon completion, the user can view a detailed results dashboard, which includes match scores, skill gap visualizations, and radar charts for each resume. If desired, the user can return to the homepage to initiate a new analysis cycle.

**Resume Analyzer**

The Resume Analyzer is the core backend system responsible for processing and analyzing the data provided by the user. Upon receiving an analysis request, the system extracts text and relevant information from the uploaded resumes using advanced NLP techniques. It also processes the job description to identify required and preferred skills, as well as other key qualifications. The system then matches and scores each resume against the job description, taking into account the user-defined weights for different criteria. After the matching process, the Resume Analyzer generates visualizations—such as skill gap bar charts and skill category radar charts—to help the user quickly interpret the results.

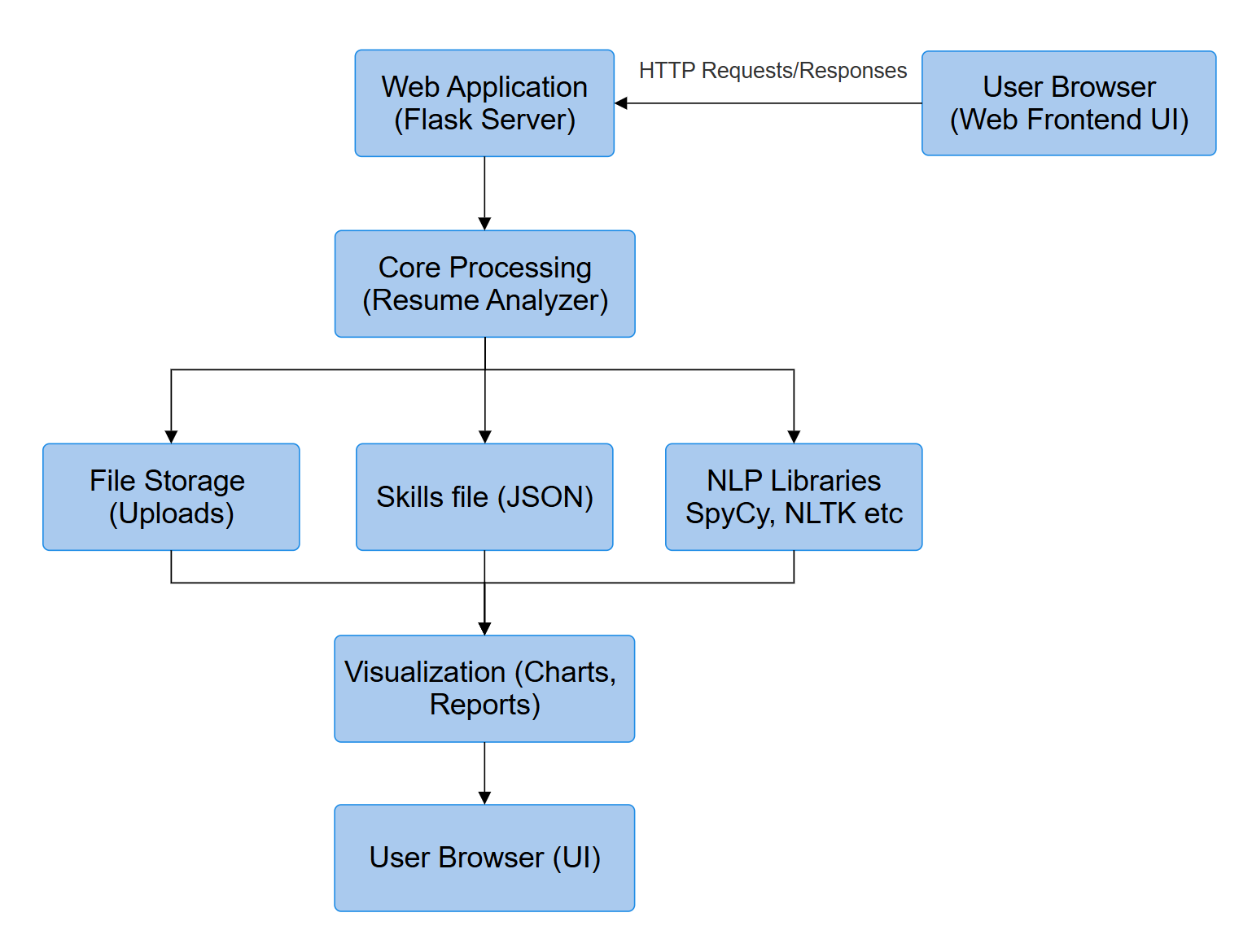
**CHAPTER 3**

**DESIGN AND ARCHITECTURE**



**3.1 System Architecture**

The architecture of the Smart Resume Analyzer is designed to ensure modularity, scalability, and maintainability. It follows a layered structure that separates the user interface, backend processing, storage components, and rendering logic. This enables smooth data flow, efficient processing, and a user-friendly experience.



**Figure 3.1: System Architecture**

**User Browser (Web Frontend UI):**

The user browser serves as the main interface where users upload resumes, input job descriptions, and view results. It provides a clean, user-friendly experience using HTML, CSS, and Jinja2 templates, allowing seamless interaction with the system functionalities.

* Allows users to upload resumes and view analysis results.
* Built using HTML and CSS for user interaction.

**Web Application (Flask Server):**

The Flask server is the core of the web application, handling all incoming HTTP requests. It routes user actions, manages sessions, and coordinates with backend modules to invoke resume processing, ensuring a smooth flow from input submission to output delivery.

* Acts as the backend server handling HTTP requests and routing.
* Manages sessions, uploads, and communication with the core analyzer.

**Core Processing:**

The core processing engine performs the primary logic of the system. It handles resume parsing, skill extraction, and scoring based on NLP techniques. This module ensures accurate analysis and relevance checks between resumes and job descriptions using machine learning algorithms.

* Contains the main logic for parsing, scoring, and skill matching.
* Integrates NLP models to analyze resume content contextually.

**File Storage:**

The File storage is responsible for saving user-uploaded resumes, temporary files, and generated reports. This module ensures secure data handling, easy retrieval, and efficient file management during and after analysis, supporting smooth backend operations without external database dependency.

* Temporarily stores uploaded resumes and generated reports.
* Ensures data is managed securely during processing.

**Skills Database:**

The skills database maintains a curated set of professional skills against which resumes are compared. This structured data helps in identifying skill gaps, overlaps, and matches, supporting job fit analysis and enhancing the accuracy of resume recommendations and scoring.

* Stores a predefined list of job-related skills.
* Used for comparing extracted resume skills against job requirements.

**NLP Libraries (spaCy, NLTK, etc.):**

NLP libraries such as spaCy and NLTK enable advanced language processing. They perform tokenization, lemmatization, entity recognition, and similarity analysis, making the system capable of understanding the context, structure, and meaning of resume content effectively and intelligently.

* Handle text processing, tokenization, and entity recognition.
* Help extract and interpret data from varied resume formats.

**Visualization**

The visualization engine transforms raw data into interactive charts and graphical reports. Using tools like Matplotlib, Seaborn, and WordCloud, it visually communicates analysis results, helping users understand resume performance and suggested improvements in a more engaging format.

* Generates charts and graphical reports from the analyzed data.
* Uses libraries like Matplotlib and WordCloud for visualization.

**Results Renderer**

HTML templates render the final output back to the user in a structured layout. They dynamically present resume scores, keyword analysis, charts, and suggestions, ensuring that users receive a comprehensive, well-formatted view of their resume assessment results.

* Renders the final output in a readable web format.
* Displays scores, suggestions, and graphical summaries to users.

**3.2 Input / Output Design**

This section focuses on how data flows into and out of the Smart Resume Analyzer system. Proper input and output design ensures that users can interact with the system effectively, while the system processes and presents information in a clear, useful manner. Input design involves uploading resumes in PDF format and, optionally, job descriptions, which are then parsed and analyzed using natural language processing techniques. The system extracts relevant skills, qualifications, and experience from the resumes. Output design includes generating structured analysis reports, skill-matching scores, and visual insights such as graphs, making it easier for recruiters to evaluate candidates quickly and accurately.

**3.2.1 Input Design**

The input design defines the way users provide information to the system. For the Smart Resume Analyzer, the primary user input is the resume file, which can be uploaded via a user-friendly web interface. The system supports common formats such as PDF to accommodate most users. Users may also input additional information such as job descriptions or keywords they want to match. The input design ensures that the data provided is structured, relevant, and ready for processing by backend modules. A simple and intuitive file upload form with drag-and-drop or browse options improves usability, while guiding instructions reduce the chances of incorrect submissions.

**3.2.2 Input Validation**

Input validation is a critical step in ensuring that the uploaded data is accurate and in an acceptable format. The system performs various checks such as file type validation (PDF ), file size limit checks, and basic structural validation to confirm that the document contains readable content. This reduces the risk of processing errors and ensures consistent results. If invalid data is detected, the user is promptly notified with a helpful error message, prompting them to correct and resubmit. This validation enhances the reliability and robustness of the system.



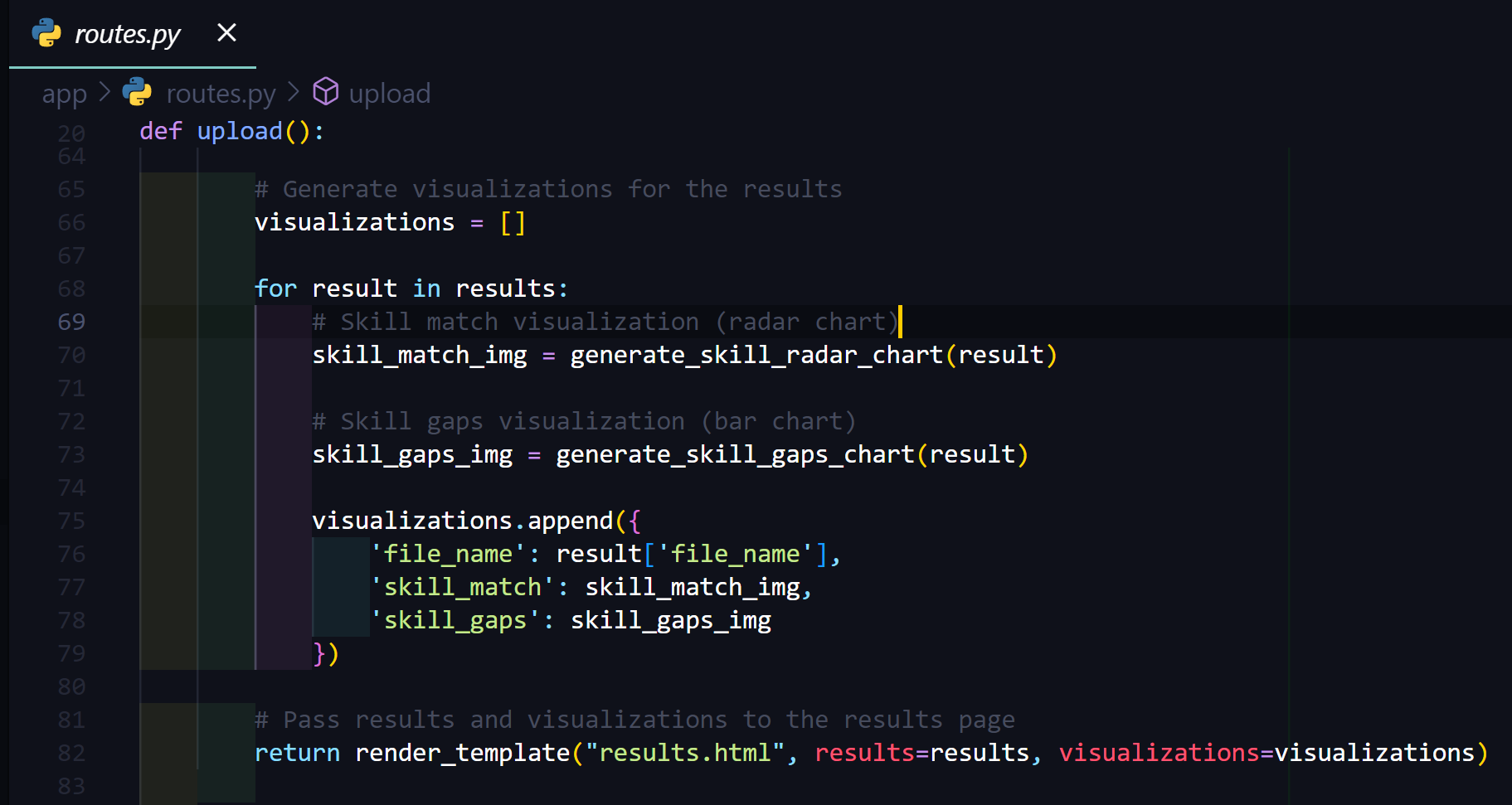
**Figure 3.2: Input Validation of System**

**3.2.3 Output Design**

Output design focuses on what kind of information the system delivers back to the user after analyzing the resume. The Smart Resume Analyzer generates multiple outputs, including the extracted details from the resume, skill match analysis, resume score, keyword suggestions, and an overview of strengths and areas for improvement. These outputs are structured in a way that they are easy to interpret and actionable. Output design ensures the generated results are relevant, well-organized, and helpful for job seekers to improve their resumes effectively.

**3.2.4 Output Presentation**

Output presentation refers to how the analyzed information is displayed to users. The system presents the output in a clean, interactive dashboard format using HTML templates. Key elements such as skill match percentage, keyword density, and scoring breakdown are shown using graphical elements like bar charts and word clouds. This ensures users can both quickly understand their resume performance and keep a documented record for self-improvement or job applications.

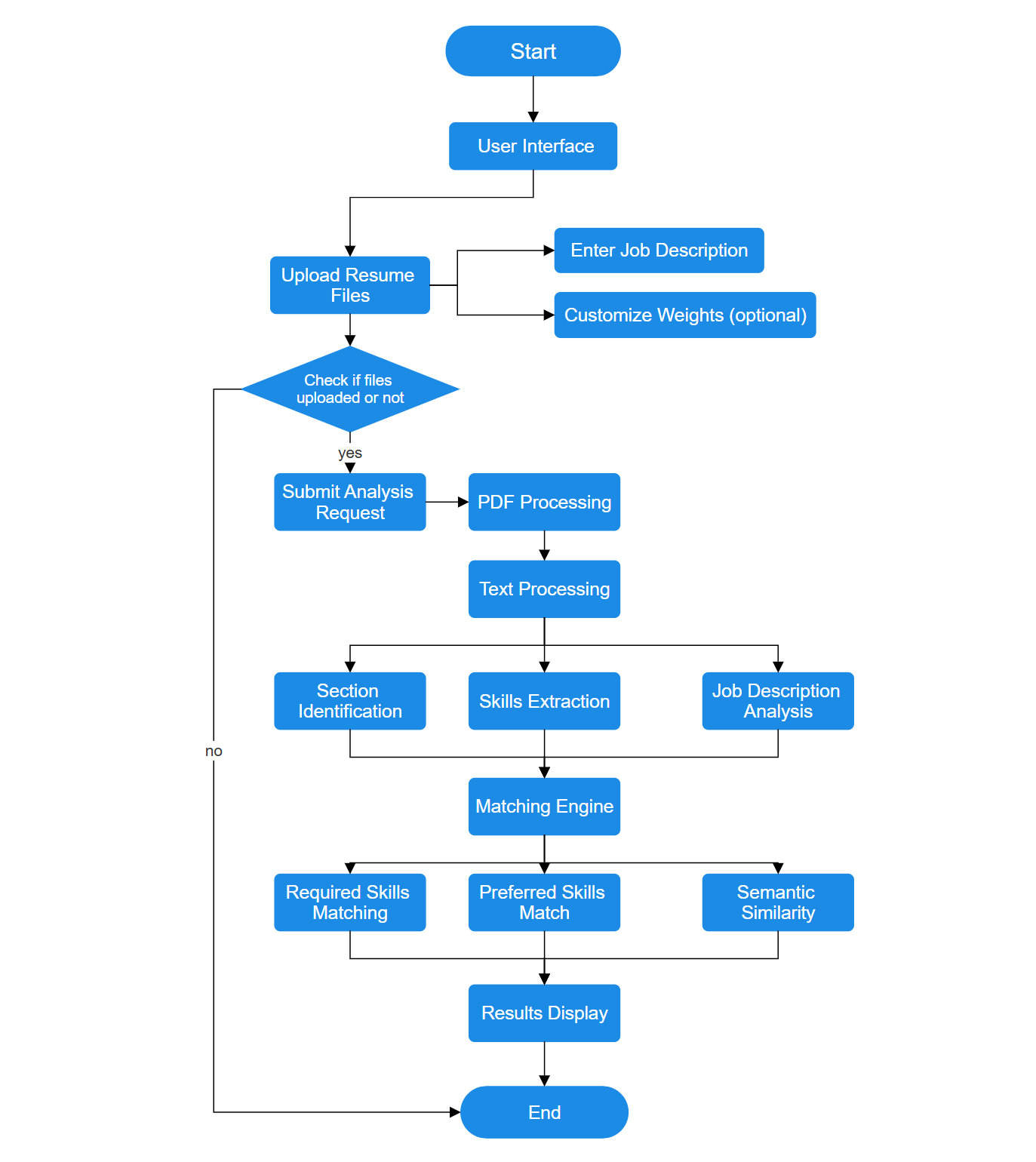


**Figure 3.2: Visualization Design of System**

**3.3 Design Models**

This section presents the design models used in the development of the Smart Resume Analyzer system. It includes essential diagrams that illustrate the system's structure and behavior. Specifically, it covers the flow chart and activity diagram, which help visualize the logical flow of data and the sequence of actions within the system. These models provide a clear understanding of how the system processes resumes and interacts with users effectively.

**3.3.1 Flow Chart**

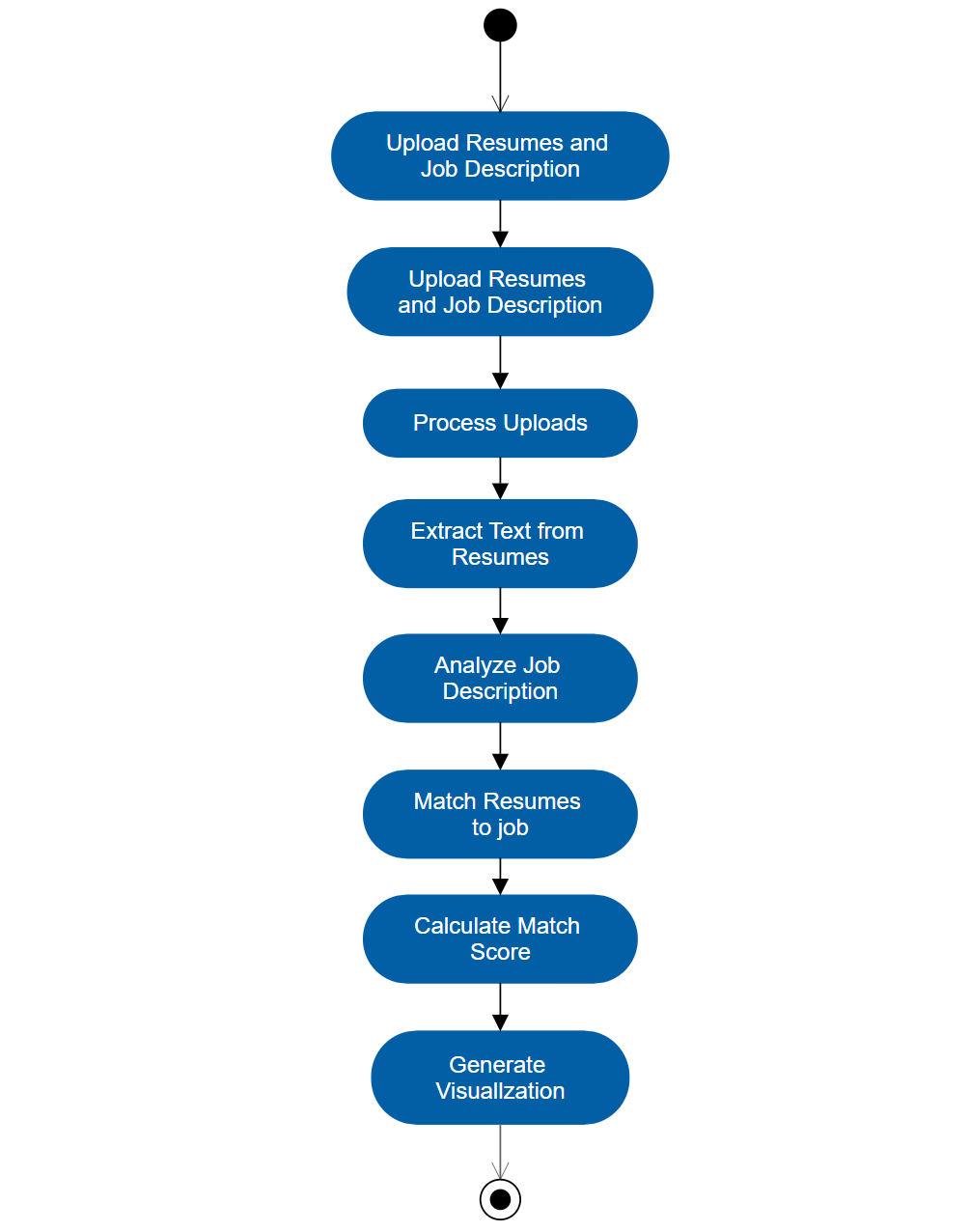
****

**Figure 3.3: Flow Chart of System**

The The flowchart illustrates the Smart Resume Analyzer process: Users upload resumes and job descriptions through the interface, then optionally customize weighting parameters. If files are validated, the system processes PDFs, extracts text, identifies sections, skills, and analyze job requirements. The matching engine compares resumes to jobs using required skills, preferred skills, and semantic similarity algorithms. Results are compiled into comprehensive visualizations and scores, then displayed to users for decision-making. If no files are uploaded, the process ends.

**3.3.3 Activity Diagram**

The activities that are at various levels of abstraction can be provided with a service with the activities that are coordinated, that is shown in the activity diagram.



**Figure 3.4: Activity Diagram of System**

This activity diagram shows the sequential workflow of the Smart Resume Analyzer: users upload documents, the system processes them, extracts text, analyzes job requirements, matches resumes to job criteria, calculates scores, and generates visualizations. Each step builds on the previous one, creating a streamlined pipeline from input to meaningful output.

**CHAPTER 4**

**IMPLEMENTATION**

The implementation phase focuses on converting the system design into a functional Smart Resume Analyzer using Python and Flask. Each module was developed and integrated to handle tasks such as resume parsing, NLP-based skill extraction, and job description analysis. The implementation ensures that user inputs are processed efficiently, and accurate recommendations are generated based on analyzed resume content.

**4.1 Algorithm Used**

The Smart Resume Analyzer employs several sophisticated algorithms to process, analyze, and match resumes to job descriptions. At the core of the system is a multi-stage algorithmic approach that ensures accurate and meaningful results.

The text extraction phase utilizes the PyPDF2 library's content extraction algorithm to convert PDF documents into machine-readable text. This is complemented by regular expression pattern matching algorithms that identify section headers, formatting structures, and key information patterns within unstructured resume text.

For text preprocessing, the system implements Natural Language Processing (NLP) algorithms including tokenization, stop word removal, and lemmatization. These algorithms reduce linguistic variations and normalize text for more effective analysis. The NLTK and spaCy libraries provide the foundation for these operations, with WordNetLemmatizer specifically employed to reduce words to their base forms.

The skill extraction algorithm uses a hybrid approach combining keyword matching against a comprehensive skills database and contextual analysis. This allows the system to identify both explicitly stated skills and those implied through experience descriptions. Skills are then categorized using a classification algorithm that maps them to predefined domains such as programming, databases, and soft skills.

For job description analysis, the system employs regex-based requirement extraction algorithms that differentiate between required and preferred skills based on linguistic patterns. This ensures accurate weighting of different skill types during the matching process.

The matching engine implements three key algorithms:

1. **Required skills matching algorithm** - comparing resume skills against mandatory job requirements
2. **Preferred skills matching algorithm** - evaluating optional but valuable competencies
3. **Semantic similarity algorithm** - using word embeddings from spaCy's en\_core\_web\_md model to calculate cosine similarity between resume and job description vectors

Finally, the weighted scoring algorithm combines these individual match metrics according to customizable weights, producing a comprehensive match score that accurately reflects how well each resume aligns with the job description. The visualization algorithms then transform these quantitative results into intuitive charts and reports.

**4.2 Modules**

* Resume Upload and Parsing
* Job Description Analysis
* NLP Processing
* Skills Extraction and Classification
* Resume-Job Matching
* Visualization and Reporting
* User Interface
* Customizable Weighting

**4.3 Modules Implementation**

The Smart Resume Analyzer is organized into distinct but interconnected modules, each responsible for specific functionality within the application. These modules work together to deliver a seamless user experience while providing sophisticated analysis and comparison capabilities.

**4.3.1 Resume Upload and Parsing**

This module handles the secure upload of resume PDF files through a Flask-based web interface. It implements file validation to ensure only PDF documents within size limits (16MB) are accepted. Once validated, the module uses PyPDF2 to extract text content from each resume, handling potential extraction errors gracefully. The module also manages temporary file storage and cleanup after processing, ensuring security and resource efficiency. Additionally, it provides visual feedback to users during the upload process and validates required inputs before submission.

**4.3.2 Job Description Analysis**

The Job Description Analysis module processes job postings to identify key requirements for matching. Using regular expression patterns, it identifies and extracts required and preferred skills sections from job descriptions. When explicit sections aren't present, the module employs intelligent parsing to infer requirements from context, classifying approximately 70% of identified skills as required and 30% as preferred. The module also extracts education requirements, experience levels, and other qualifications for comprehensive matching. This ensures accurate identification of crucial job requirements that will be weighted more heavily in the final analysis.

**4.3.3 NLP Processing**

This module applies advanced Natural Language Processing techniques to prepare text for meaningful analysis. It implements a multi-stage preprocessing pipeline that includes tokenization, stopword removal, and lemmatization using NLTK and spaCy libraries. The module identifies sentence boundaries and normalizes text to ensure consistent formatting. It also employs Named Entity Recognition to identify organizations, educational institutions, and other entities within resumes. Additionally, it handles language variations and technical terminology common in professional resumes, ensuring accurate interpretation of text content regardless of phrasing or formatting differences across documents.

**4.3.4 Skills Extraction and Classification**

The Skills Extraction module identifies professional skills within resume text using a comprehensive database of technical and soft skills stored in skills\_db.json. It employs pattern matching and context analysis to recognize skill mentions even when exact matches aren't present. Once identified, skills are categorized into domains such as programming, web development, databases, cloud technologies, and soft skills. The module maintains relationships between similar skills and accommodates various naming conventions (e.g., "React" vs "React.js"). This categorization enables more meaningful visualization and comparison, helping users understand a candidate's expertise distribution across different professional areas.

**4.3.5 Resume-Job Matching**

This core module compares extracted resume data with job requirements using a multi-faceted approach. It calculates three primary match metrics: required skills match (identifying how many must-have skills the candidate possesses), preferred skills match (evaluating desired but optional competencies), and semantic similarity (measuring overall contextual relevance between resume and job description). The module handles partial matches and related skills intelligently, recognizing when similar technologies indicate relevant experience. It also identifies skill gaps - required and preferred qualifications missing from the resume - providing actionable insights for both recruiters and job seekers.

**4.3.6 Visualization and Reporting**

The Visualization module transforms analysis results into intuitive graphical representations using Matplotlib. It generates radar charts displaying skill category distribution across domains like programming, databases, and soft skills. For skill gaps, it creates color-coded bar charts that distinguish between missing required and preferred skills. The module also produces downloadable PDF reports via FPDF, containing comprehensive analysis results, match scores, and recommendations. These visualizations help users quickly interpret complex matching data, identify strengths and weaknesses, and make informed decisions about candidate suitability or resume improvements needed for specific positions.

**4.3.7 User Interface**

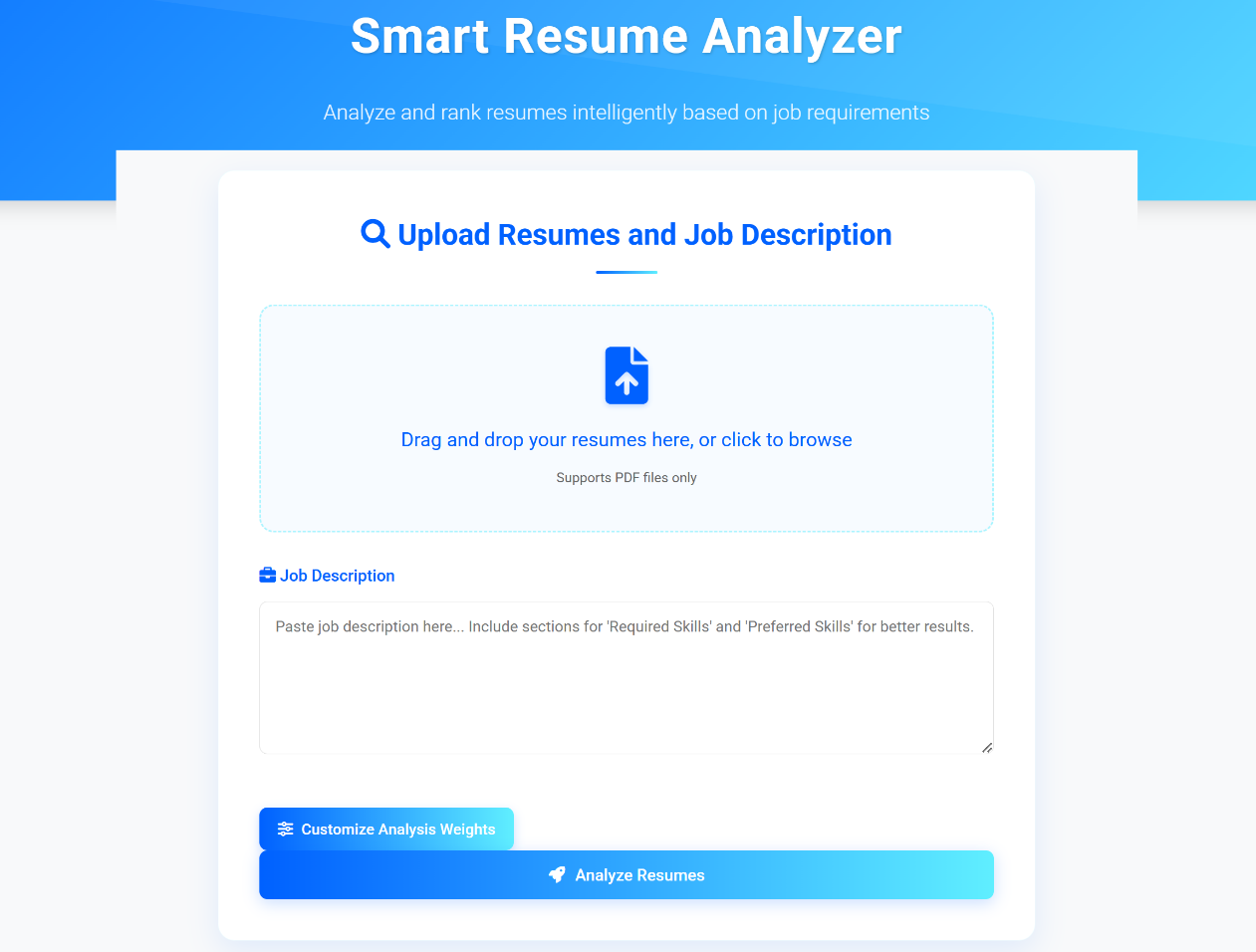
The User Interface module provides an intuitive, responsive web interface built with HTML, CSS, and JavaScript. It features a modern drag-and-drop file upload area for resumes, a text input area for job descriptions, and interactive controls for customizing analysis parameters. The module implements real-time validation to ensure proper inputs before submission and provides loading indicators during processing. For results, it presents a clean, tabbed interface organizing scores, skills, education, experience, and visualizations in a logical hierarchy. The responsive design works across devices, making the application accessible to users regardless of screen size or device type.

**4.3.8 Customizable Weights**

This module allows users to personalize the analysis by adjusting the importance of different matching criteria. Through an interactive slider interface, users can modify the weights assigned to required skills (default 60%), preferred skills (default 20%), and semantic similarity (default 20%). The module validates that weights sum to 100% and applies these customizations to the final score calculation.

**4.3 Homepage/Landing Page**

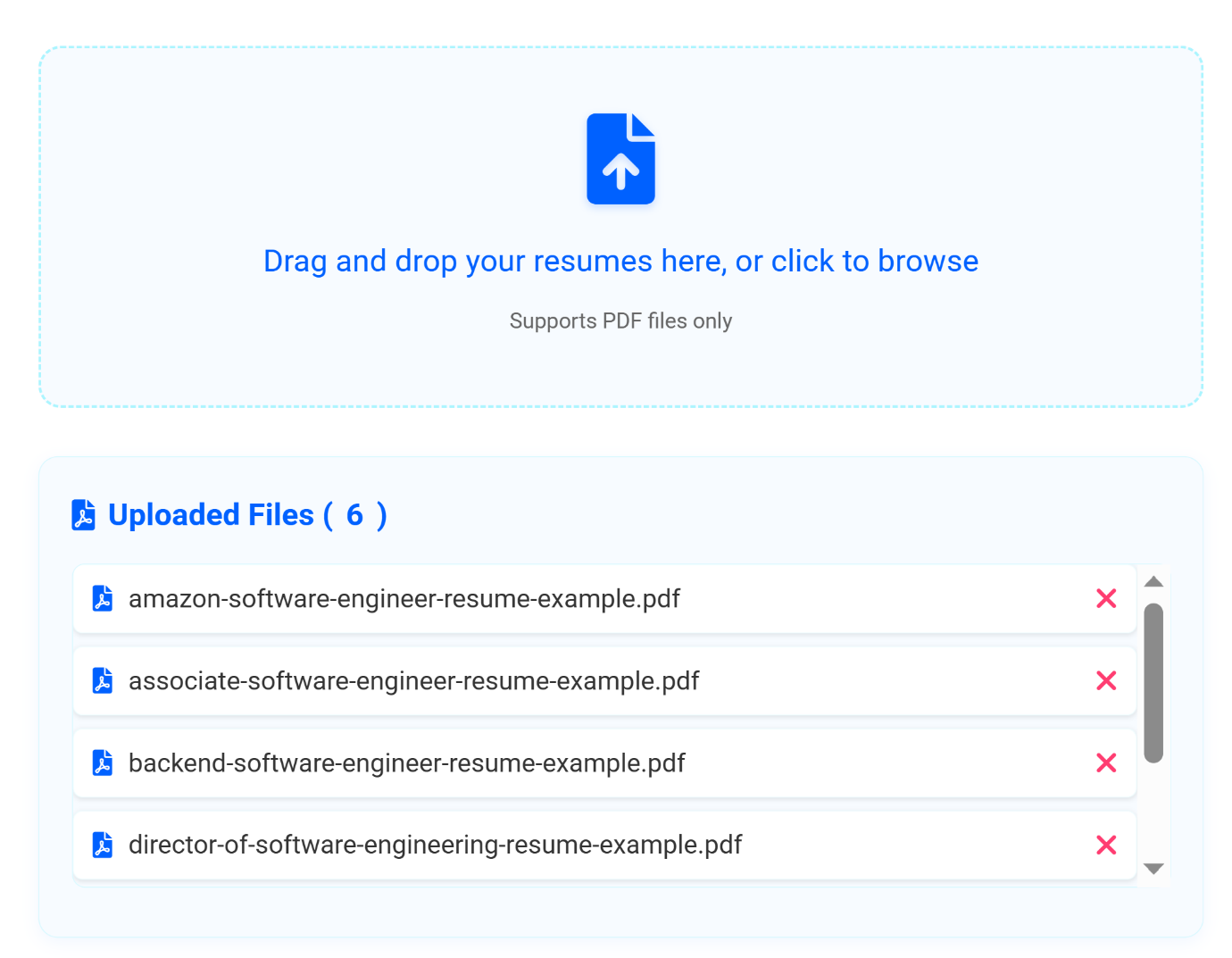
Figure 4.1 displays the Smart Resume Analyzer's homepage, featuring a modern, intuitive user interface designed for easy navigation. The page showcases a clean layout with a prominent hero section containing the application title and a brief description of its functionality. Below, users find the main action area with a visually distinct drag-and-drop zone for resume uploads. The interface employs a blue gradient colour scheme creating visual hierarchy and drawing attention to key interactive elements. Clear instructional text guides users through the initial steps of the analysis process, while responsive design elements ensure accessibility across devices. This thoughtfully designed entry point establishes user confidence and simplifies the resume analysis workflow.

****

**Figure 4.1: Homepage/Landing Page**

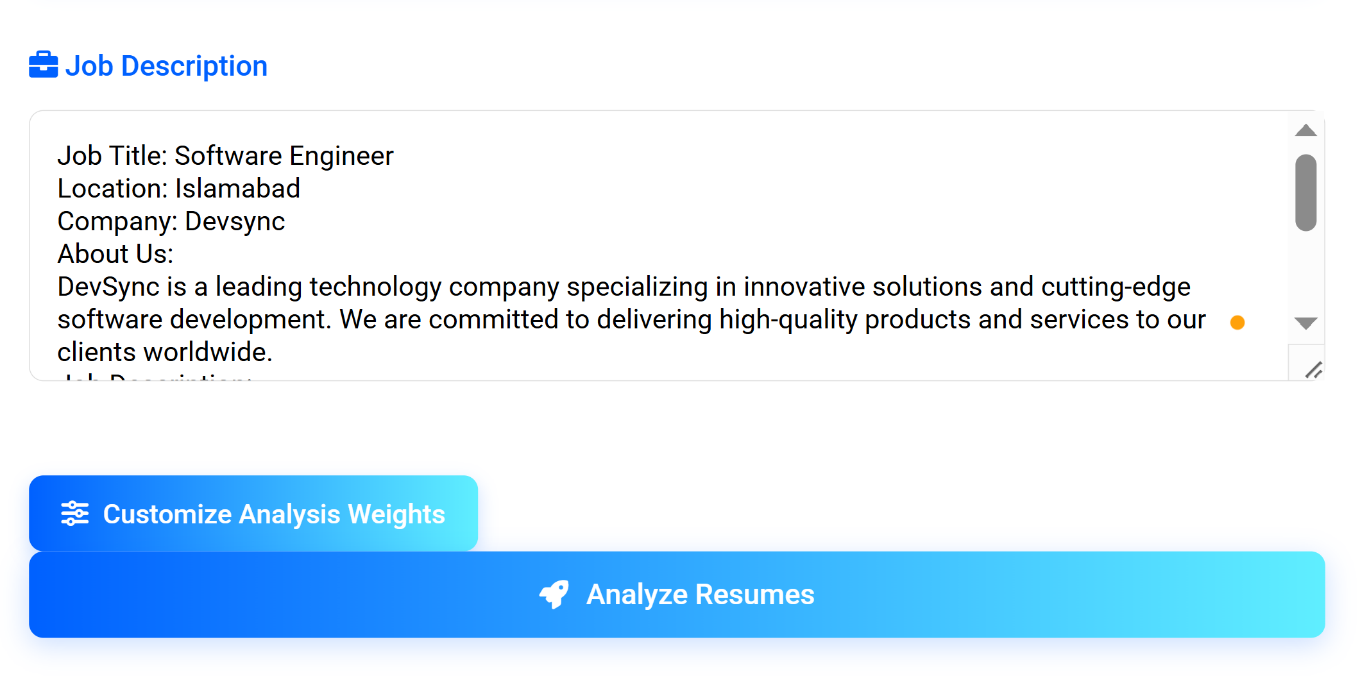
**4.4 Resume Upload Interface**

Figure 4.2 illustrates the resume upload interface, highlighting the application's user-friendly file handling capabilities. The interactive drag-and-drop box prominently displays file selection instructions and supported format information (PDF files only). Once files are uploaded, the interface transforms to show a dynamic list of selected resumes with file names and removal options, providing users with clear visual confirmation of their selections. The design incorporates subtle visual cues including hover effects and transition animations that enhance usability. Below the upload area sits the job description text field, strategically positioned to guide users through the logical sequence of actions. This interface element effectively combines visual appeal with functional efficiency to streamline the document submission process.

****

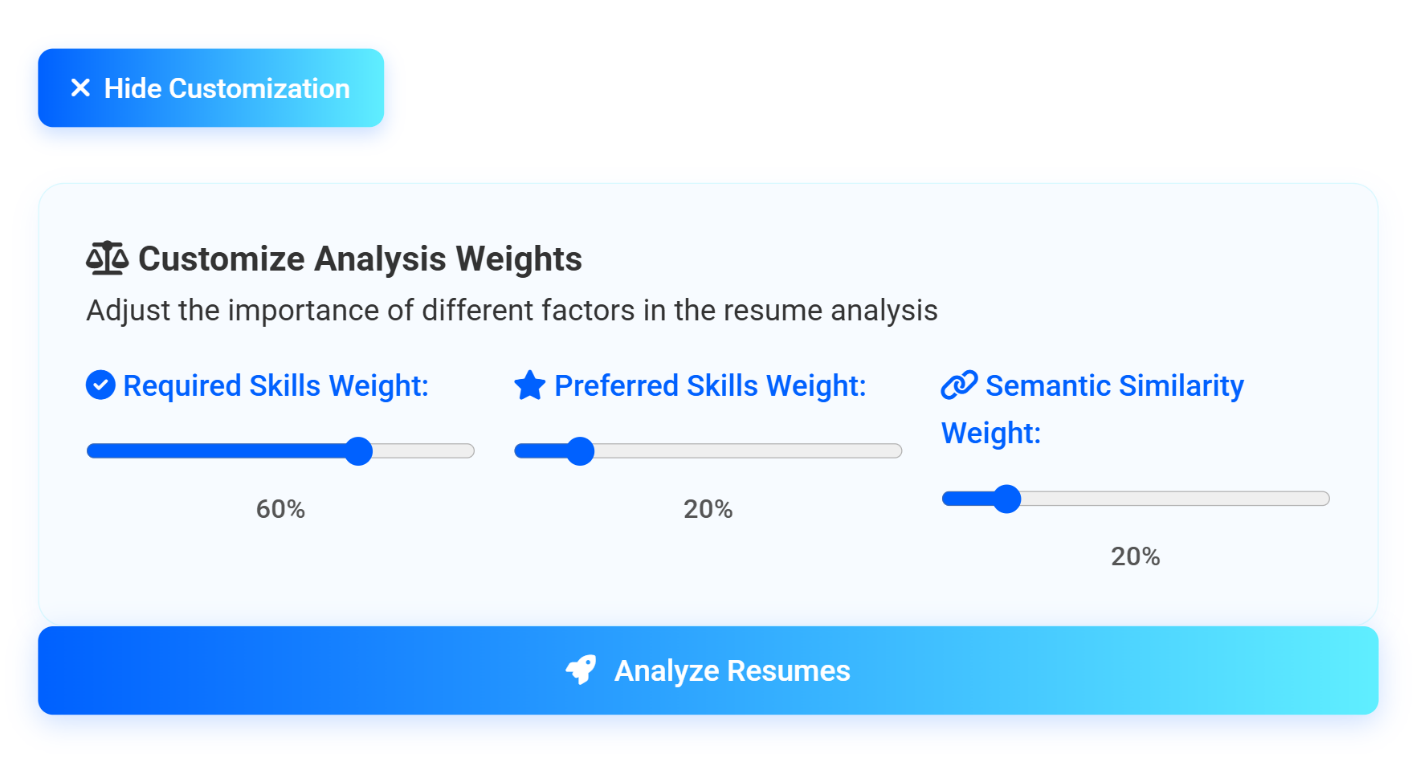
**Figure 4.2: Resume Upload Interface**

**4.5 Job Description Input Form**

****

**Figure 4.3: Job Description Input Form**

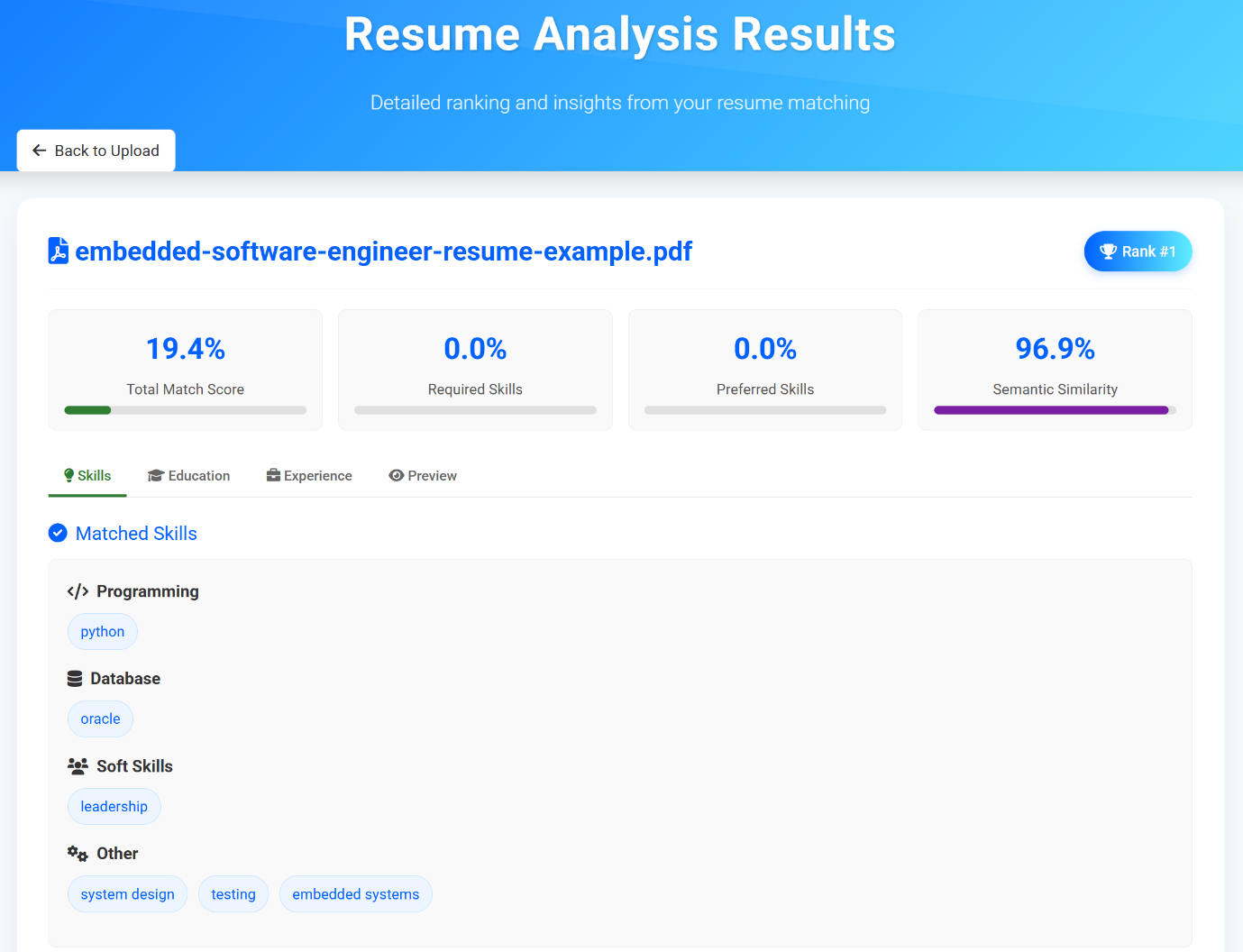
**4.5 Weight Customization Panel:**

****

**Figure 4.4: Weight Customization Panel**

**4.6 Results Dashboard Overview**

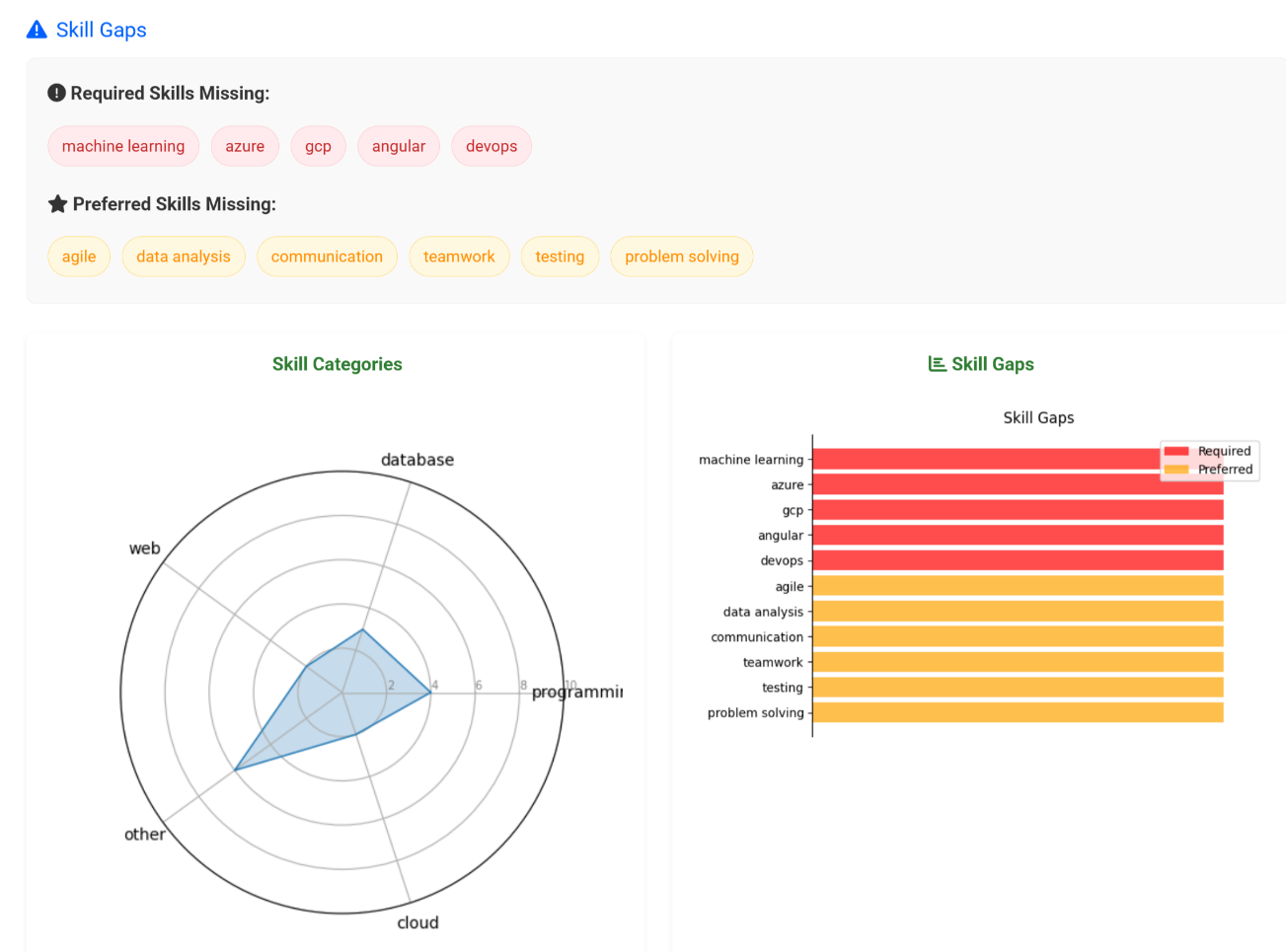
Figure 4.6 showcases the comprehensive results dashboard, the central hub for displaying resume analysis outcomes. The interface presents a clean, card-based layout organizing results in descending order of match scores. Each resume card features prominent match percentages with visual progress indicators, creating immediate visual hierarchy. The dashboard employs a tabbed navigation system that elegantly segments information into categories including skills, education, experience, and visualizations. Color-coding effectively distinguishes between match types (required, preferred, semantic), while the responsive design ensures optimal viewing across devices. This thoughtfully structured display enables quick identification of top candidates.

****

**Figure 4.5: Weight Customization Panel**

**4.7 Visualization of Skill Analysis and Gaps**

Figure 4.6 showcases the comprehensive results dashboard, the central hub for displaying resume analysis outcomes. The interface presents a clean, card-based layout organizing results in descending order of match scores. Each resume card features prominent match percentages with visual progress indicators, creating immediate visual hierarchy. The dashboard employs a tabbed navigation system that elegantly segments information into categories including skills, education, experience, and visualizations. Color-coding effectively distinguishes between match types (required, preferred, semantic), while the responsive design ensures optimal viewing across devices. This thoughtfully structured display enables quick identification of top candidates.



**Figure 4.6: Figure 4.5: Visualization of Skill Analysis and Gaps**



**CHAPTER 5**

**TESTING AND EVALUATION**

**5.1 Testing Dataset Evaluation**

The testing and evaluation phase of the Smart Resume Analyzer project was crucial to ensure the application's reliability, accuracy, and usability. A comprehensive testing strategy was implemented to validate all functional components, from file uploads to resume analysis and result presentation. This chapter details the methodologies employed to test the application across different scenarios and user interactions. Various testing approaches including unit testing, integration testing, system testing, and user acceptance testing were conducted to verify the system's conformance to requirements. Performance metrics were established to evaluate the application's effectiveness in matching resumes to job descriptions, with particular focus on the accuracy of skill extraction, match scoring, and the overall user experience. The findings from these evaluations informed refinements to the application's algorithms and interface.

**5.1 Unit Testing**

Unit testing formed the foundation of our testing strategy, focusing on validating the functionality of individual components in isolation. This ensured that each module performed correctly before integration with other system elements.

**File Upload Component**

* **Input Validation Tests**: Verified that only PDF files were accepted
* **Size Limit Tests**: Confirmed enforcement of the 16MB file size restriction
* **Error Handling**: Validated appropriate error messages for invalid file types
* **Edge Cases**: Tested with empty files, corrupted PDFs, and password-protected documents

**Text Extraction Module**

* **PDF Parsing Tests**: Evaluated extraction accuracy across different PDF structures
* **Character Encoding**: Verified proper handling of special characters and symbols
* **Section Identification**: Tested the accuracy of resume section detection algorithm
* **Robustness Tests**: Verified handling of scanned PDFs versus digitally created documents

**Skill Extraction Component**

* **Pattern Recognition**: Assessed accuracy in identifying technical skills from text
* **Skills Database**: Validated matching against the skills database
* **Classification Tests**: Verified correct categorization of identified skills
* **Context Analysis**: Tested extraction of skills mentioned in different contexts

**Matching Algorithm**

* **Scoring Function**: Verified mathematical accuracy of match score calculations
* **Weight Application**: Confirmed correct application of customized weights
* **Semantic Analysis**: Tested vector-based similarity measurements
* **Edge Cases**: Validated behavior with empty resumes or job descriptions

**5.2 Integration Testing**

Integration testing examined the interactions between connected components to ensure seamless data flow and proper functionality of the entire system. This phase was critical for identifying interface issues between modules.

**Upload-to-Processing Pipeline**

* **Data Transfer:** Verified that uploaded files were correctly passed to the processing module
* **Temporary Storage:** Confirmed proper file handling and cleanup after processing
* **Session Management:** Tested persistence of uploaded files during the analysis process
* **Multiple File Handling:** Validated batch processing of multiple resumes

**Text Processing to Analysis Flow**

* **Data Transformation:** Verified correct preprocessing of extracted text
* **NLP Pipeline Integration:** Tested the complete text normalization, tokenization, and lemmatization sequence
* **Section to Skill Mapping:** Confirmed accurate transfer of identified sections to the skill extraction module
* **Error Propagation:** Tested how errors in one module affected downstream components

**Analysis to Visualization Chain**

* **Data Formatting:** Verified proper formatting of analysis results for visualization
* **Chart Generation:** Tested accurate creation of radar charts and bar charts from analysis data
* **Results Rendering:** Validated proper display of all analysis components in the web interface

**Frontend-Backend Communication**

* **API Endpoints:** Tested all interfaces between frontend and backend components
* **Response Handling:** Verified proper handling of server responses and error conditions
* **Asynchronous Operations:** Confirmed loading states during processing
* **Data Consistency:** Validated that data remained consistent across all system layers

**CHAPTER 6**

**CONCLUSION AND FUTURE WORK**

**6.1 Conclusion:**

**6.2 Future Work:**

**References**

1. Deepa, Y.G., Sindhu, A., Shruthi, A. and Neha, B., 2025. Automated Resume Parsing: A Review of Techniques, Challenges and Future Directions.
2. Prashanth, V.J., Gopinath, S., Udith, S. and Kavitha, C.R., 2024, July. Resume Analyzer and Skill Enhancement Recommender System. In 2024 Asia Pacific Conference on Innovation in Technology (APCIT) (pp. 1-6). IEEE.
3. Susan, S., Sharma, M. and Choudhary, G., 2024. Uniqueness meets Semantics: A Novel Semantically Meaningful Bag-of-Words Approach for Matching Resumes to Job Profiles. *Inteligencia Artificial*, *27*(74), pp.117-132.
4. Patil, S.M. and Shinde, V.D., 2024, February. Analyzing the Effects of K-Means and Random Forest Algorithms on Resume Classification through Manipulation of Cluster Sizes and Datasets. In *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 761-766). IEEE.
5. Gulati, V., Gupta, I., Firdous, F. and Narwal, R., 2024. Resume Analyzer Using Natural Language Processing (NLP).
6. Thangaramya, K., Logeswari, G., Gajendran, S., Roselind, J.D. and Ahirwar, N., 2024, May. Automated Resume Parsing and Ranking using Natural Language Processing. In *2024 3rd International Conference on Artificial Intelligence for Internet of Things (AIIoT)* (pp. 1-6). IEEE.
7. Sharif, H.E., Sharif, M.E., Al-Dilami, R., Al-Khawlani, D.M., Khushafah, A.A., Shadad, Y.H. and Zahary, A.T., 2024, November. Smart Assistant to Choose the Best Resume Using Natural Language Processing and Machine Learning. In *2024 1st International Conference on Emerging Technologies for Dependable Internet of Things (ICETI)* (pp. 1-6). IEEE.
8. Mohamed, O., Sameh, D., Mohamed, H., Ashraf, M., Kamal, R. and Kandil, Y., 2024, November. ResuFit: Precision CV Matching System. In *2024 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)* (pp. 363-367). IEEE.
9. Pabalkar, S., Patel, P., Choudhary, R., Panoch, V., Yadav, S. and Ghogale, H., 2024, October. Resume Analyzer Using Natural Language Processing (NLP). In *2024 International Conference on Intelligent Systems and Advanced Applications (ICISAA)* (pp. 1-6). IEEE.
10. Mirajkar, R.R., Shinde, G.R., Shelke, P.M., Yadav, S., Shinde, B. and Shinde, G., 2024, December. Skill Recommendation System and Resume Analysis using AI. In *2024 IEEE Pune Section International Conference (PuneCon)* (pp. 1-3). IEEE.
11. Bhavsar, N., Singla, S., Sharma, D., Dey, R. and Maji, S., 2024, December. SATYA: Smart AI-driven Talent Yield Analyzer for Resume Screening and Recommendation Evaluation. In *2024 4th International Conference on Robotics, Automation and Artificial Intelligence (RAAI)* (pp. 221-225). IEEE.
12. Patel, H., Patel, A., Patel, D., Shah, D., Tailor, S. and Deb, N., 2024, July. Streamlining Talent Acquisition by Gaining Insights from Resumes. In *International Symposium on Applied Computing for Software and Smart Systems* (pp. 17-28). Singapore: Springer Nature Singapore.
13. Ganga, V.S., Kranthi, S., Regalla, S.N.D. and Vissakodeti, L.L., 2024, April. Skill Extractor in Resumes using Machine Learning and Streamlit. In *2024 1st International Conference on Trends in Engineering Systems and Technologies (ICTEST)* (pp. 1-5). IEEE.
14. Kamkar, S.A., Sanjay, S., Vaishnavi, P.S. and Chaitra, M., 2024, November. Resume Parser and Job Description Matcher. In *2024 8th International Conference on Computational System and Information Technology for Sustainable Solutions (CSITSS)* (pp. 1-6). IEEE.
15. Sarveshwaran, R., Karthikeyan, S., Cruz, M.V., Shreyanth, S., Niveditha, S. and Rajesh, P.K., 2024. NLP-Based AI-Driven Resume Screening. In *Proceedings of Ninth International Congress on Information and Communication Technology: ICICT 2024, London, Volume 2* (Vol. 2, p. 359). Springer Nature.
16. Chandak, A.V., Pandey, H., Rushiya, G. and Sharma, H., 2024, February. Resume Parser and Job Recommendation System using Machine Learning. In *2024 International Conference on Emerging Systems and Intelligent Computing (ESIC)* (pp. 157-162). IEEE.