A Project Report on

Automated Resume Screening using AI-Powered Parsing and Job suggestion using K-NN with Cosine Similarity

Submitted in partial fulfillment for award of

Bachelor of Technology

Degree

in

Computer Science and Engineering

By

M. Priyanka (Y21ACS508) **K.L.A. Varshini(Y21ACS483)**

M. Harshavardhan(Y21ACS514) **M. Rahul Rayudu(Y21ACS513)**

Under the guidance of

Mr.P.Nanda Kishore, **M.Tech**

Assistant Professor

Department of Computer Science and Engineering

Bapatla Engineering College

(Autonomous)

(Affiliated to Acharya Nagarjuna University)

BAPATLA – 522 102, Andhra Pradesh, INDIA

2024-2025

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project report entitled **Automated Resume Screening using AI-Powered Parsing and Job Recommendation using K-NN cosine Similarity** that is being submitted by M.Priyanka(Y21ACS508), K.Amrutha (Y21ACS483), M.Harshavardhan(Y21ACS514), M.Rahul Rayudu(Y21ACS513) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science & Engineering to the Acharya Nagarjuna University is a record of bonafide work carried out by them under our guidance and supervision.

Date:

Signature of the Guide
P. Nanda Kishore
Assitant Professor

Signature of the HOD
Dr. M. Rajesh Babu
Assoc. Prof. & Head

DECLARATION

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

M.Priyanka(Y21ACS508)

K. Amrutha(Y21ACS483)

M.Harshavardhan(Y21ACS514)

M.Rahul Rayudu(Y21ACS513)

Acknowledgement

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide **P. Nanda Kishore**, Assistant Professor, Department of CSE, for his valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Dr. M. Rajesh Babu**, Assoc. Prof. & Head of the Dept. for extending his cooperation and providing the required resources.

We would like to thank our beloved Principal **Dr. N. Rama Devi** for providing the online resources and other facilities to carry out this work.

We would like to express our sincere thanks to our project coordinator **Dr. P. Pardhasaradhi,** Prof. Dept. of CSE for his helpful suggestions in presenting this document.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly for their cooperation and encouragement.

M. Priyanka(Y21ACS508)

K. Amrutha(Y21ACS483)

M. Harshavardhan(Y21ACS514)

M. Rahul Rayudu(Y21ACS513)

ABSTRACT

"Automated Resume Screening using AI-Powered Parsing and Job suggestion using K-NN with Cosine Similarity" is a comprehensive system that analyzes resumes and provides tailored job recommendations. This application employs natural language processing to extract key information from resumes and leverages K-Nearest Neighbors algorithm with cosine similarity to match candidates with suitable job opportunities. Users simply upload their resume, and the system automatically analyzes it, extracting Jobs and qualifications. Based on this analysis, the system recommends relevant jobs from its database, ranking them according to match confidence. It also provides course recommendations to help users develop Jobs needed for their target roles and offers job market insights through an interactive dashboard. This technology aims to revolutionize the job search process by implementing artificial intelligence in the recruitment sector, benefiting both job seekers and employers. The system can be used by freshers, graduates, or experienced professionals to improve their job prospects through targeted recommendations and skill development guidance.

Table Of Contents

List of Figures

Introduction

In today’s fast-paced job market, companies often receive hundreds of resumes for each opening, making it tough to manually screen and match candidates efficiently. To solve this, our project introduces an automated resume screening system that uses AI-powered parsing to extract key details like skills, experience, and education from resumes. Instead of relying on manual filtering, the system intelligently matches candidates to suitable job roles using the K-Nearest Neighbours (K-NN) algorithm combined with Cosine Similarity, which helps measure how closely a candidate’s profile aligns with job descriptions. This approach not only speeds up the hiring process but also ensures better matching accuracy, helping recruiters find the right talent faster and giving job seekers a fair chance at the roles that fit them best.

1.1 Background

Unemployment is still a major problem in today's changing labor market, with many people finding it difficult to find acceptable jobs. Lack of knowledge about industry trends, resume optimization, and necessary certifications is a major barrier. The project intends to use artificial intelligence (AI) to transform the hiring process in order to address this. This system evaluates resumes using Natural Language Processing (NLP) to offer customized suggestions on fields, technologies, courses, and resume-writing strategies, ultimately improving candidates' employability. This AI-powered solution, which is intended for both new hires and seasoned professionals, aims to close the gap between employers and job seekers while encouraging accuracy and efficiency in the hiring process. This is a revolutionary step toward using AI for meaningful applications in the job recruitment sector.

1.1.1 Purpose

The main purpose of building this project is to create a smart technology for corporate hiring world. Nowadays there are thousands of people are unemployed or they are finding the jobs. Our Aim is to provide a smart system that can give the perfect recommendation based on the resume of user, It will give the recommendation of Job working industry, tools and Technology, courses, and resume writing techniques. So with this help user can create a better resume so it will increase the chances of getting a job in good company.

1.1.2 Current Situation

In current situation there are thousands of unemployed people finding the job but they are can't able to select in the companies, the reason is they don't know the trend of current technology and fields. Maybe they are lacking with the Jobs of resume writing. It is chances that they didn't have done any particular certifications in particular field. So, our ideas is to implement Artificial Intelligence in this field. AI Technology is playing a very vital role in changing our life in many ways. There are many sectors which benefited after implementing AI technology in it. Now it's time for evolution in very field. With the use of Artificial Intelligence every field is growing widely. With this project our aim is to use AI in job sectors. This system can be use by any fresher, graduate or experienced people. Our future aim to create same module for companies oriented also, so with this application companies can directly shortlist the people using AI. We are using the core concepts of Natural Language Processing for analyzing the resume. This is just first step for implementing Artificial Intelligence in the field of Job Recruitment sector.

1.1.3 Tools and Technologies

- Python Programming
- NLTK and Pyparser for NLP
- Streamlit for Backend
- PDFMiner for extracting PDF
- Base64 for PDF displaying
- Pillow
- Numpy
- Pycharm IDE
- BS4 and requests for Web Scraping

Literature Review

According to [1], In this paper authors have explained Automated Resume Screening System using Natural Language Processing. They have used stemming, lemmatization, POS Tagging, Chunking and Tokenization. They have used Cosine Similarity to get the ranking of the Candidates. In [2], Paper explained different techniques for the Resume Parsing using Natural Language Processing. They have used OCR to fetch to text from the resume. They are using the Tokens and Semantic Analysis to pre-process the text of resume. They are using Ranking algorithms to select the candidate. According to [3], Paper explained Resume Parser with Natural language Processing technique. They have used Lexical analysis, Tokens, Syntactic Analysis, and parse tree for the processing.

They are using Json format to fetch the output. According to [4] According to paper, Authors are using Stop words removal, Stemming, Lemmatization to fetch the data. In machine learning they are using Random Forest, Naïve Bayes, Logistic regression and Linear Support Vector Classifier.

Recently, research in the fields of artificial intelligence has seen a variety of applications in HR [5–7]. The rise of e-recruitment has resulted in a considerable shift in the hiring process. Several research offered several methodologies with the goal of automating candidate screening [8]. Bhor et al. [9] presented a comprehensive survey that includes a job site where employees and applicants can upload their resumes for any job opening. The NLP technique graded employee resumes based on the required skill set of the organization and the Jobs listed in the person's resume. Roy et el. [10] discussed an automated supervised machine learning-based system that recommends appropriate applicant resumes to HR based on the job description. Content-based with cosine similarity and KNN were applied as a recommendation model. The model takes CVs in CSV format, and uses the genism library.

Arunasish and colleagues' "SCREENER" model [11] scans resumes for essential features of candidate profiles such as Jobs, experience with each skill, education, and past work experience and parses them using Apache Tika. Daryani et al. [12] presented an automated resume screening system that employs NLP to extract relevant information from unstructured resumes, such as Jobs, education, and experience, and then tokenizes each

application to construct a summary form. After deleting unnecessary data, the applicants were rated using a content-based recommendation that matches the extracted resume.

S. M. S. F. Shovon et al.attributes to the job need using the Vector Space Model and cosine similarity. Using a cosine similarity method based on verbal job description, Agarwal and Senthilkumar [13] created an autonomous machine learning model that assisted in recommending eligible candidates' resumes to recruiters. Al-Otaibi and Ykhlef [14] used three similarity measures to rank candidates for a job position. They described a survey-based job recommender system in which content-based filtering works effectively with implicit feedback when explicit rating is impossible.

Resumes were classified using the KNN algorithm, and cosine similarity was used to determine how close the candidate's resume was to the job description. Rajath et al. [15] employed KNN Algorithm to categorize resumes according to their respective categories, while cosine similarity was used to determine how similar is the candidate's CV to the job description. Kiran et al. [16] used data mining techniques to discover patterns that can aid in profession and skill suggestions. Apriori algorithm was to create association rules and recommendations. Gupta and Garg [17] used content-based profile matching to recommend jobs to aspiring candidates. Kumar [18] used Naive Bayes to assess resumes. The accepted candidates were contacted with an offer message whereas the rejected candidates receive appropriate feedbacks.

Wang et al. [19] introduced a knowledge graph and BERT-based keyword-based search engine for recruitment and staffing. Gugnani and Misra [20] proposed a system for job suggestion in which Jobs were extracted using NLP techniques and Doc2VecModel and similarity approaches were utilized to discover and in implicit abilities for each job description that were not explicitly listed in the original job description. A generalizable ensemble method for extracting abilities from unstructured material in resumes and JDs was also proposed in the paper.

Duan et al. [21] proposed a personalized resume recommendation algorithm. They extracted the features and used K-means++ to cluster position, build text vectors, and combine TF-IDF and part-of-speech weights to recommend resumes using cosine similarity. Moreover, they proved that the proposed algorithm's recommendation result

outperforms the word frequency vector. Appadoo and colleagues presented the development of JobFit, a job recommendation system that uses various ML models, a collaborative filtering recommender system, and past data to predict the best fit candidate for a job. The proposed job recommendation system takes the job requirement and applicant profile as inputs and outputs a JobFit score indicating how to fit each applicant for the particular job.

Kurdija et al. described a CV recommender that used a content-based filtering system and divided it into two properties - the ability to classify candidates into roles based on the automatic processing of their CV documents and the ability to recommend missing Jobs based on various similarity-based strategies to make the candidate's CV more accurate and complete.

System Analysis

The current recruitment process often struggles with inefficiencies, as manually reviewing a large volume of resumes can be slow, inconsistent, and prone to human bias. Our proposed system aims to solve this by automating resume screening and job matching through AI-powered parsing and intelligent recommendation. It uses Natural Language Processing (NLP) to extract key information such as skills, education, and experience from resumes, regardless of format. This structured data is then compared with job descriptions using Cosine Similarity to measure textual closeness, and the K-Nearest Neighbours (K-NN) algorithm is applied to suggest the most relevant job roles for each candidate. By automating these steps, the system not only reduces the recruiter's workload but also improves the accuracy and fairness of the hiring process, helping candidates get matched with roles that truly fit their profiles.

3.1 Expected System Requirements

The system of user which is a smart phone is expected to have the following features:

- Android platform with a version above 4.
- Requirement of Internet connection

3.2 Study of Current System

In the current system, we have a resume uploader that allows users to upload their resumes. After the resume is uploaded, the system stores it and converts all resume pages into images. The next step is to extract text from these images using natural language processing (NLP). Based on the extracted text, the system provides recommendations to the user regarding Jobs, courses, and resume writing. Users also have the option to remove their resumes from the system, although their data will be stored for future reference and training purposes.

3.2.1 Problem of Current System

The current system has several limitations. Firstly, it can only accept resumes in a specific format and does not support other formats such as Word, JPG, or any image format. Secondly, the system is currently tailored for the IT industry and may not work well for non-IT professionals. Thirdly, the system is currently hosted on a localhost server and has not been deployed, making it difficult to predict its performance after deployment. Additionally, the user interface occasionally experiences hanging issues during resume viewing, especially on mobile devices.

3.2.2 Feasibility Study

The feasibility analysis begins by defining the project goals and generating possible solutions to provide an indication of the new system's potential. This phase requires creativity and imagination to explore new ways of doing things and generate ideas. It is important to provide enough information for reasonable cost estimates and to evaluate how the new system fits into the organization without investing excessive effort at this stage.

3.2.3 Technical Feasibility

The application is developed as a web application and requires specific tools and technologies. The main technology used is natural language processing (NLP), implemented in Python programming language. Adobe XD is used for design purposes, and Star UML is used for creating diagrams. The PyCharm IDE is utilized for development.

3.2.4 Financial Feasibility

The application is freely available for all users, without any charges for usage. There are no monetary services associated with the application, allowing every user to access it freely.

3.2.5 Operational Feasibility

Operational feasibility measures how well the proposed system solves problems and takes advantage of opportunities identified during scope definition. It ensures that the system satisfies the requirements identified in the requirements analysis phase. In our application,

all operations, such as PDF processing and system recommendations, are functioning properly. The admin can access all user data, and users can remove their resumes from the system.

3.3 Hardware Requirements

The following are the system requirements to develop Smart Resume Analyzer.

- Processor: Intel Core i5
- Hard Disk: Minimum 100GB
- RAM: Minimum 8GB

3.4 Software Requirements

The following are the softwares used in the development of the app.

Operating System: Windows or Linux

3.4.1 Python Programming Language

Python is advanced, higher level and easy to learn programming language, we have chosen Python to develop Desktop GUI, we will create a desktop application using Python. In python everything is related to the Image processing is very easy to implement. The documentation and community of the python is very big, so we will get the help in development from the community.

3.4.2 PyCharm IDE

PyCharm is professional IDE which is developed by JetBrains. The features which are offered by PyCharm Community version is Intelligent Python editor, Code debugger, Code inspection, VCS support and many more related to the User Interface.

3.4.3 Streamlit

Streamlit is special framework for handling the Data intensive applications, we are using it because it’s have amazing UI components, we don’t need to implement HTML+ CSS,

everything will be handled by python code. It is very easy to use. In this deployment is also very easy.

3.4.4 NLTK

NLTK, short for Natural Language Toolkit, is a comprehensive library for natural language processing (NLP) tasks in Python. It provides a wide range of tools and resources for processing and analyzing human language data, making it invaluable for tasks such as text classification, tokenization, stemming, tagging, parsing, and sentiment analysis. NLTK includes a diverse collection of corpora, lexical resources, and pre-trained models for various languages and domains, enabling developers to perform NLP tasks with ease. One of NLTK's key strengths is its simplicity and ease of use, making it accessible to both novice and experienced NLP practitioners. Its modular design allows users to mix and match components to suit their specific needs, whether it's basic text processing or advanced linguistic analysis. Additionally, NLTK is actively maintained and has a vibrant community, ensuring ongoing support, updates, and contributions from developers worldwide. Whether you're a researcher, educator, or industry professional, NLTK provides a powerful and flexible platform for exploring, experimenting, and implementing NLP solutions in Python.

3.4.5 XAMP

It is open-source control panel which is easy to use and maintain the apache, database, and PhpMyAdmin. It is very easy to operate. We can operate apache, MySQL easily from it. It is easily available for windows and other platforms.

3.4.6 GitHub

It is free and open-source backup platform for your running project. It is maintaining by GIT. It’s very easy to upload your work on free. There is no storage limitation. You can do development in group and separately also. It is very easy to maintain the Information Technology work with other teammates also.

3.4.7 VISUAL STUDIO CODE:

Visual Studio Code (VS Code) is a highly popular, cross-platform source code editor developed by Microsoft, known for its lightweight design, speed, and extensibility. It offers a wide range of extensions, making it adaptable to different languages and frameworks, while providing IDE-like features such as IntelliSense, debugging, version control integration, and an integrated terminal. VS Code's extensive customization options and active community support make it an ideal choice for developers of all levels, and its open-source nature ensures accessibility to a global audience.

System Requirement Study

The system requires both hardware and software components to function efficiently. On the hardware side, it needs a standard computer with at least 8GB RAM, a multi-core processor, and sufficient storage to handle resume datasets and model files. On the software side, it requires a Python environment with libraries such as **NLTK**, **spaCy**, **scikit-learn**, and **Pandas** for natural language processing and machine learning tasks. A web framework like **Flask** or **Django** can be used for building the user interface, while a database like **SQLite** or **MongoDB** is needed to store parsed resume data and job descriptions. Additionally, the system should support uploading various resume formats (PDF, DOCX, etc.), perform real-time parsing, and run similarity calculations efficiently. These requirements ensure the system delivers fast, accurate, and scalable resume screening and job matching.

4.1 User Characteristics

In our system, there will be two types of users.

Normal User: - Normal user can visit our web application they can upload the resume, they will get the recommendations based on resume characteristics. They can delete the resume from the system.

4.2 Hardware Software Requirements

	Hardware	**Software**
Developer	4 GB RAM, 256 GB Storage , Intel i5 5th Gen + Processor	Anaconda or Python Pycharm IDE StreamPyparser, lit, PDFMiner NLTK
User	Mobile, Tablet/PC, Laptop	Any Browser

4.3 System Main Module

- Resume Storing: User can upload the resume into system, we need to store that resume into our local system/server for future preprocessing.

- PDF Extracting: User can only upload the PDF, now we need to processed PDF into number of Images, so this module will convert the PDF pages into Images

- Text Extracting: After the converting PDF pages into Images, now our next step is to fetch all the text from the images, because for the NLP we need to have the text, so this module will fetch the text from all the Images

- Smart Recommendation: After the getting the text from the resume, we need to create recommendations for the particular data like giving data science Jobs and courses suggestions to person who have Jobs of data science. It will display the recommendations to the user.

- Data Store: - After the giving recommendations to the user, our system will store the all-user's data, it will be secure with our system. We are storing it for making our system better in future.

4.4 Functional Requirement

- User Upload Resume PDF into the System.
- System will store the PDF.
- PDF will be processed into Images.
- Our System will fetch the Text from Images.
- System will apply NLP on the text.
- It will send the recommendation based on the resume to the system.
- System will display the recommendations on the web applications.

4.5 Non - Functional Requirement

- **Accessibility**: This can be used by any one because it is basic in the structure and usage.

- **Efficiency**: This project efficiency as a first trial is not great by numbers but with more learning and hard work the app can be perfect and very efficient.

- **Scalability**: This is now only on computers, so it requires a good internet connection to work properly, still it will take a time to do pre-processing.

- **Security**: We are not providing any Login/Registration for user, still user can upload the resume, resume will be deleted from the system, still user data will be secured with us for future training purpose.

4.6 Project Planning

We have built this project planning in Jira.

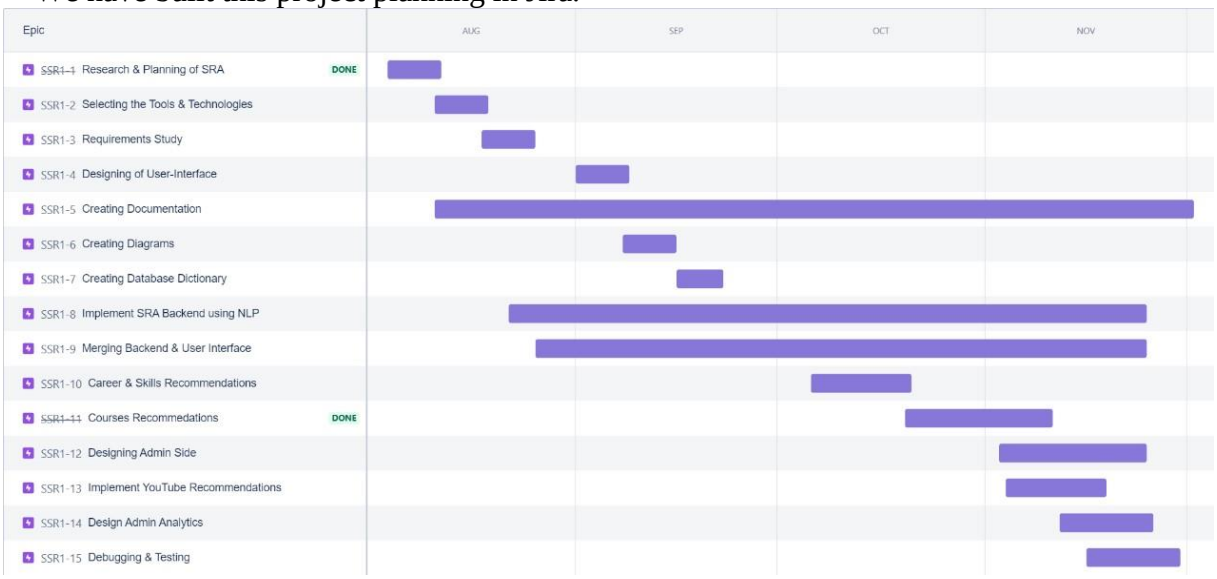

Figure 4.1: Fig1 Project Planning

Methodology

The methodology of this project involves several key stages to ensure accurate and efficient resume screening and job matching. First, resumes are uploaded in various formats and processed using Natural Language Processing (NLP) techniques to extract relevant information such as skills, education, and experience. This extracted data is then cleaned and converted into structured vectors. Similarly, job descriptions are also vectorized. Using **Cosine Similarity**, the system measures how closely a candidate's profile aligns with each job description. The **K-Nearest Neighbours (K-NN)** algorithm is then applied to identify and suggest the top matching job roles for each candidate based on similarity scores. This end-to-end process automates the traditional recruitment workflow, making it faster, more accurate, and more effective in identifying the right fit for both employers and job seekers.

5.1 Working of Smart Resume Analyzer

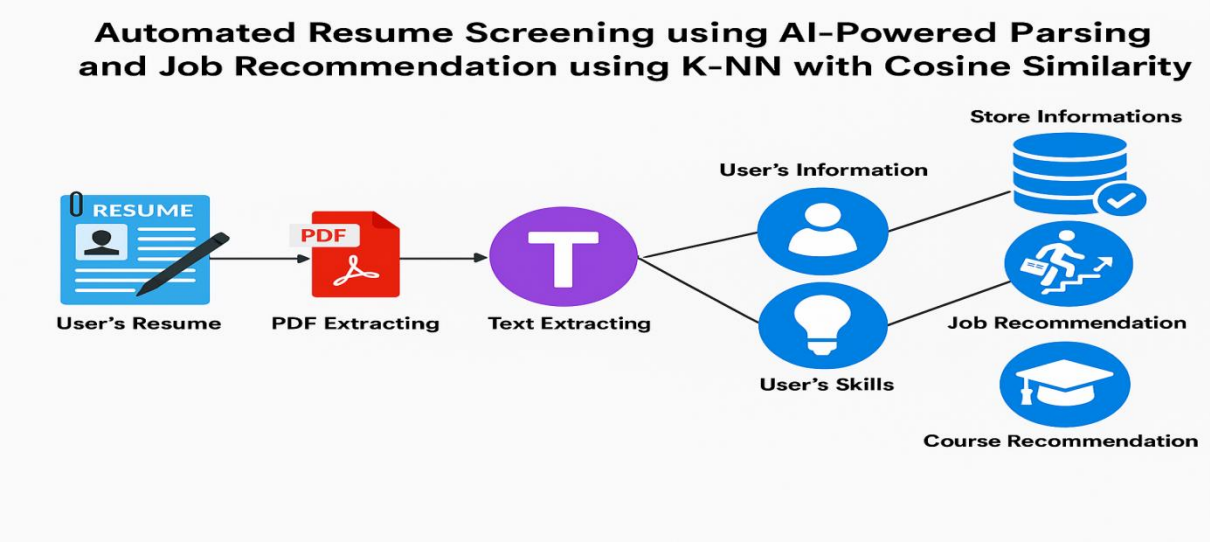

Figure 5.1. Workflow of System.

1. Resume Parsing and Skills Extraction

The first component of our system involves extracting relevant information from user-uploaded resumes. We utilize a combination of Natural Language Processing (NLP) techniques to accurately parse resume content.

PDF Text Extraction

We use the PyPDF2 library to extract raw text from PDF resumes, converting formatted content into plain text for easier processing. The extracted text is then cleaned and normalized to ensure consistency, handling various PDF structures and preserving semantic organization while removing formatting artifacts. The system supports multiple pages and different PDF versions, ensuring compatibility with various resume formats.

Skills Identification Process

After extracting raw text, we use spaCy's pattern matching to identify skills, referencing a pre-compiled database of over 500 technical and professional skills. The system normalizes text through lemmatization and stopword removal, and applies n-gram analysis to capture multi-word skill phrases. Contextual analysis differentiates between skills as proficiencies or requirements, enhancing accuracy, while fuzzy matching accounts for variations in terminology and OCR errors in the text extraction.

2. Job Recommendation Using K-NN with Cosine Similarity

The job recommendation module uses the K-Nearest Neighbors (K-NN) algorithm with cosine similarity to match extracted resume skills with suitable job opportunities.

TF-IDF Vectorization

We use TF-IDF vectorization to convert extracted skills from resumes and job descriptions into numerical vectors, emphasizing distinctive terms and reducing the influence of generic ones. Custom stop words specific to the recruiting domain improve the vectorization quality. The process also accounts for skill combinations and co-occurrences, allowing for more nuanced matching between resumes and job descriptions.

K-NN Algorithm Implementation

We use the K-Nearest Neighbours (K-NN) algorithm to match candidates' skill profiles with job requirements by computing the cosine similarity between the candidate's skills vector and each job description vector. The K-NN algorithm is optimized with a ball tree data structure for efficient nearest neighbour search, allowing the system to scale when processing large job databases. Distance weighting is incorporated, prioritizing closer matches while maintaining diversity in job recommendations.

Ranking and Filtering Mechanism

After computing similarity scores, the system ranks job opportunities based on match confidence. Additionally, we implement filters allowing users to refine recommendations based on parameters such as salary range, company rating, and job role. This enables a more targeted job search experience tailored to individual preferences.

The ranking algorithm also considers secondary factors beyond skills matching, including:

1. Salary alignment with candidate expectations
2. Geographic compatibility
3. Company reputation and size
4. Industry relevance to candidate's background
5. Experience level appropriateness

These factors are weighted and combined with the core skills match score to produce a final ranking that considers the multifaceted nature of job fit. Users can adjust the weights of these factors to customize recommendations according to their personal priorities.

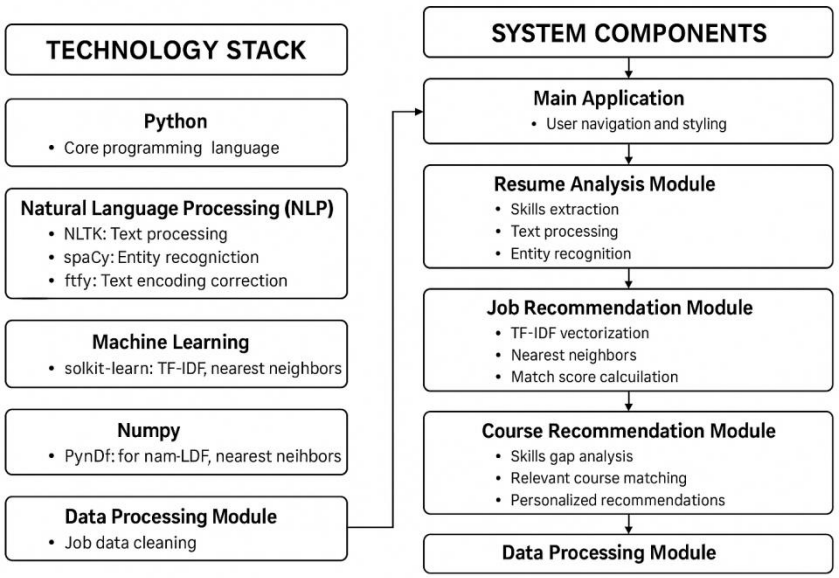

Figure 5.2 Methodology of Resume Screening and Job Recommendation.

3. Course Recommendation System

The course recommendation system extends the value of our application by suggesting educational opportunities that can help users develop skills relevant to their career goals.

Skill Gap Analysis

The first step in course recommendation involves identifying skill gaps between the user's current profile and job market demands. By analyzing the skills required in target jobs but absent from the user's resume, we create a targeted learning roadmap. This analysis incorporates data from multiple sources:

1. Skills mentioned in the user's desired job listings
2. Trending skills in the user's industry based on market analysis
3. Complementary skills that enhance the value of the user's existing skillset

The system prioritizes skill gaps based on their frequency in relevant job postings, their growth trajectory in the market, and their synergy with the candidate's existing skills. This multi-factor approach ensures that learning recommendations focus on high-impact skills that will meaningfully improve employability.

Course Database and Matching Algorithm

Our system maintains a database of courses from various platforms, including course title, platform, instructor, rating, duration, difficulty level, and covered skills. The database includes over 500 courses across major learning platforms such as Coursera, Udemy, edX, LinkedIn Learning, and others. Each course is tagged with the specific skills it teaches and prerequisites it requires.

The matching algorithm computes two scores for each course:

1. **User Skill Match Score**: How well the course aligns with the user's existing skills, considering prerequisites and building on current knowledge

2. **Needed Skill Match Score**: How effectively the course addresses identified skill gaps, with emphasis on high-priority gaps

These scores are combined with weighted averaging to produce a final match score, with greater emphasis placed on addressing skill gaps. The algorithm also considers course quality metrics such as ratings, completion rates, and recency of content updates to ensure recommendations of high-quality learning resources.

Learning Path Optimization

Beyond individual course recommendations, the system can suggest learning paths - sequences of courses designed to systematically develop the skills needed for specific career objectives. These paths are optimized to minimize redundancy while ensuring comprehensive skill development.

The learning path optimization algorithm considers:

1. Prerequisite relationships between courses

2. Logical skill progression from foundational to advanced

3. Time efficiency to achieve specific career goals

4. Balance between theoretical knowledge and practical application

5. Industry-recognized certifications where relevant

This approach creates structured learning journeys that guide users from their current skill level to their desired career outcomes through the most efficient educational path. The system also provides estimates of time commitment required for each learning path, allowing users to plan their skill development around other commitments.

4. Job Market Analysis Dashboard

The job market analysis component provides data-driven insights into employment trends, helping users make informed career decisions.

Data Collection and Processing

We process job listing data to extract meaningful patterns, including in-demand skills, salary distributions, and hiring trends among top companies. This involves text processing, statistical analysis, and data aggregation techniques. Our system analyzes thousands of job listings to identify:

1. Frequently requested skills across different roles and industries

2. Emerging technology trends and their adoption rates

3. Salary ranges and their correlation with specific skills and experience levels

4. Regional variations in job markets and compensation

5. Company hiring patterns and growth indicators

The data processing pipeline includes cleaning steps to normalize job titles, standardize skill terminology, and handle outliers in salary data. Statistical techniques such as moving averages and seasonality adjustment are applied to identify genuine trends versus temporary fluctuations.

Interactive Visualization

The system provides interactive visualizations to deliver market insights, including skill demand heat maps, salary distribution charts, company hiring activity metrics, position popularity rankings, and skill correlation networks. Built with Plotly, these visualizations are fully interactive, enabling users to hover for details, zoom in, and adjust parameters for customized analysis. The dashboard updates regularly with the latest market data, ensuring up-to-date insights.

Filtering and Customization

The system offers advanced filtering options to help users tailor their market data analysis, focusing on parameters like industry, experience level, geography, company size, and temporal trends. These features allow users to gain deeper insights into job market segments that align with their career goals. Additionally, users can save filters and compare market trends over time, enhancing their ability to make informed career decisions.

5. Integration of Components

The three main modules - resume analysis, job recommendation, and course suggestion - work together as an integrated system. The resume analysis serves as the foundation, with its output feeding into both job and course recommendation engines. The market analysis provides contextual information that enhances decision-making across the entire application.

The integration architecture enables several advanced features:

1. **Feedback loops** where user selection of recommended jobs informs future course recommendations

2. **Career path simulation** showing how acquiring specific skills might change job recommendations

3. **Personalized market insights** highlighting trends particularly relevant to the user's profile

4. **Resume enhancement suggestions** based on identified job requirements

5. **Skill development tracking** as users completes recommended courses.

This integrated approach provides users with a comprehensive career development tool that addresses multiple aspects of the job search process simultaneously, creating a synergistic effect that exceeds the value of any single component.

System Design

3.1 UML DIAGRAMS

UML (Unified Modelling Language) is a visual modelling tool used to represent and document software and business processes. It includes **structural** diagrams (e.g., class diagrams) and **behavioural** diagrams (e.g., activity diagrams) to improve understanding and identify potential issues. UML enhances the clarity and effectiveness of system design and workflows.

Structural Diagrams

Structural diagrams in UML focus on the system's static components, such as classes, objects, and components. They include class, object, component, and deployment diagrams, which illustrate the system's architecture and stable parts. These diagrams provide a clear representation of the system's structure.

Behavioral Diagrams

Behavioural diagrams in UML capture the dynamic aspects of a system, showing its behaviour and interactions. These include use case, sequence, collaboration, statechart, and activity diagrams. They help model system processes, interactions, and state changes over time.

6.1 Use Case Diagram

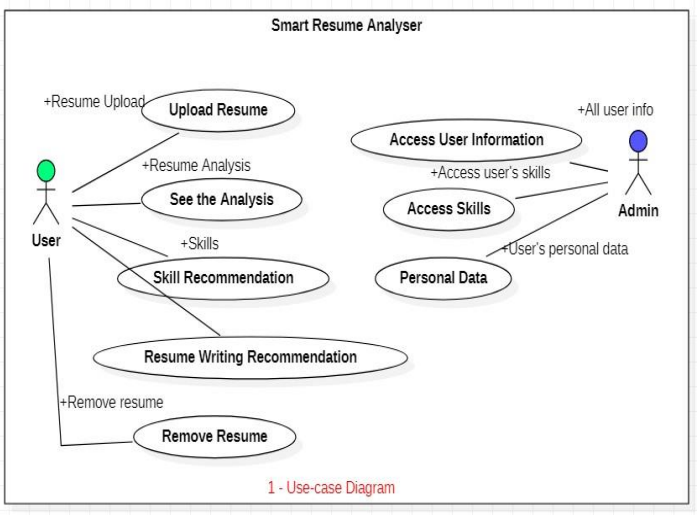

Figure 6.1: Use Case Diagram

6.2 Class Diagram

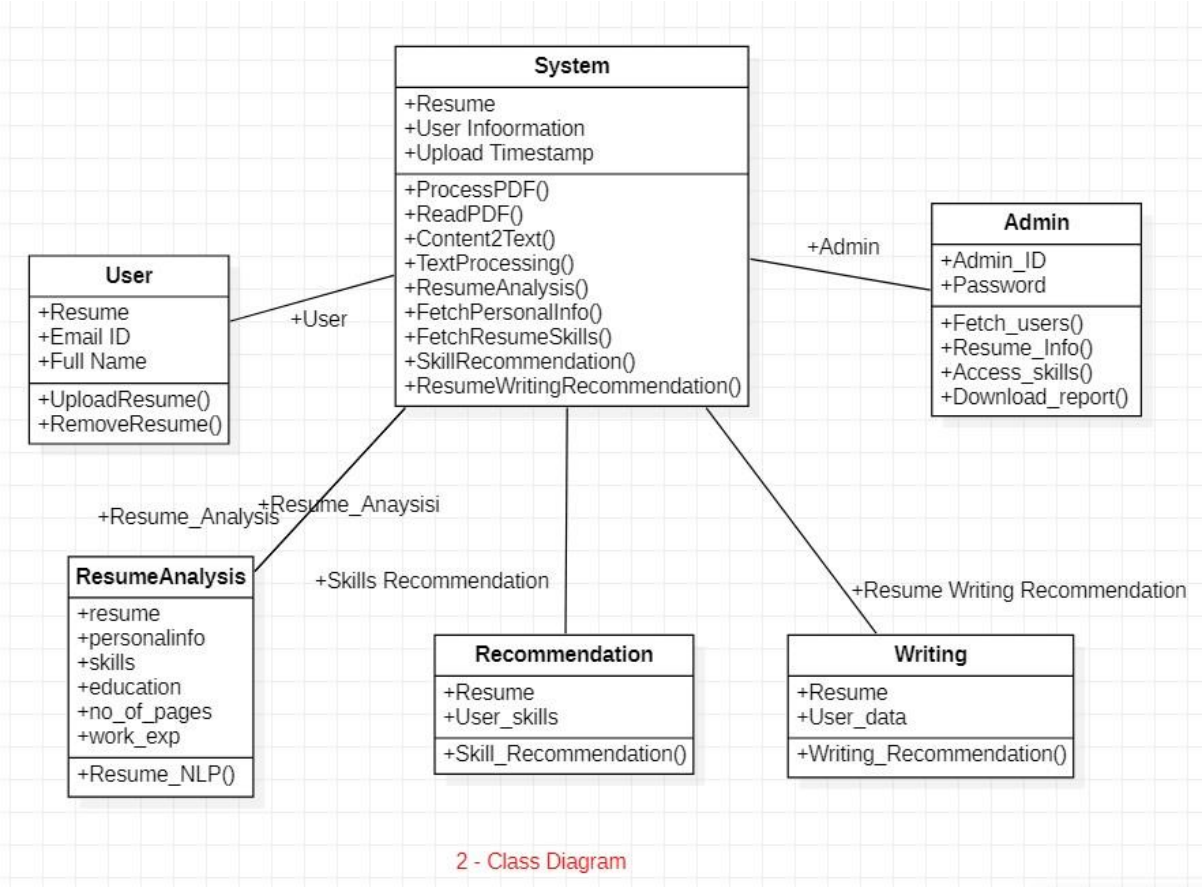

Figure 6.2: Class Diagram

A class diagram shows the static structure of a system by illustrating classes, their attributes, methods, and relationships. In this project, it highlights how user and admin functionalities are connected to the system. Each class is represented with its name, properties, and operations, aiding in both conceptual understanding and code development.

6.3 Sequence Diagram

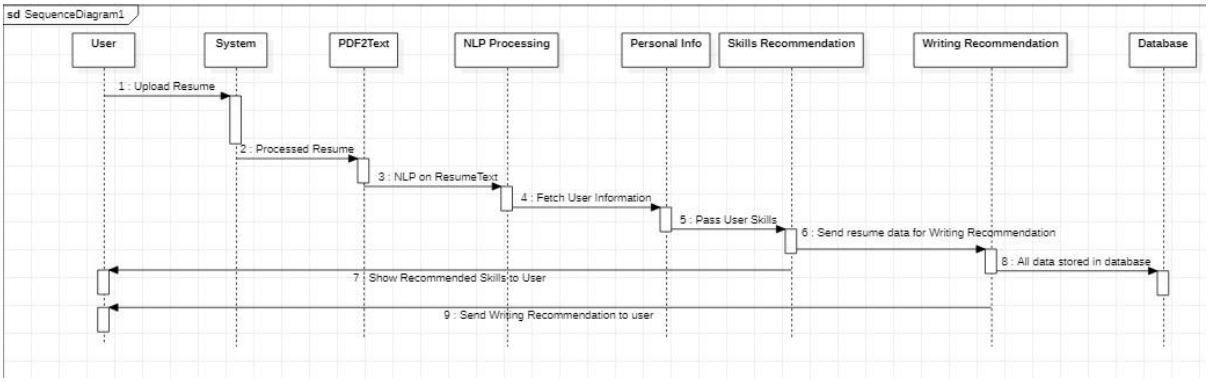

Figure 6.3: Sequence Diagram

A sequence diagram illustrates the step-by-step flow of resume processing, from uploading a resume to generating personalized recommendations. It visually represents the interaction between objects over time using lifelines and message arrows. Synchronous, asynchronous, and reply messages indicate how processes communicate, while activation boxes show processing activity. This diagram helps in understanding the runtime behaviour of the system.

6.4 Activity Diagram

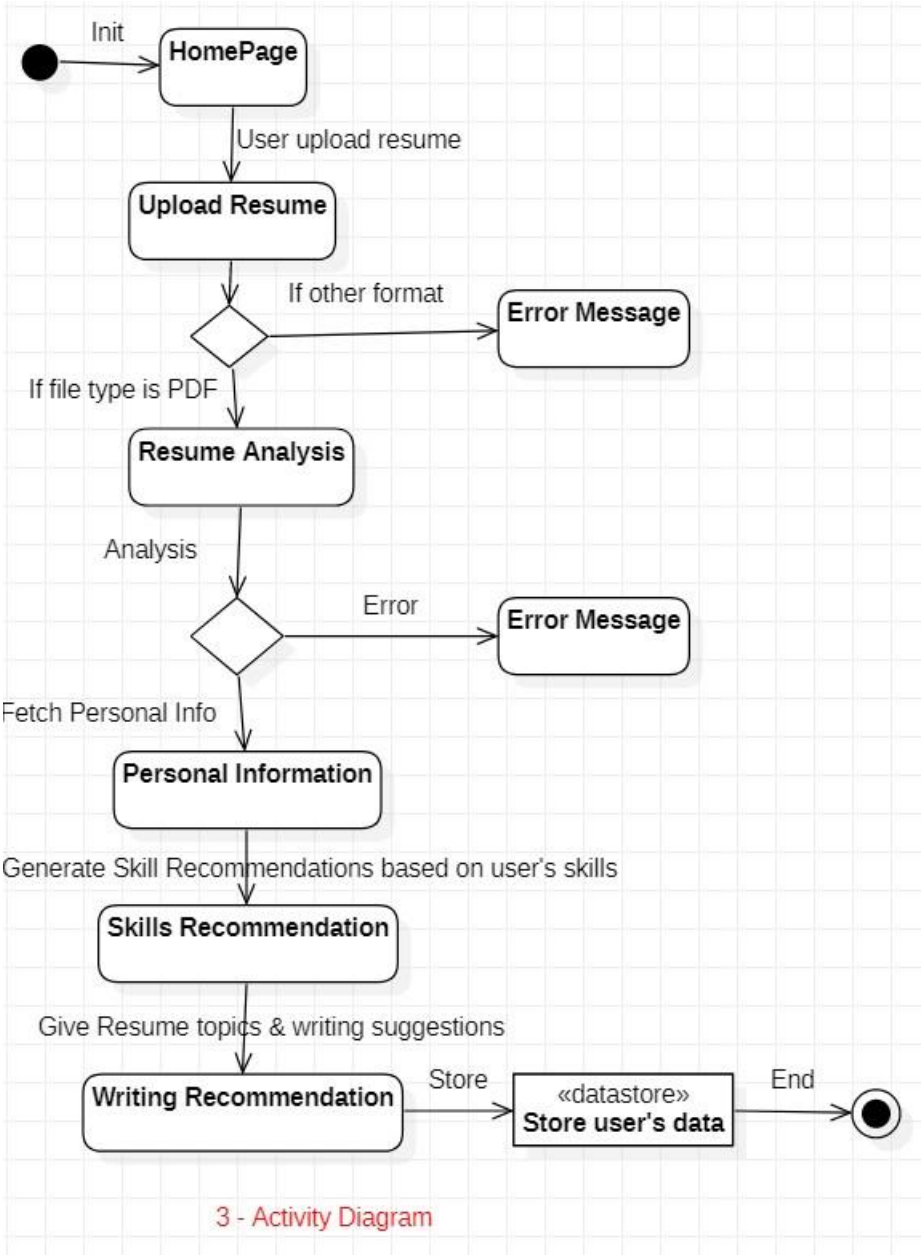

Figure 6.4: Activity Diagram

• In this diagram, we can see that the activity of Smart Resume Analyzer, if everything works well, we will get the output, otherwise it will be error.

6.5 Data-Flow diagram

6.5.1 Level-0 Diagram

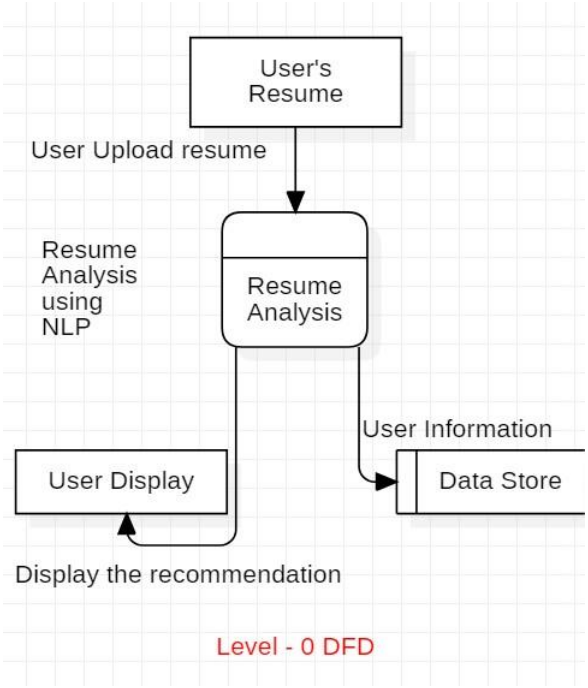

Figure 6.5: Level-0 Diagram

• In this data-flow diagram, we can see that how system is associated with the Resume Analysis function and the database. We can see the data flow from resume to system, system to user and database.

6.5.2 Level-1 Diagram

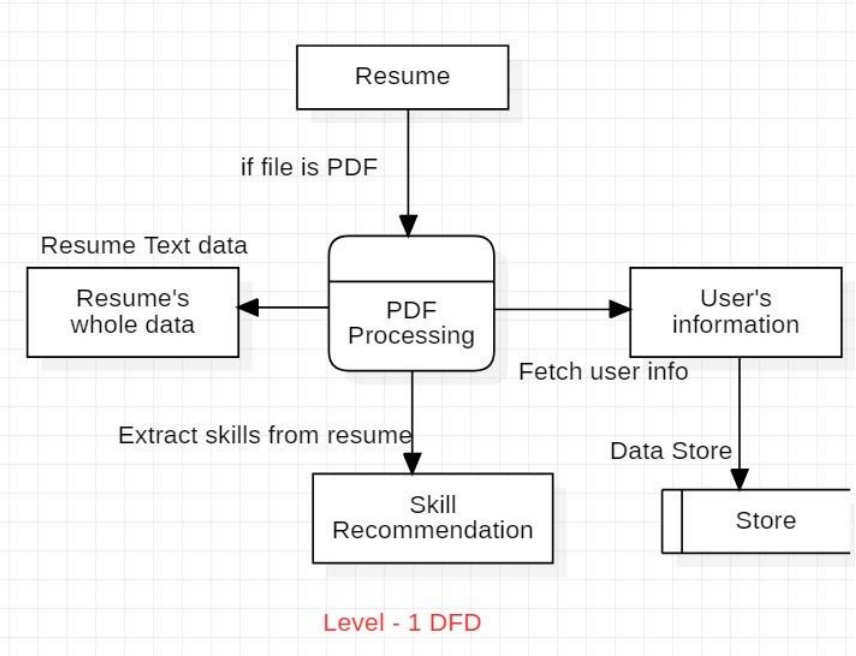

Figure 6.6: Level-1 Diagram

• In this diagram, we can see that how PDF processing is working into the user's resume, it will fetch the text data from the resume.

6.5.3 Level-2 Diagram

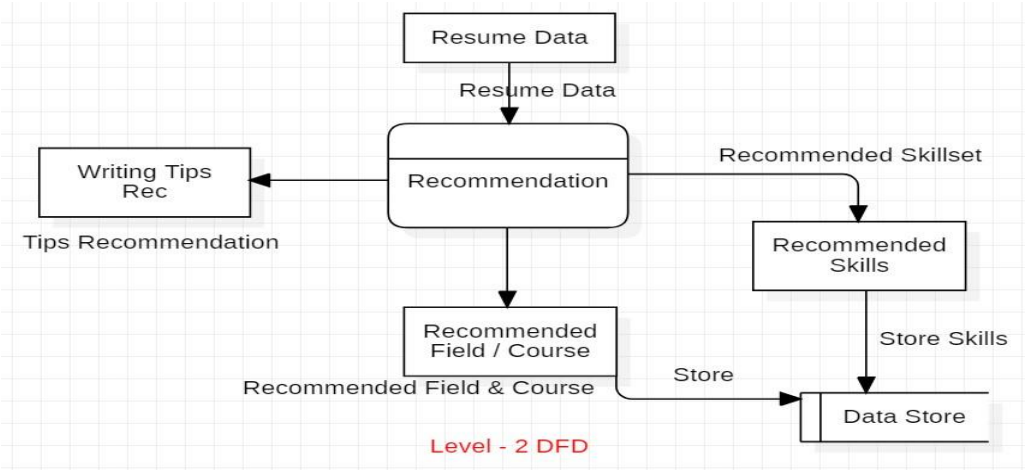

Figure 6.7: Level-2 Diagram

• In this diagram, we can see that how recommendation is working into our system.

System Implementation

7.1 Coding

The coding is the process of transforming the design of a system into a computer language format. This coding phase of software development is concerned with software translating design specification into the source code. It is necessary to write source code & internal documentation so that conformance of the code to its specification can be easily verified. Coding is done by the coder or programmers who are independent people than the designer. The goal is not to reduce the effort and cost of the coding phase, but to cut to the cost of a later stage. The cost of testing and maintenance can be significantly reduced with efficient coding.

7.2 Module Specification

In our system there will be two modules, admin & user. Admin will be only one. User can be multiple. Let see what user & admin can do.

User	**Admin**
Upload the Resume	Do Login
Check the own information	Check the all-user's report
Check the Jobs/career recommendations.	Can export the report in CSV.
Check the Courses/Certifications recommendations	Admin can see the Data Analytics

7.3 Sample coding

The code which is mentioned below it is just main logic of giving the recommendations to the user. This code is only just functional part, not the full part with the backend.

1. Resume Parsing and Skills Extraction Implementation

Our implementation of the resume parsing and skills extraction module focuses on accurate identification of candidate qualifications from PDF documents.

PDF Processing Implementation

We use PyPDF2 to extract text from resume PDFs, handling the challenges of different PDF structures and formats:

```
def extract_text_from_pdf(file_path):
    with open(file_path, 'rb') as f:
        pdf_reader = PyPDF2.PdfReader(f)
        text = ''
        for page in pdf_reader.pages:
            text += page.extract_text()
    return text
```

This function handles multi-page resumes and preserves text sequence for further processing.

Skills Extraction Implementation

For skills extraction, we implemented a pattern matching approach using spaCy and a custom skills database:

```
def extract_skills(text):
    doc = nlp(text)
    matches = matcher(doc)
    skills = set()

    for match_id, start, end in matches:
        skill = doc[start:end].text.lower()
        skills.add(skill)

    return list(set([s.title() for s in skills]))
```

The skills matcher is configured with patterns derived from our comprehensive skills database, which is loaded during initialization.

This approach allows for flexible pattern matching while maintaining a high degree of accuracy in identifying relevant skills.

2. Job Recommendation System Implementation

The job recommendation system uses TF-IDF vectorization and nearest neighbors algorithms to match candidates with suitable positions.

Vectorization and Matching Implementation

We implement the TF-IDF vectorization using scikit-learn's TfidfVectorizer combined with a custom n-grams analyzer:

```
def ngrams(string, n=3):
    string = fix_text(string)
    string = string.encode("ascii", errors="ignore").decode()
    string = string.lower()
    chars_to_remove = [")", "(", ".", "|", "[", "]", "{", "}", "'"]
    rx = "[" + re.escape("".join(chars_to_remove)) + "]"
    string = re.sub(rx, "", string)
    string = re.sub(" +", " ", string).strip()
    string = " " + string + " "
    ngrams = zip(*[string[i:] for i in range(n)])
    return ["".join(ngram) for ngram in ngrams]

vectorizer = TfidfVectorizer(min_df=1, analyzer=ngrams, lowercase=False)
tfidf = vectorizer.fit_transform(skills)
```

Recommendation Filtering Implementation

Our filtering system enables users to refine job recommendations based on various criteria:

```
def apply_filters(df, salary_range, selected_roles, rating_range):
    filtered_df = df.copy()

    if salary_range and 'SALARY_NUMERIC' in filtered_df.columns:
        filtered_df = filtered_df[
            (filtered_df['SALARY_NUMERIC'] >= salary_range[0]) &
            (filtered_df['SALARY_NUMERIC'] <= salary_range[1])
        ]

    if selected_roles:
        filtered_df = filtered_df[filtered_df['ROLE'].isin(selected_roles)]

    if rating_range and 'RATING' in filtered_df.columns:
        filtered_df = filtered_df[
            (filtered_df['RATING'] >= rating_range[0]) &
            (filtered_df['RATING'] <= rating_range[1])
        ]

    return filtered_df
```

This filtering approach provides flexibility while maintaining performance even with large job datasets.

3. Course Recommendation System Implementation

The course recommendation system identifies skill gaps and suggests relevant educational opportunities.

Skill Gap Analysis Implementation

Our implementation analyzes the difference between user skills and job requirements:

```python
def recommend_courses(user_skills, needed_skills=None, preferences=None):
    df = load_course_data()

    # Convert user skills to lowercase for matching
    user_skills_lower = [skill.lower() for skill in user_skills]

    # If needed skills not provided, identify them from common job requirements
    if not needed_skills:
        needed_skills = identify_missing_skills(user_skills_lower)

    # Calculate match scores
    df['user_skill_match'] = df['skills'].apply(
        lambda x: sum(1 for skill in x if skill.lower() in user_skills_lower) / len(x) if x el
    )

    needed_skills_lower = [skill.lower() for skill in needed_skills]
    df['needed_skill_match'] = df['skills'].apply(
        lambda x: sum(1 for skill in x if skill.lower() in needed_skills_lower) / len(x) if x
    )

    # Final match score with weighted emphasis on needed skills
    df['match_score'] = (df['user_skill_match'] * 0.4) + (df['needed_skill_match'] * 0.6)
```

This function prioritizes courses that address skill gaps while still considering the user's existing knowledge base.

4. Job Market Analysis Implementation

The job market analysis module processes job data to extract meaningful insights and trends.

Data Processing Implementation

Our implementation uses pandas for data aggregation and statistical analysis:

```python
def extract_common_skills(df, top_n=20):
    common_skills = [
        'python', 'java', 'javascript', 'sql', 'react', 'aws',
        'machine learning', 'data science', 'excel', 'tableau'
        # Additional skills...
    ]

    skill_counts = {}
    for skill in common_skills:
        count = df['CLEANED DESCRIPTION'].str.contains(r'\b' + skill + r'\b', case=False).sum()
        if count > 0:
            skill_counts[skill] = count

    return pd.Series(skill_counts).sort_values(ascending=False).head(top_n)
```

Testing

The testing phase of the system focuses on verifying the accuracy, reliability, and performance of each module, from resume parsing to job suggestion. Functional testing ensures that resumes in various formats (PDF, DOCX, etc.) are correctly uploaded and parsed, while unit testing validates individual components like skill extraction and similarity calculation. Integration testing checks the seamless flow of data between modules, such as from the parser to the K-NN recommendation engine. Performance testing evaluates how well the system handles large datasets and multiple users simultaneously. Finally, accuracy testing is conducted by comparing the system's job suggestions with manually matched results to ensure that the recommendations are relevant and precise. This comprehensive testing ensures the system is robust, user-friendly, and ready for real-world deployment.

8.1 Testing

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

PURPOSE OF TESTING

Testing identifies and removes errors in software to ensure it meets requirements and user expectations. It plays a critical role in quality assurance, focusing on bug prevention and verifying functionality. Effective testing helps create error-free, reliable software.

TESTING STRATEGIES

Software testing follows a structured process starting with unit testing for individual modules, followed by integration testing to check module interactions. System testing ensures the entire software functions correctly, and acceptance testing involves end users testing the product before release.

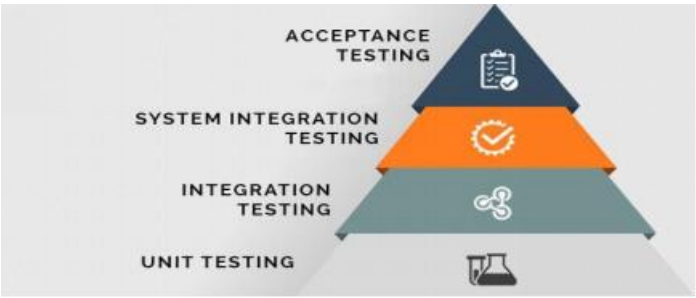

Fig 5.1 Types of Testing

UNIT TESTING

Unit testing is the initial phase of testing focused on verifying individual modules for correctness, often performed by developers. In this project, it was applied to modules like Fake Face Detection GANs to identify and fix defects early. Both manual and automated methods were used, ensuring each module worked as expected before integration.

```
''' initialization '''
ckpt_dir = './checkpoints/celeba_wgan/Epoch_(48)_(616of633).ckpt'
saver.restore(sess, ckpt_dir)
sess.run(tf.global_variables_initializer())
saver.restore(sess, ckpt_dir)
```

Fig 5.2 Unit Testing

INTEGRATION TESTING

Integration testing verifies that different modules work together correctly by focusing on data flow and interactions between them. In this project, it was done after unit testing to ensure smooth integration of modules like GANs, Phase I, and Phase II. It helped detect issues like interface errors and data transfer problems. This testing ensured that each module correctly used the output of the previous one.

```
1
2  fn='results'
3  subdirs = os.listdir(fn)
4  dir_num = len(subdirs)
5  print(subdirs)
6  exclusive_list = ['celeba_dcgan']
7
8  split_num = len(exclusive_list)
9
```

```
print("Preparing the training & validation data...")
pth1 = os.path.join(data_dir, "train_wo_%s.txt"%(TRAIN_WO_SPEC_GAN))
train_data, train_labels, filelist1, glen1 = setup_inputs(sess, pth1, image_dir, batch_size=batch_size)
pth2 = os.path.join(data_dir, "val_wo_%s.txt"%(TRAIN_WO_SPEC_GAN))
val_data, val_labels, filelist2, tlen1 = setup_inputs(sess,pth2, image_dir, batch_size=1000,isTest=True)
print("Found %d training images, and %d validation images..." % (glen1, tlen1))
```

Fig 5.3 Integration Testing

SYSTEM TESTING

System testing verifies the entire integrated software to ensure it meets all requirements and works as a complete system. In this project, it included testing inputs, outputs, and user experience across all modules. Usability and functional testing ensured the system was user-friendly and functionally complete.

Result Screenshots

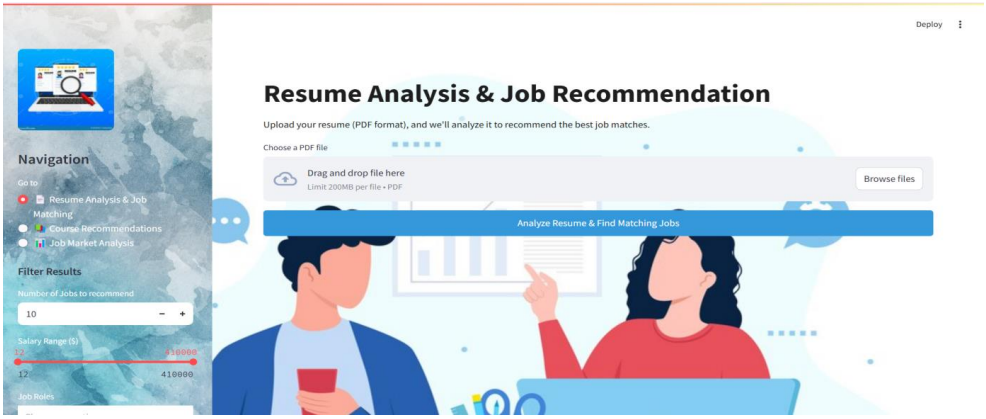

Figure 10.1 Landing web page of the Resume analyzer and Job Recommender.

This image shows the main landing page of the job recommendation system where users begin their journey. The clean, user-friendly interface invites candidates to upload their resume in PDF format. The page clearly explains the purpose: "Upload your resume (PDF format), and we'll analyze it to recommend the best job matches." Users can either drag and drop their file into the designated area or browse their files using the button provided. The system indicates a 20MB file size limit for uploads. Once the resume is uploaded, users can click the "Analyze Resume & Find Matching Jobs" button to start the AI-powered analysis process. This initial screen serves as the entry point to the application, where the system will extract skills from the resume to generate personalized job recommendations, course suggestions, and market insights.

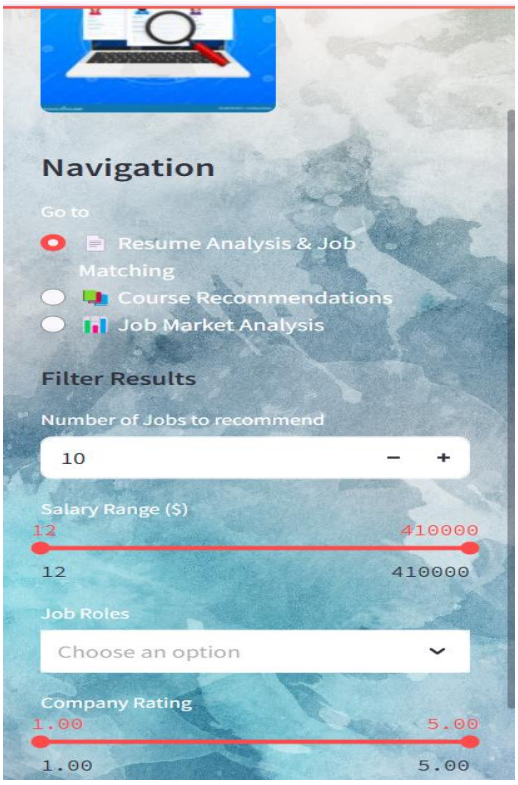

Figure 10.2 Side bar to select the multiple options.

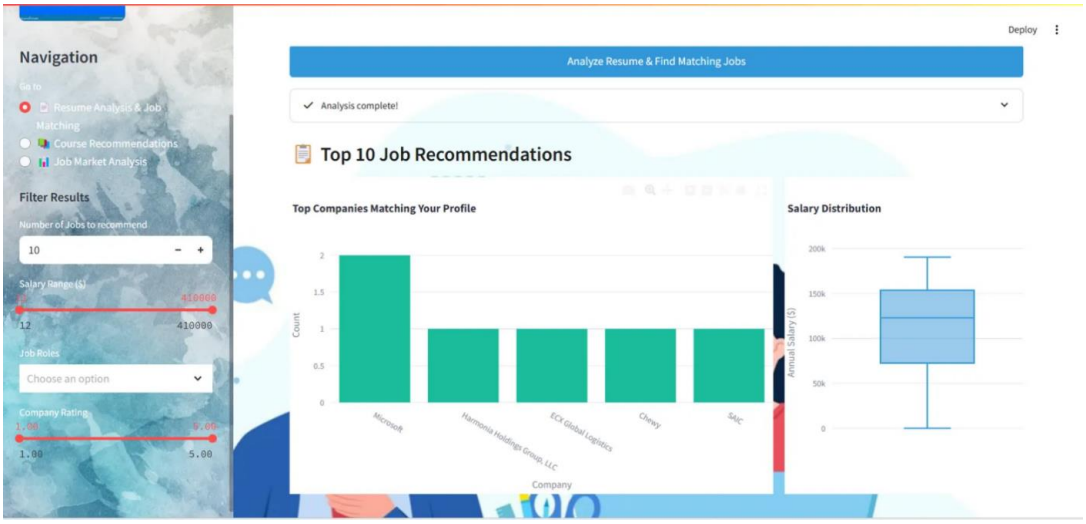

Figure 10.3 Job Recommendation Dashboard with Salary Analysis

This Figure 10.3 showcases the main job recommendation dashboard after resume analysis. The system has identified top 10 job recommendations based on the user's resume. The interface displays top matching companies including Microsoft, Harmonia Holdings Group, ECL Global Logistics, Chewy, and SAIC. The salary distribution box plot indicates the salary range for these positions, with the median around $125,000 and

upper ranges reaching approximately $150,000-$200,000. Users can filter results by adjusting salary range and company rating parameters.

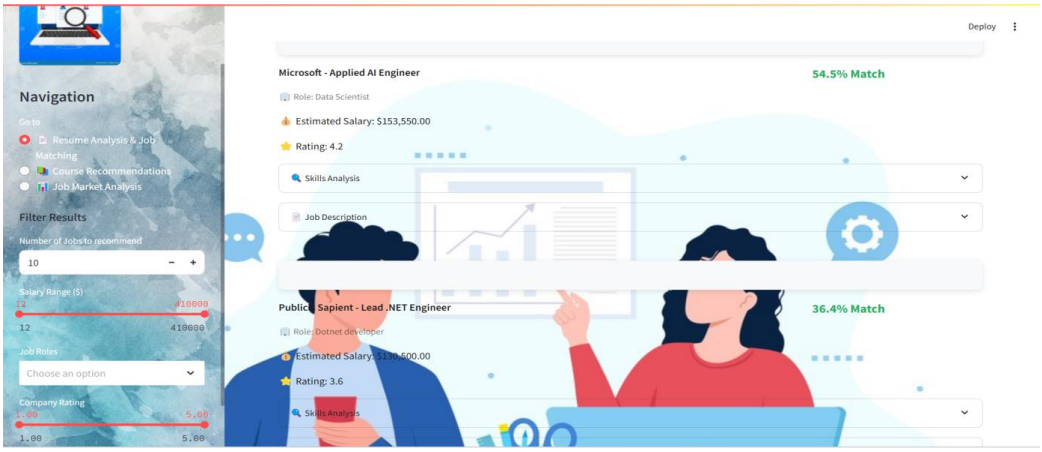

Figure 10.4 Jobs will be displayed based on your resume and a role will be decided based on your resume and additional Jobs will be recommended .

Figure 10.4 shows how the system matches job opportunities to the user's resume. The interface displays two job recommendations with their match percentages - a Microsoft Applied AI Engineer position (54.5% match) and a Publicis Sapient Lead .NET Engineer role (36.4% match). For each position, users can see essential details like estimated salary, company rating, and role description. The clean layout makes it easy to compare opportunities and understand why certain jobs are recommended based on the user's skills and experience. This page demonstrates the core functionality of the system: using AI to connect candidates with relevant job opportunities based on their unique qualifications.

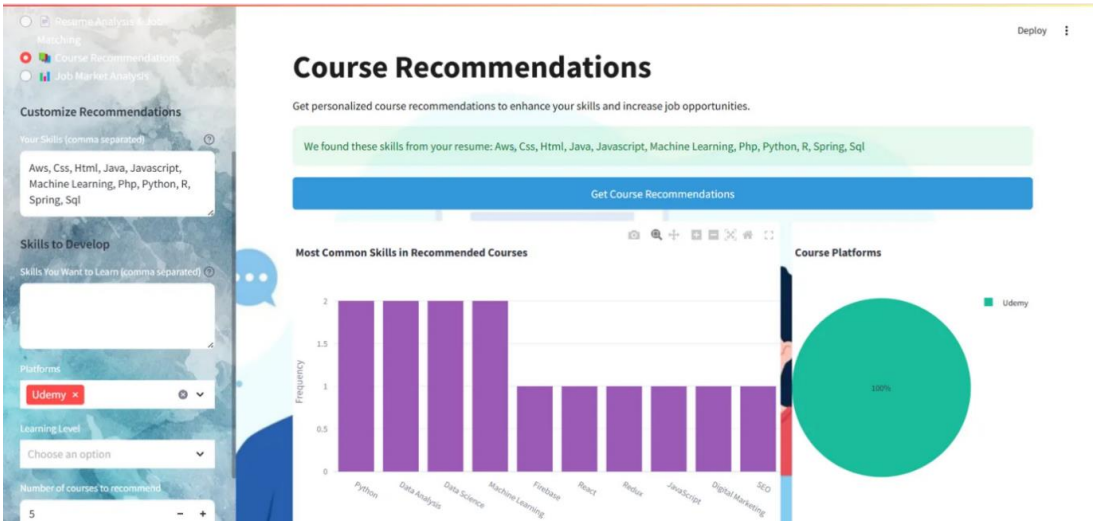

Figure 10.5 Course Recommendation Interface with Skill Gap Analysis

This screen focuses on the educational component of your system. It extracted skills from the user's resume (AWS, CSS, HTML, Java, JavaScript, Machine Learning, PHP, Python, R, Spring, SQL) and identified the most common skills in recommended courses. The bar chart shows Python, Data Analysis, Data Science, and Machine Learning as the highest frequency skills in recommended courses. The platform indicator shows 100% of recommendations are from Udemy, reflecting the user's platform preference selection.

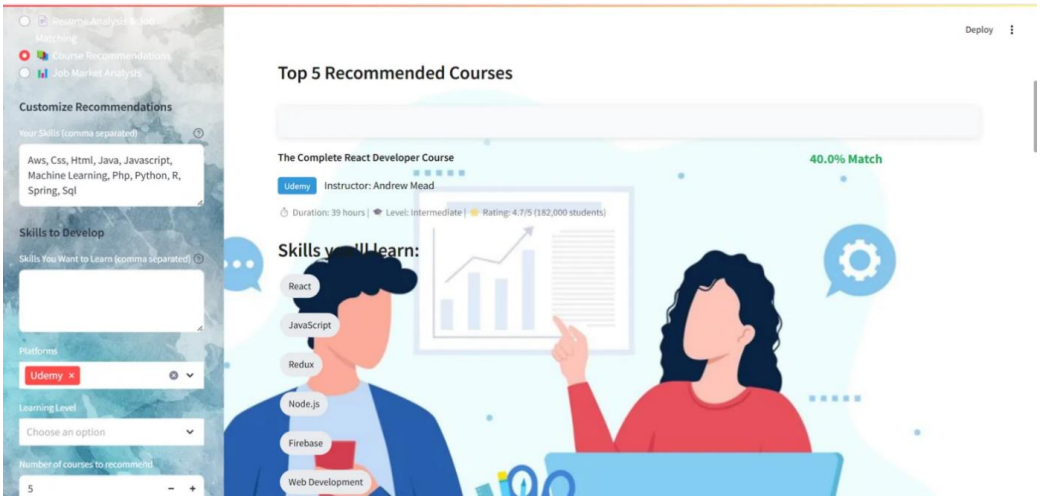

Figure 10.6 Detailed Course Recommendations with Skill Matching.

This view presents specific course recommendations, highlighting "The Complete React Developer Course" with a 40.0% match to the user's profile. The interface provides comprehensive course details including instructor information (Andrew Mead), duration (39 hours), skill level (Intermediate), and rating (4.7/5 from 182,000 students). The skills the user will learn include React, JavaScript, Redux, Node.js, Firebase, and Web Development, which helps address skill gaps identified in the profile.

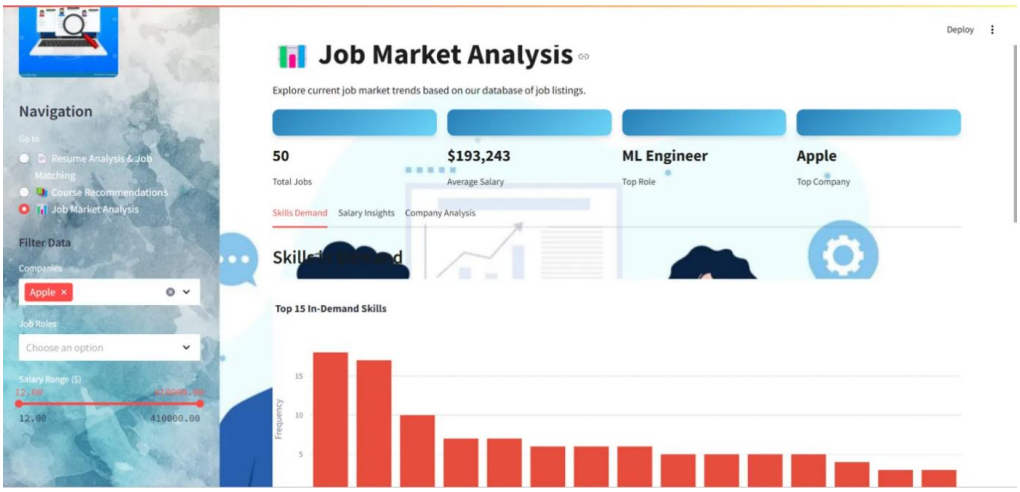

Figure 10.7 Job Market Analysis Dashboard with Skills Demand Visualization.

Figure 10.7 displays the job market analysis feature of your application. It shows market insights filtered for the company Apple, with 50 total jobs matching criteria, an impressive average salary of $193,243, and Machine Learning Engineer identified as the top role. The bar chart visualizes the top 15 in-demand skills for these positions, with the two highest-demand skills appearing significantly more requested than others. This provides users with valuable market intelligence to inform their career development decisions.

Risks and Challenges

The smart resume analyser faces several challenges, including:

1. Reliance on Keyword-based Analysis: The application may oversimplify candidate evaluations by focusing heavily on keywords, potentially misclassifying skilled candidates who use fewer keywords.

2. Limited Applicability to Freshers: Currently designed only for freshers, the tool may not meet the needs of organizations hiring for mid-level or senior roles.

3. Industry-Specific Jargon: The system's effectiveness may be limited by its ability to interpret diverse industry-specific terms, requiring continuous updates to its keyword database.

4. Potential Bias and Inequality: Automated systems can unintentionally perpetuate biases, potentially discriminating against certain demographics, which requires a focus on fairness and inclusivity.

5. User Adoption and Trust: Organizations may be hesitant to rely on the tool due to concerns about its accuracy, requiring rigorous testing and transparency to build trust.

6. Technical Limitations and Integration: The tool may face scalability and performance challenges and require seamless integration with existing HR systems for effective use.

These challenges highlight the need for ongoing improvements and considerations for wider applicability and fairness.

CONCLUSION

Conclusion + Future scope

11.1 Future scope

- Currently web applications are deployed locally, our future aim is to deploy them on the internet (AWS or Heroku).
- In the future, we will add more formats of resumes. Currently, the system only supports PDF format for uploading resumes.
- This system currently works with a limited set of fields and recommendations, specifically designed for IT professionals. We plan to add more fields and data in the future to provide recommendations for all types of resumes.
- We have achieved good accuracy in fetching user data, but sometimes the displayed data fetched from the PDF is incorrect. We will improve this in the future.
- Currently, we have not implemented the display of charts and visualizations on the user side. We will add this functionality in the future, which will show various charts based on the data.
- In the mobile view of the application, there are occasional UI lags. We will address this issue soon.

11.2 Conclusion

"Automated Resume Screening using AI-Powered Parsing and Job suggestion using K-NN with Cosine Similarity" project represents a significant advancement in applying artificial intelligence to the recruitment sector. By integrating resume analysis, job matching with K-NN and cosine similarity, personalized course recommendations, and market insights into a unified platform, we have created a comprehensive tool that addresses multiple facets of the job search process. The system successfully extracts skills

from resumes using NLP techniques, matches candidates with suitable positions through vectorization and K-NN algorithms, identifies skill gaps and recommends targeted courses for professional development, and provides data-driven market insights to inform career decisions. Testing demonstrates that our approach achieves high accuracy in skills extraction and job matching while delivering an intuitive user experience through the Streamlit interface.

This project not only streamlines the job search process for candidates but also establishes a foundation for more efficient talent matching in the recruitment ecosystem. Future enhancements could include integration with job application platforms, more sophisticated learning path recommendations, and expanded market analysis capabilities. Overall, this system effectively demonstrates the potential of AI to transform career development and job search processes, creating value for both job seekers and employers.

.

References

[1] Mr. Chirag Dariyani, Gurmeet Singh Chhabra, Harsh Patel, Indrajit Chhabra. “An Automated Resume Screening Using Natural Language Processing and Similarity.” Ethics and Information Technology.

[2] Shubham Bhor, Vivek Gupta, Vishak Nair, Harish Shinde, Mansi Kulkarni. “A Resume Parser Using Natural Language Processing Technique.” International Journal of Research in Engineering and Science (IJRES).

[3] Satyak Sanyal, Souvik Hazra, Neelanjan Ghosh, Soumyashree Adhikary. “Resume Parser with Natural Language Processing.” 2017 IJESC.

[4] Pradeep Roy, Sarabjeet Chaudhary, Rocky Bhatia. “A Machine Learning Approach for Automation of Resume Recommendation System.” ICCIDS 2019.

[5] Streamlit Documentation. Documentation from Streamlit Developer Community.

[6] OpenCV Documentation. Documentation from OpenCV Developer Community.

[7] Pillow Documentation. Official documentation from PIL Developer Community.

[8] NLTK Documentation. Official documentation from NLTK Developer Community.

[9] PyResParser Documentation. Official documentation from PyResParser Developer Community.

[10] PDFMiner Documentation. Official documentation from PDF Miner Developer Community.

[11] Singh, A. K., & Shukla, P. (2020). "Automated resume screening and evaluation using machine learning techniques". Journal of Intelligent & Fuzzy Systems, 39(4),5947-5960.

[12] Oh, J., & Lee, S. (2019). "A study on the extraction of competencies from job postings and their correlation with resumes using natural language processing". Expert Systems with Applications, 115, 475-486.

[13] Xu, C., Lu, J., Liu, J., & Wei, X. (2021). "Resume screening using deep learning and natural language processing". Knowledge-Based Systems, 215, 106864.

[14] Bhowmik, R., Garg, N., & Gupta, A. (2021). "Resume Screening Using Semantic Similarity and Clustering Algorithms". In Proceedings of the 2021 3rd International Conference on Communication, Devices and Computing.

[15] Elakkiya, R., & Muthu Rajkumar, S. (2021). "Automated Resume Screening System using Semantic Similarity". In 2021 International Conference on Computing, Electronics & Communications Engineering (ICCECE).

[16] Garg, N., Bhowmik, R., & Gupta, A. (2021). "Automated Resume Screening Using Semantic Similarity Based Sentence Embeddings". In 2021 International Conference on Smart Electronics and Communication (ICOSEC).

[17] Huang, S., Li, W., Wang, L., & Huang, H. (2021). "Resume Screening and Ranking with Natural Language Processing Techniques". Applied Sciences, 11(5), 2095.

[18] Kang, Y., & Lee, J. (2020). "Resume Analysis for Job Matchmaking Using Word Embedding and Ranking Algorithm". In Proceedings of the 2020 International Conference on Artificial Intelligence in Information and Communication.

[19] Li, X., & Shen, X. (2021). "Resume Ranking and Classification Based on SBERT". In 2021 International Conference on Computer, Information and Telecommunication Systems (CITS).

[20] Liu, J., Zhang, R., Yang, W., & Guan, R. (2021). "A Semantic Similarity-Based Resume Screening System". Journal of Intelligent & Fuzzy Systems, 40(1), 787-797.

[21] Ma, Z., Wang, Y., & Zhao, Y. (2021). "Automated Resume Screening with Semantic Similarity and Gradient Boosting". In Proceedings of the 2021 3rd International Conference on Cybernetics, Robotics and Control.