Six Weeks Industrial Training Project Report

On

"AI-Based Resume Screening and Analysis System"

Submitted in the partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

In

Computer Science and Artificial Intelligence

Batch (2022-2026)

Submitted to :

Mrs. Bindu Bansal

Assistant Professor

Submitted by :

Komal Saini
12200302

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DAV UNIVERSITY

JALANDHAR-PUNJAB 144012

ACKNOWLEDGEMENT

I express my gratitude to all those who helped us in various stages of the development of this project. First, I would like to express my sincere gratitude indebtedness to Dr. Rahul Hans (Coordinator) of DAV University for allowing me to undergo the summer training of 45 days at Coderroots, Mohali. I am also thankful to all faculty members of Department of Computer Science and Engineering, for their true help, inspiration and for helping me for the preparation of the final report and presentation.

Last but not least, I pay my sincere thanks and gratitude to all the Staff Members of Coder roots for their support and for making our training valuable and fruitful.

DECLARATION

I, Komal Saini, hereby declare that the work which is being presented in this project/training titled " AI-Based Resume Screening and Analysis System" by me, in partial fulfilment of the requirements for the award of Bachelor of Technology (B.Tech) Degree in "Computer Science and Artificial Intelligence" is an authentic record of my own work carried out under the guidance of Mr. Gurminder Singh ("Machine learning Engineer").

To the best of my knowledge, the matter embodied in this report has not been submitted to any other University/ Institute for the award of any degree or diploma.

Komal Saini

(12200302)

CERTIFICATE

ABSTRACT

Manual resume screening has become increasingly challenging as the number of applicants for software and AI/ML roles continues to grow in colleges and IT companies. Recruiters and faculty coordinators often spend hours reading similar resumes, applying subjective judgement, and maintaining marks in spreadsheets, which leads to slow shortlisting, inconsistency between evaluators, and very little visibility into the overall skill profile of a batch. To overcome these limitations, the "AI-Based Resume Screening and Analysis System" has been developed as a training project using Python, Natural Language Processing (NLP) and Streamlit, with the goal of automating the first-level filtering of resumes in a transparent, data-driven manner.

In this system, multiple candidate resumes are uploaded in PDF format through a web interface and processed automatically by the backend engine. The application extracts raw text from each PDF, identifies technical skills (for example Python, machine learning, web development frameworks), detects email and phone numbers, and estimates years of experience using pattern-based NLP and regular expressions. A job description provided by the recruiter or faculty is analyzed in parallel, and a simple yet effective scoring algorithm computes a job-match score for every candidate based on the overlap between resume skills and job requirements, producing a percentage score between 0 and 100. The results are presented in a ranked candidate table that allows users to quickly focus on the strongest profiles while still being able to review the underlying details whenever needed.

The project goes beyond plain ranking by including a small analytics dashboard so that it looks and behaves like a complete AI/ML application rather than a single script. Using Pandas for data handling and Plotly, Matplotlib, and WordCloud for visualization, the system generates a skill word cloud for the batch, bar charts of experience levels, and graphs of match scores across all resumes. These visual insights help recruiters, teachers and students understand which skills are most common, how experience is distributed, and how many candidates strongly match the given job role. The user interface is implemented in Streamlit with a multi-page layout (Home, Upload Resumes, Job Description, Ranking, Analytics), a professional background image, sidebar navigation, and metric cards summarizing total resumes, highest score and average score, giving the feel of a modern ATS-style dashboard.

TO INDUSTRY/ INSTITUTE CONTENT

Nature of business of the Industry / Institution

CODER ROOTS, is an ISO 9001:2015 & 27001:2022 certified IT Company, founded in 2022. We are specialized in web development, web design, software solutions, App development, digital marketing & branding, cyber security, internship programs etc.

This start-up was founded to help and increase business owner's efficiency through cuttingedge Digital Transformation of traditional working with the use of latest and advanced technological platforms. Since then, we are committed to provide end-to-end solutions through web development, wire-framing and highly skilled engineering execution. We prefer to form long-lasting strategic partnerships with clients by offering the solutions at affordable prices with timely deliveries and measurable business results.

We are a single point Software and IT internship company who can work as a guide in any of your project with the aim of cost saving without compromising the quality. We are aiming to become the market leader in providing Digital assistance to various industries in their transformation to increase productivity.

In addition to it, CODER ROOTS provide training & internship programs including 6 Weeks/6 Months Industrial Training, Project-Based Training, Corporate Training, and Job-Oriented Course Training while covering major IT trends such as Full Stack Development (MEAN/MERN), Flutter, Kotlin Android, Firebase, Python, React.js, Node.js, Core PHP, Laravel, Software Testing, Cloud Computing, DevOps, Data Science, Artificial Intelligence, Machine Learning, UI/UX Designing, Digital Marketing, WordPress, Linux, CCNP, CCNA Security, Network Security, Cyber Security, Java, Spring, Hibernate, C/C++, Photoshop, Adobe Illustrator, Figma, CorelDraw and many more.

By leveraging our extensive technological expertise, We, here at CODER ROOTS ensure that every project meets the highest standards of quality and efficiency.

Website: www.coderroots.com Support: hr@coderroots.co Contact+91-82641-23555

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION TO PROJECT

1.1 Purpose and Significance

- To design and implement an automated Resume Screening and Analysis System that can support recruiters and faculty during campus placements.
- To replace manual, time-consuming resume reading with an AI/NLP-based process that extracts skills and experience directly from PDF resumes.
- To provide a standardized, data-driven match score for every candidate with respect to a given job description, reducing subjectivity in evaluation.
- To give HR and coordinators a quick way to identify top-matching candidates while still allowing them to review detailed information if required.
- To build a practical AI/ML web application during 45-day industrial training and demonstrate the integration of Python, NLP and Streamlit in a single project.

1.2 Objectives

- Automatically read multiple resumes uploaded in PDF format and extract key information such as skills, email, phone number and approximate years of experience.
- Accept a job description as text input and analyze it to identify required technologies and keywords for the target role.
- Compute a numeric job-match score (0–100%) for each resume using skill overlap and simple rule-based scoring logic.
- Display all candidates in a ranked table with sorting, filtering and CSV export so that the highest-scoring profiles can be shortlisted quickly.
- Provide an analytics view with visualizations like skill word cloud, experience distribution and match score charts to understand batch-level trends.

- Design an attractive, multi-page Streamlit interface (Home, Upload Resumes, Job Description, Ranking, Analytics) that is easy to use for non-technical users.

1.3 Problem Definition

1.3.1 Overview

- Traditional resume screening in colleges is manual: faculty or HR read each resume, highlight important points and maintain marks in Excel sheets.

- With increasing number of applicants for software and AI/ML roles, manual screening becomes slow, repetitive and difficult to manage consistently.

- There is no integrated tool that can upload multiple resumes together, compare them with a specific job description, and instantly give a ranked list.

1.3.2 Key Challenges

- Time consumption: Screening tens or hundreds of resumes one by one delays the shortlisting process and increases workload on faculty and HR.

- Lack of consistency: Different evaluators may give different ratings to the same resume, leading to subjective and sometimes unfair decisions.

- Limited visibility: Manual methods do not provide overall insights such as which skills are most common in the batch or what percentage of students match a role.

- No automation: There is no automatic scoring engine to map resumes to job descriptions and generate comparable match scores.

- Difficult to reuse: Marks and comments stored in spreadsheets are difficult to reuse or update when job requirements change.

1.4 Scope of the Project

- The project focuses on software/IT roles where skills can be identified using keyword-based NLP (e.g., Python, machine learning, web development, databases).

- Core modules include PDF text extraction, skill and experience detection, job description analysis, scoring engine, candidate ranking and analytics dashboard.

- The system is targeted for use on a single machine within a college or training environment; it does not cover large-scale cloud deployment or integration with external ATS platforms.

- Advanced deep-learning or transformer-based semantic matching is outside the current scope but the architecture is kept flexible so that such models can be plugged in later.

- Within the 45-day industrial training period, this scope is sufficient to demonstrate understanding of AI/ML, NLP, data visualization and full-stack Python development in a realistic project.

CHAPTER 2

EXISTING SYSTEM

2.1 Current Methods of Resume Evaluation

- Manual Resume Reading:
  Faculty coordinators or HR representatives open each resume one by one, read the content, and manually judge whether the candidate is suitable for a particular job role.

- Spreadsheet-Based Shortlisting:
  Basic details like name, branch, CGPA and remarks are typed into Excel sheets; shortlisting decisions are taken based on these manually entered fields rather than structured skill analysis.

- Simple Keyword Search:
  Sometimes evaluators use the search feature inside a PDF viewer (Ctrl + F) to look for specific skills like “Python” or “Machine Learning”, but this is done resume-wise and not aggregated across the batch.

- Informal Faculty Discussion:
  Different teachers compare their lists verbally or through messages, consolidating shortlists based on experience and intuition instead of a uniform scoring model.

- One-time Evaluation:
  Once a shortlist is made for a particular company, the detailed reasoning and skill mapping are rarely stored; if another job profile comes, the process almost starts again from scratch.

2.2 Limitations of the Existing System

- No Real-Time or Bulk Processing:
  Manual reading of resumes cannot keep up when applications are large in number; there is no way to quickly re-evaluate the same set of resumes for a new job description.

- High Manual Effort and Time:
  Evaluators spend a lot of time scrolling through resumes, which reduces the time available for interviews, mentoring and other academic activities.

- Inconsistent and Subjective Evaluation:
  Different faculty members may weigh skills, projects and internships differently; two evaluators can rank the same candidate in different positions because there is no common scoring standard.

- Lack of Structured Skill Data:
  Skills and experience information remain locked inside PDF text; there is no central dataset showing which technologies are common or rare in the batch.

- No Visual Analytics or Trends:
  Existing methods do not provide charts for skill distribution, experience levels or suitability percentage for a particular role, making it hard to analyse strengths and gaps of the batch.

- Difficult to Reuse for Future Drives:
  Since most data is stored as free-text remarks or scattered Excel sheets, it is hard to reuse previous shortlists or quickly adapt them when a new company with slightly different requirements visits the campus.

2.2 Limitations of the Existing System

No Real-Time Data: Prices, ratings, and stock change rapidly, but manual tracking cannot keep pace.

- Scattered Information: Product details differ across platforms, making consistent comparison difficult.
- High Manual Effort: Comparing many models requires significant time and often leads to errors.
- No Trend Visualization: Users cannot see brand popularity, price ranges, or spec distributions clearly.
- Inconsistent Reviews: Reviews vary widely between platforms, causing confusion in evaluating product quality.
- Lack of a Unified Dashboard: No single system combines data scraping, cleaning, and visualization for all categories

CHAPTER 3

PROPOSED SYSTEM

The proposed system is an AI-based Resume Screening and Analysis web application developed using Python, Natural Language Processing (NLP) techniques and the Streamlit framework. It is designed to automate the first-level shortlisting of candidates by converting unstructured resume PDFs into structured, comparable information. Instead of reading each resume manually, the user uploads a set of PDF resumes along with the job description for a particular role, such as Python Developer or Machine Learning Engineer. The backend pipeline then extracts text from every resume, identifies important technical skills, estimates years of experience and collects contact details using pattern-based NLP and regular expressions. A rule-based scoring engine analyses the job description, finds key skills and compares them with each resume to generate a numeric job-match score between 0 and 100 for every candidate. On the front end, a modern Streamlit interface presents these results through clean tables, metric cards and visual analytics, so that the application behaves like a mini ATS dashboard suitable for academic and training demonstrations.

3.1 System Overview

- The system follows a simple but complete pipeline: Upload Resumes → Parse and Extract Data → Enter Job Description → Compute Match Scores → Display Ranking → Show Analytics.
- It is implemented as a multi-page Streamlit app with five main pages: Home, Upload Resumes, Job Description, Ranking and Analytics, all accessible from the sidebar.
- When resumes are uploaded, a PDF parser reads each file, extracts the textual content and passes it to helper functions that detect skills, email, phone number and approximate experience years.
- Candidate information is stored in in-memory data structures (lists and Pandas DataFrames), which makes it easy to sort, filter and export results without additional databases.
- The scoring module reads the job description, converts it to lowercase, searches for known skills and then calculates a score based on how many of these skills appear in each resume's skill list.
- Finally, the user interface presents a ranked table of candidates, summary metrics (total resumes, highest score, average score) and charts showing skill frequency, experience and match score distribution.

3.2 Key Features of the Proposed System

- Multi-Resume Upload:
  Users can upload multiple PDF resumes in a single step. This makes the system practical for batch-level evaluation where tens of candidates apply for the same role.

- Automatic Information Extraction:
  The backend automatically extracts raw text from each PDF, identifies technical skills based on a predefined skill dictionary, locates email and phone patterns and estimates years of experience using simple regular expressions. This removes repetitive manual reading and copying of details.
- Job Description Matching and Scoring:
  The job description is entered once in a text area and processed to identify required skills. For each candidate, the overlap between resume skills and JD skills is used to compute a match percentage, giving a clear numerical measure of suitability for the role.
- Candidate Ranking and CSV Export:
  All candidates are listed in descending order of match score, forming an instant shortlist that can be exported as a CSV file for sharing with HR, faculty or placement coordinators.
- Visual Analytics Dashboard:
  The Analytics page contains a word cloud that highlights the most frequent skills in the batch, a bar chart of experience per candidate and a bar chart of match scores. These visuals provide a quick understanding of how strong the batch is for a given job and which technologies dominate the resumes.
- User-Friendly and Attractive UI:
  The application uses sidebar navigation, a descriptive Home page, metric cards, and a background image to create a professional look and feel. This makes the system easy to demonstrate during viva and project presentations while clearly reflecting the 45-day AI/ML training effort.

3.3 Benefits of the Proposed System

- Faster Shortlisting and Reduced Workload:
  By automating text extraction, skill detection and scoring, the system can prepare a preliminary ranked list of candidates in seconds, significantly reducing manual workload compared to traditional screening.
- Consistent and Transparent Evaluation:
  Every candidate is scored using the same rule-based logic, which improves consistency and fairness. Recruiters can easily explain how scores are generated because the method is transparent and based on clearly defined skills.
- Improved Decision Support with Insights:
  Ranking tables combined with analytics charts help HR and faculty not only select individual candidates but also understand overall trends, such as which skills are popular, what is the typical experience range and how many students strongly match a given profile.
- Educational Value for Students:
  The project demonstrates how AI-driven resume screening works in simplified form and encourages students to design resumes that clearly highlight relevant skills and projects. It also showcases practical integration of Python, NLP and Streamlit into an end-to-end solution, which is valuable for learning and interviews.

3.4 Expected Outcomes

- A fully working Streamlit web application that performs automatic resume parsing, job-match scoring, candidate ranking and analytics, ready to be demonstrated as a major-level training project.
- A structured candidate dataset (filename, contact details, skills, experience, score) that can be tested with different job descriptions and can later be reused for training more advanced models such as semantic similarity or transformer-based matching.
- Complete project documentation including architecture diagrams, module descriptions, screenshots, testing results and a detailed report, providing evidence of the full software development lifecycle from requirement analysis to deployment-ready prototype.

Chapter 4 –

Feasibility Study

4.1 What is Feasibility Study

A feasibility study checks whether the proposed AI-Based Resume Screening and Analysis System can be developed and used successfully with the available technology, resources and users. It evaluates the project from technical, economic and operational angles before investing full effort into implementation, so that the solution is practical within a 45-day training period.

4.2 Technical Feasibility

- The system is built entirely with open-source technologies such as Python, spaCy, Pandas, Streamlit, Plotly and WordCloud, which are well supported and widely used for NLP and data apps.

- It runs on a normal laptop with at least 8 GB RAM and does not require GPU or special hardware, because the NLP and scoring logic are lightweight and rule-based.

- All components (PDF reading, text processing, scoring and visualization) are implemented as Python functions in a single Streamlit application, so integration and maintenance are technically straightforward.

4.3 Economic Feasibility

- No commercial licences or paid APIs are required; all libraries and tools used in the project are free, which makes the overall development cost almost zero apart from the existing laptop and internet connection.

- Deployment can be done locally for demonstrations or through free tiers of platforms like Streamlit Community Cloud or GitHub, which means the system can be shared without additional hosting charges.

4.4 Operational Feasibility

- The user interface is simple and menu-driven: users move step-by-step through Home, Upload Resumes, Job Description, Ranking and Analytics using the sidebar, which makes the system easy to operate even for non-technical HR staff.

- The workflow closely matches the existing manual process (collect resumes, read JD, shortlist candidates), so faculty and recruiters can adopt the system without changing their habits dramatically.

- Error messages and prompts (such as warnings when no resumes are uploaded or when job description is empty) make the application robust enough for use in real training and placement activities.

CHAPTER 5

SYSTEM REQUIREMENT SPECIFICATION

5.1 Functional Requirements

- The system shall allow the user to upload multiple resumes in PDF format and read the textual content of each file.

- The system shall extract key information from every resume, including skills, approximate years of experience, email ID and phone number, and store it in a structured dataset.

- The system shall accept a job description as text input and analyse it to identify required skills and keywords for the target role.

- The system shall compute a job-match score (0–100%) for every candidate based on the overlap between extracted skills and the job description, and display all candidates in a ranked table with options to view, sort and download the results as CSV.

- The system shall provide an Analytics page that shows a skill word cloud, charts of experience per candidate and charts of match scores to give batch-level insights.

5.2 Non-Functional Requirements

- The application should respond quickly for typical batch sizes (for example 20–50 resumes) on a normal laptop without noticeable lag.

- The user interface must be simple, menu-driven and consistent so that HR/faculty can operate it without programming knowledge.

- The system should handle invalid inputs gracefully, such as empty job descriptions or unreadable PDFs, by showing clear warning messages instead of crashing.

- The solution should be portable and reproducible using freely available tools (Python 3.11, open-source libraries) so that it can be run on different machines in the lab or at home.

5.3 Hardware and Software Requirements

- Hardware: Standard PC or laptop with at least Intel i5-class processor (or equivalent), 8 GB RAM, 50 GB free disk space and internet/browser for running the Streamlit app.

- Software: Windows 10/11 (or equivalent OS), Python 3.11, required Python libraries (Streamlit, PyPDF2, spaCy, Pandas, Plotly, Matplotlib, WordCloud), and a code editor such as VS Code for development and maintenance.

CHAPTER 6

TOOLS AND TECHNOLOGIES USED

6.1 Technologies Used

6.1.1 PYTHON

Fig 6.1

6.1.1.1 What is python?

Python is a high-level, interpreted, and object-oriented programming language known for its simplicity and readability. Its dynamic typing, built-in data structures, and support for modular programming make it ideal for rapid application development. Python allows the use of modules and packages, making programs easy to structure and reuse. The Python interpreter and standard library are freely available across all major platform

6.1.2 PANDAS

Fig. 6.2

6.1.2.1 What is Pandas?

Pandas is one of the most widely used Python libraries for data manipulation and data analysis. It provides powerful tools to handle structured datasets efficiently and is a core component of the Python data science ecosystem. It provides specialized data structures—mainly Series and DataFrame—designed to handle tabular, time-series, and statistical data.The name “pandas” comes from panel data, reflecting its ability to work with multidimensional datasets.

6.1.3 PyPDF2 – PDF Parsing Library

PyPDF2 is a Python library used in this project to read and extract text from resume files stored in PDF format. It allows the system to open each uploaded PDF, iterate over its pages and collect the raw text content without requiring any external tools or manual conversion. This extracted text becomes the input for the NLP pipeline, where skills, contact details and experience information are detected and processed for scoring and analysis.

6.1.5.1 What is Pandas?

Pandas is a powerful Python data manipulation and analysis library. It provides fast and flexible data structures like DataFrames, which are used to clean, organize, and process scraped product data. Pandas simplifies tasks such as handling missing values, converting data types, filtering records, and performing aggregations. In this project, Pandas plays a crucial role in preparing structured datasets for visual analysis, ensuring that all product information is consistent, accurate, and ready for visualization.

6.2 TOOLS AND EDITORS USED

6.2.1 Jupyter Notebook

Fig. 6.7

6.2.1.1 What is a Jupyter Notebook?

Jupyter Notebook provides a browser-based interface that lets users run code interactively and document their work in a single environment. It supports multiple programming languages through different kernels, making it flexible for various computational tasks.

6.2.2 VSCODE

Fig. 6.8

6.2.2.1 What i sVScode?

Visual Studio Code (VS Code) is a lightweight yet powerful open-source code editor developed by Microsoft. It works on Windows, macOS, Linux, and even web browsers. VS Code supports debugging, syntax highlighting, IntelliSense, code refactoring, Git integration, and a wide range of extensions, making it one of the most popular development tools today.

6.2.3 STREAMLIT

Fig 6.10

6.2.3.1 What is Streamlit

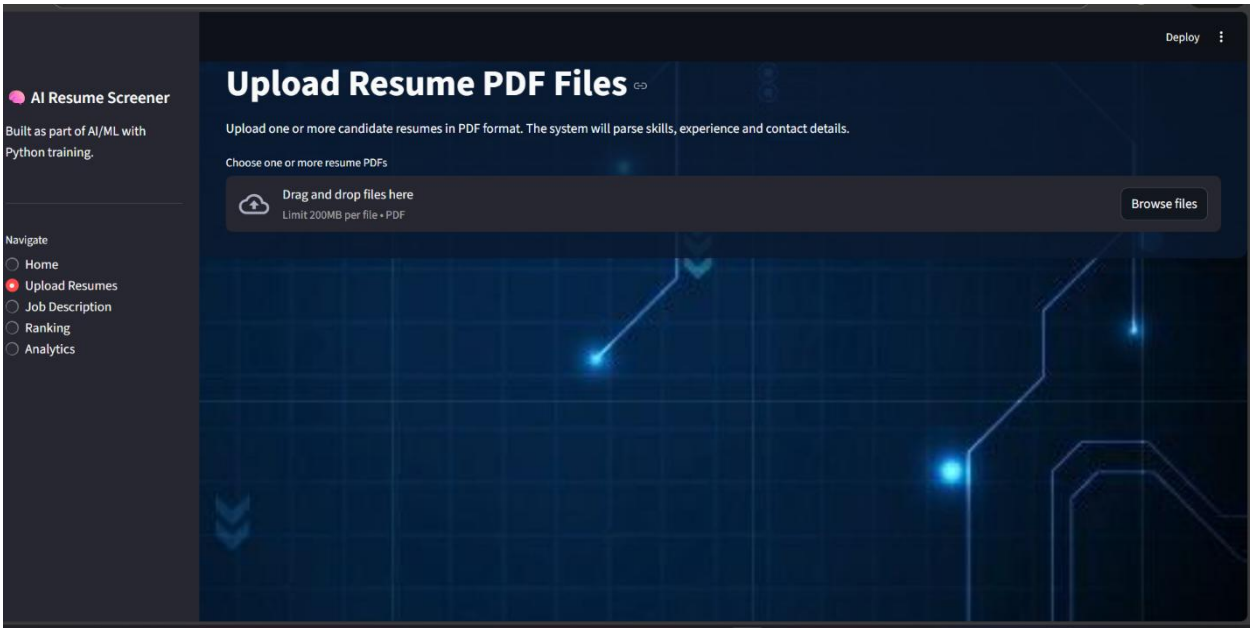
Streamlit is an open-source Python library that turns Python scripts into interactive web applications. By writing a simple Python script, users can create dashboards, data apps, and ML prototypes. Streamlit automatically refreshes the app whenever a user interacts with it and runs on a local server that opens in a web browser.

CHAPTER 7

SCREENSHOTS

1. **Home Screen :**

2. **Features :** Screen displaying main features.



CHAPTER 8

SYSTEM TESTING

8.1 Introduction to Testing

Testing is performed to verify that the AI-Based Resume Screening and Analysis System works correctly from end to end and that each module (PDF parsing, NLP extraction, scoring, and Streamlit interface) behaves as expected. It helps ensure that resumes are read without errors, match scores are calculated correctly for different job descriptions and the dashboard displays accurate information to the user.

8.2 Testing Strategies Used

- **Unit Testing** – Individual functions such as text extraction, skill detection, experience calculation and score computation were tested with small sample inputs to confirm correct outputs.
- **Integration Testing** – The complete flow from uploading resumes to viewing rankings and analytics was tested to make sure data passes correctly between modules and pages.
- **Black Box / Functional Testing** – The system was used like an HR tool (without looking at internal code) to check that each page behaves according to the specified functional requirements.
- **Real-Data Testing** – Multiple sample resumes with different skills and job descriptions were tried to see whether high-matching profiles consistently appear at the top of the ranking.

8.3 Testing Methods and Results (Sample Table)

Test Type	**Description / Test Cases**	**Status**
Unit Testing	Verified PDF reading, text extraction, email/phone regex and experience parsing on sample resumes.	Pass
Scoring Function Test	Checked match score calculation by manually comparing resume skills with job description keywords.	Pass
Integration Testing	Tested full pipeline: Upload Resumes → JD Input → Score Calculation → Ranking Table → Analytics Charts.	Pass
Functional Testing	Ensured buttons, file uploader, navigation, warnings (no resumes / empty JD) and CSV export work correctly.	Pass

Test Type	Description / Test Cases	Status
Edge / Error Handling	Tried corrupt / non-PDF files, empty resumes and missing fields to verify that the system shows safe errors.	Handled

8.4 Testing Scenarios

- Uploading a mix of strongly matching, weakly matching and unrelated resumes for the same job description to see if the ranking order is logical.
- Changing the job description (e.g., from “Python Developer” to “Web Developer”) and confirming that the same resumes receive different scores based on new skill requirements.
- Uploading only a single resume to verify that the system still works and generates analytics without failures.

8.5 Output Validation

- Randomly selected candidates were checked manually by reading their resumes and comparing them with the job description; their manually estimated suitability was consistent with the automatically generated scores in the table.
- Word cloud and bar charts were cross-checked with underlying DataFrame values to ensure that the height of bars and size of words accurately reflect the frequency of skills and scores stored in the system.

CHAPTER 9

CONCLUSION AND FUTURE

9.1 Conclusion

The AI-Based Resume Screening and Analysis System successfully demonstrates how AI and NLP can automate the first-level screening of multiple resumes for a specific job role. By reading PDF resumes, extracting skills and experience, matching them against a job description and generating a job-match score, the system reduces manual effort for recruiters and faculty while making the evaluation process more consistent and transparent. The Streamlit dashboard, with its ranking table, skill word cloud and analytical charts, converts raw candidate data into clear visual insights and shows that open-source tools like Python, spaCy, Pandas and Streamlit are sufficient to build a complete ATS-style prototype within a short 45-day industrial training period.

9.2 Future Scope

The current version uses rule-based keyword matching, which can be extended with advanced machine learning and deep-learning models for semantic similarity between resumes and job descriptions. Future work can include support for DOCX/online profiles, weighting of mandatory and optional skills, sentiment analysis of project descriptions or recommendations, and scheduled batch processing for large-scale campus drives. The system can also be deployed on cloud platforms so that placement cells, HR teams and students can access it from anywhere, turning this training project into a practical tool for real-world recruitment scenarios.

REFERENCES

- Python Software Foundation, “Python 3 Documentation.” Available at: https://docs.python.org/
- Explosion AI, “spaCy Usage Documentation.” Available at: https://spacy.io/usage
- Streamlit Inc., “Streamlit Docs – Build data apps in Python.” Available at: https://docs.streamlit.io/
- Pandas Development Team, “Pandas Documentation – Python Data Analysis Library.” Available at: https://pandas.pydata.org/docs/
- Plotly Technologies Inc., “Plotly Express – High-Level Interface for Interactive Graphing.” Available at: https://plotly.com/python/plotly-express/
- WordCloud Project, “wordcloud – A Little Word Cloud Generator in Python.” Available at: https://pypi.org/project/wordcloud/
- Peerbits, “What is AI Resume Screening? Benefits & Tips Explained.”
- Leoforce, “Guide to AI Resume Screening – Tools, Process and Algorithms.”