AI-Powered Resume Analyzer

A course project report submitted in partial fulfilment of the requirement of

AI ASSISTED CODING

by

G. SANJAN SAH	-	2503A52L20
O.PANINDRA KUMAR	-	2503A52L19
G.ASHOK	-	2503A52L21

Under the guidance of

B.Raju

Asst. Professor, Department of CS & AI

SR University, Warangal

ABSTRACT

The AI-Powered Resume Analyzer is an intelligent web-based system designed to evaluate and enhance the quality of resumes using Natural Language Processing (NLP) and Artificial Intelligence (AI). The project enables users to upload resumes in PDF or DOCX formats, automatically extracts the textual content, and analyzes it for grammar accuracy, readability, formatting structure, and job-role relevance. By integrating spaCy for linguistic processing and the OpenAI API for advanced language understanding, the system provides detailed, actionable feedback to help users create professional and competitive resumes.

This tool addresses common challenges faced by job seekers—such as unclear descriptions, poor formatting, and missing skills—while improving their chances of success in both human and ATS (Applicant Tracking System) evaluations. The application serves as a practical solution for students, professionals, and recruiters, and acts as a strong demonstration of the integration of AI-driven text analysis within modern web applications.

1. Introduction

The AI-Powered Resume Analyzer is an intelligent web-based application designed to help job seekers evaluate and enhance their resumes using modern Natural Language Processing (NLP) and Artificial Intelligence (AI) techniques. As hiring processes become increasingly competitive, applicants must present well-structured, clear, and job-relevant resumes. This tool automates the evaluation process by analyzing grammar, clarity, formatting, overall structure, and suitability for specific job roles.

The system uses a combination of **Flask** for the web interface, **spaCy** for the foundational NLP tasks, and the **OpenAI API** to generate deep and contextual feedback. The goal is to empower users with actionable insights that improve their chances of passing both human screening and automated ATS (Applicant Tracking System) filters.

This documentation provides a comprehensive breakdown of the system architecture, modules, workflows, challenges, testing procedures, and future enhancements. It is designed to serve as a guide for developers, students, and organizations looking to deploy or expand similar AI-driven tools.

---

2. Project Objectives

The primary objective of the AI-Powered Resume Analyzer is to build an interactive system that:

1. Allows users to upload resumes in PDF or DOCX formats.
2. Extracts readable text from the uploaded files.
3. Applies NLP methods to understand and analyze resume content.
4. Utilizes AI models to evaluate grammar, clarity, and job-role alignment.
5. Generates detailed, structured feedback that users can apply to improve their resumes.

Secondary Objectives

- Provide a clean, intuitive front-end interface.
- Support multiple job roles and personalize feedback.
- Test the tool against diverse resume datasets.
- Ensure consistent and meaningful output across different file formats.

---

3. System Architecture Overview

The system is composed of three primary layers:

3.1 User Interface Layer

- Built using HTML, CSS, and Flask templating (Jinja2).
- Allows the user to:
    - Upload a resume (PDF/DOCX).
    - Select a job role for targeted analysis.
    - View structured feedback.

3.2 NLP + AI Processing Layer

- Performs text extraction.
- Applies spaCy pipelines for fundamental NLP tasks.
- Calls the OpenAI API for advanced contextual feedback.

3.3 Output Generation Layer

- Formats the AI-driven suggestions.
- Presents results in categorized sections:
    - Grammar & Language Quality
    - Readability & Clarity
    - Structural/Formatting Suggestions
    - Job Role–Specific Feedback
    - Missing Skills

- Overall Assessment

---

4. Technologies Used

Component	Technology
Backend Framework	Flask (Python)
AI Model	OpenAI GPT APIs
NLP Engine	spaCy
PDF Processor	pdfplumber
DOCX Processor	python-docx
Frontend	HTML, CSS, JS
Testing	Manual + multiple resume datasets

---

5. Detailed Implementation Steps

5.1 Step 1 — Creating the Flask Web Interface

The Flask server manages the application's routing and integrates both the frontend and backend.

Key Functionalities:

- / — Displays the resume upload page.
- /analyze — Processes file input and returns report.
- Error handling for unsupported file types.

Example Folder Structure:

```
project/
 ├── app.py
 ├── static/
 │    ├── style.css
 ├── templates/
 │    ├── index.html
 │    ├── result.html
 └── utils/
     ├── extractor.py
     ├── analyzer.py
```

5.2 Step 2 — Text Extraction from Resumes

PDF Extraction (pdfplumber)

PDFs often contain complex layouts. pdfplumber allows controlled extraction of raw text.

Challenges:

- Misaligned columns
- Headers and footers mixed with body text

DOCX Extraction (python-docx)

Simple extraction through paragraph loops.

Preprocessing Steps:

- Removing extra spaces
- Normalizing newline characters
- Removing non-ASCII or unnecessary symbols

5.3 Step 3 — NLP Analysis using spaCy and OpenAI

spaCy Module Responsibilities:

- Sentence segmentation
- Tokenization
- Named Entity Recognition (NER)
- Keyword extraction (skills, roles)

OpenAI API Responsibilities:

- Grammar and fluency scoring
- Clarity and tone analysis
- Format and structure evaluation
- ATS optimization suggestions
- Job role–specific skill gap identification

Prompt Example:

Analyze the following resume text for grammar, clarity, structure, and relevance to the job role of “Software Engineer”. Provide detailed suggestions.

5.4 Step 4 — Generating Feedback

Feedback is categorized into structured sections for readability.

Categories:

1. Grammar & Language Issues
2. Readability & Clarity
3. Formatting & Structure
4. Role-Specific Feedback
5. Missing Hard/Soft Skills
6. Overall Summary

5.5 Step 5 — Testing the System

Testing is performed using 15+ resumes with varying:

- Experience levels (entry → senior)
- Industrial sectors
- Writing styles
- Lengths and formats

Evaluation Metrics:

- Accuracy of text extraction
- Accuracy of skill detection

- Relevance of AI suggestions
- Response consistency

---

6. Module Breakdown

6.1 File Extraction Module

Handles file validation and text extraction.

6.2 NLP Preprocessing Module

Cleans and prepares text for analysis.

6.3 AI Feedback Generator Module

Creates structured feedback using advanced prompts.

6.4 Renderer Module

Formats results into HTML templates.

---

7. User Interface Details

7.1 Upload Page

Elements:

- File upload field
- Job role dropdown
- Submit button

7.2 Results Page

Displays categorized feedback with collapsible sections.

---

8. Error Handling

Types of errors handled:

- Incorrect file type
- Empty file
- Malformed PDF
- API failure

---

9. Performance Considerations

- Caching API responses
- Reducing unnecessary API calls
- Handling large resume files gracefully

---

10. Security Considerations

- Input validation
- Secure file handling (temporary storage)
- Rate limiting for API endpoints

---

11. Improvements and Enhancements

11.1 Short-Term Enhancements

- Resume scoring system

- Export feedback as PDF or DOCX

11.2 Long-Term Enhancements

- Full ATS simulation
- AI-powered resume rewriting
- Multi-language support
- Integration with job portals

---

12. Limitations

- AI interpretations vary slightly based on input phrasing.
- Complex PDFs still pose extraction challenges.
- Accuracy depends on OpenAI API quality.

---

13. Conclusion

The AI-Powered Resume Analyzer is a robust and scalable system that leverages modern NLP and AI technologies to significantly improve resume quality. By analyzing grammar, clarity, formatting, and job-role alignment, it helps users produce professional and competitive resumes.

This documentation outlines the full architecture, workflow, implementation steps, and future directions—

serving as a complete reference for development and enhancement.

---

14. References

- Flask Documentation
- spaCy Documentation
- OpenAI API Documentation
- pdfplumber Documentation
- python-docx Documentation

---

15. Appendix

Includes sample prompts, testing resumes, and architecture diagrams (to be added if required).