AI Resume Analyzer: A Smart NLP-Powered Resume Analysis and Career Recommendation System

Satish L Datar[1], Rohit S Rathod[2], Nachiket R Shrikhande[3]
[1,2,3]*Department of Electronics & Computer Engineering, Shreeyash College of Engineering & Technology, Aurangabad*

***Abstract*—The AI Resume Analyzer is an intelligent web application developed using Python and Streamlit that automates the process of resume analysis, career field prediction, and skill recommendation. The system leverages Natural Language Processing (NLP) techniques to parse PDF resumes, extract key information such as skills, experience level, and contact details, and then generates actionable recommendations including missing skills, relevant courses, interview questions, and LinkedIn profile suggestions.**

The application features a dual-panel architecture comprising a User Module for real-time resume feedback and an Admin Module for analytics and user management. An integrated ATS (Applicant Tracking System) Score Checker evaluates keyword compatibility against job descriptions, while a Cover Letter generation module and a multi-dimensional feedback system further enhance the platform's utility. The system stores all user interaction data in a MySQL database and presents visual analytics via Plotly charts. This paper discusses the system design, modules, implementation details, complete source code architecture, and future enhancement opportunities.

***Index Terms*—Resume Analysis, NLP, ATS Score, Streamlit, Career Recommendation, Skill Extraction, Python, Machine Learning Portfolio**

I. INTRODUCTION

The rapid digitization of recruitment processes has placed significant pressure on job seekers to craft resumes that satisfy both human recruiters and automated Applicant Tracking Systems (ATS). Studies suggest that over 75% of resumes are rejected by ATS tools before reaching a hiring manager, largely due to poor keyword optimization, formatting issues, and missing section content.

The AI Resume Analyzer addresses this challenge by providing an automated, intelligent system that evaluates resumes holistically — analyzing content quality, keyword density, structural completeness, and career-field alignment. Built as a Streamlit web application, it offers a clean, multi-page interface accessible via any browser, with zero configuration required from end users.

This project was developed as a B.Tech final-year capstone project with a focus on practical deployability. The system has already tracked over 37,000 visitor interactions (as seen in the live counter), validating its real-world utility.

1.1. Problem Statement

Job seekers, particularly fresh graduates and career-switchers, frequently lack insight into:

- Whether their resume contains the right keywords for target job roles
- What skills are missing relative to industry standards for their domain
- How their resume scores against ATS parsing algorithms
- How to improve their LinkedIn profile to attract recruiters
- What interview questions they might face based on their skill set

1.2. Objectives

- Automatically extract and classify resume content using NLP
- Predict the most suitable career field based on detected skills
- Recommend additional skills, courses, and certifications
- Generate ATS compatibility scores with detailed keyword analysis
- Provide LinkedIn headline and About section suggestions

- Enable admins to monitor usage analytics and feedback trends

II. SYSTEM ARCHITECTURE

The AI Resume Analyzer follows a layered multi-module architecture. The front-end presentation layer is handled entirely by Streamlit. The processing layer uses Python NLP libraries. The persistence layer relies on MySQL for structured data storage and CSV files for lightweight feedback tracking.

2.1. Architecture Overview

Layer	Technology	Responsibility	Key Libraries
Presentation	Streamlit 1.x	UI, navigation, state	streamlit
Processing	Python 3.10+	NLP, PDF parsing, scoring	spacy, pdfminer, re
Database	MySQL + CSV	User data, feedback	pymysql, pandas
Visualization	Plotly	Charts, analytics	plotly.express
Export	ReportLab	PDF report generation	reportlab

2.2. Application Pages / Modules

The application is structured around six navigable pages, selectable from the sidebar dropdown:

Page	Description	Key Features
User	Main resume analysis interface	PDF upload, skill extraction, recommendations
ATS Score	ATS keyword compatibility checker	Score, grade, keyword gap analysis
Cover Letter	AI cover letter generator	Job-tailored cover letter drafting
Feedback	User rating and comments	Star rating, comment storage, pie chart
About	Tool documentation	Usage instructions, admin credentials info
Admin	Password-protected analytics dashboard	User data table, feedback analytics, charts

III. USER MODULE — DETAILED IMPLEMENTATION

3.1. User Registration & Resume Upload

The User page collects three mandatory fields — Name, Email, and Mobile Number — before allowing resume upload. The resume must be submitted as a PDF (max 200 MB). Upon upload, Streamlit renders a live preview of the PDF inline using its native file viewer.

```
# app.py — User Page Core Logic
import streamlit as st
import pdfminer
from pdfminer.high_level import extract_text
import re, random, string, datetime
import pymysql

def get_session_token():
return ''. join (random.choices(string.ascii_letters + string.digits, k=12))

def show_user_page():
st.image('logo.png')
name = st.text_input('Name*')
email = st.text_input('Mail*')
mobile = st.text_input('Mobile Number*')
uploaded_file = st.file_uploader('Choose your Resume', type='pdf')

if uploaded_file is not None:
st.markdown('---')
resume_text = extract_text(uploaded_file)
analyze_resume(resume_text, name, email, mobile)
```

3.2. Resume Parsing & Information Extraction

After PDF text extraction via pdfminer, the system applies regex patterns and keyword matching to extract structured information:

```
def extract_resume_info(text):
info = {}
# Name: first non-empty line heuristic
lines = [l.strip() for l in text.split('\n') if l.strip()]
info['name'] = lines [0] if lines else 'Unknown'

# Email extraction
email_match = re.findall(r'[\w.-]+@[\w.-]+\.[a-z]{2,}', text, re.IGNORECASE)
```

```
info['email'] = email_match[0] if email_match else ''

# Phone extraction
phone_match = re. findall(r'[\+]?[(]?[0-9]{3}[)]?[-\s.]?[0-9]{3}[-\s.]?[0-9]{4,6}', text)
info['phone'] = phone_match[0] if phone_match else ''

# Degree detection
degree_keywords = ['B.Tech', 'B.E.', 'M.Tech', 'MCA', 'BCA', 'MBA', 'PhD', 'B.Sc', 'M.Sc']
info['degree'] = [d for d in degree_keywords if d.lower() in text.lower()]

# Page count approximation
info['pages'] = max(1, text.count('\x0c') + 1)
return info
```

3.3. Experience Level Detection

The system classifies users into three experience levels — Fresher, Intermediate, and Experienced — based on the number of years mentioned and keyword density:

```
def detect_experience_level(text):
text_lower = text.lower()
year_mentions = re.findall(r'(\d+)\+?\s*year', text_lower)
total_years = sum(int(y) for y in year_mentions) if year_mentions else 0

if total_years == 0:
return 'Fresher'
elif total_years <= 3:
return 'Intermediate'
else:
return 'Experienced'
```

3.4. Skill Detection & Career Field Prediction

Skills are matched against curated keyword dictionaries per domain. The domain with the highest skill-match count is predicted as the user's target career field:

```
DOMAIN_SKILLS = {
'Data Science': ['tensorflow','keras','pytorch','machine learning','deep learning',
'flask','streamlit','pandas','numpy','matplotlib','seaborn',
'Scikit-learn','data visualization','sql','tableau','nlp'],
'Web Development': ['react','angular','vue','node','django','flask','html','css',
'javascript','typescript','mongodb','postgresql','rest api'],
'Android': ['android','java','kotlin','xml','flutter','dart','firebase'],
'IOS': ['ios','swift','xcode','objective-c','core data','cocoa'],
'UI-UX': ['figma','adobe xd','sketch','wireframe','prototyping','user research'],
}

def predict_career_field(skills_list):
scores = {}
skills_lower = [s.lower() for s in skills_list]
for domain, keywords in DOMAIN_SKILLS.items():
scores[domain] = sum (1 for k in keywords if k in skills_lower)
return max (scores, key=scores.get) if any(scores.values()) else 'General'
```

3.5. Skill Recommendation Engine

Based on the predicted career field, the system recommends skills the user is currently missing from the domain-standard skill set:

```
def recommend_skills(current_skills, predicted_field):
domain_skills = DOMAIN_SKILLS.get(predicted_field, [])
current_lower = [s.lower() for s in current_skills]
missing = [s for s in domain_skills if s not in current_lower]
return missing [:15]  # Return top 15 missing skills
```

3.6. Resume Tips & Score Calculation

The resume writing score (0–100) is computed based on the presence of key resume sections. Each section adds a fixed weight:

Resume Section	Points	Detection Method
Objective / Summary	15	Keyword: 'objective', 'summary', 'profile'
Education Details	15	Keyword: 'education', 'university', 'college'
Work Experience	20	Keyword: 'experience', 'worked at', 'employed'
Skills Section	20	Keyword: 'skills', skill list detection

Projects	15	Keyword: 'project', 'built', 'developed'
Achievements	5	Keyword: 'achievement', 'award', 'recognition'
Certifications	5	Keyword: 'certified', 'certification', 'course'
Hobbies / Interests	5	Keyword: 'hobbies', 'interests', 'passion'

3.7. LinkedIn Profile Generator

The system auto-generates a professional LinkedIn headline and About section based on the detected job role and skills:

```
def generate_linkedin_suggestions(name, role, skills):
top_skills = ', '. join (skills [:5])
headline = f'{role} | {top_skills}'
about = (
f'{role} is a data science with expertise in {top_skills}. '
f'Driven by strong collaboration, problem solving, and results-oriented work, '
f'they deliver measurable outcomes for employers. '
f'Focus on achievements and impact in every role, with a commitment to '
f'continuous learning and growth.'
)
return headline, about
```

3.8. Interview Question Generator

The system contains a curated dictionary of technical interview questions per skill, plus a set of standard HR questions:

```
TECHNICAL_QA = {
'python': ['What is the difference between list and tuple in Python?',
'Explain Python decorators with an example.',
'What is the Global Interpreter Lock (GIL)?'],
'Machine learning': ['What is the bias-variance tradeoff?',
'Explain the difference between supervised and unsupervised learning.',
'What is cross-validation and why is it important?'],
'sql': ['What is the difference between INNER JOIN and LEFT JOIN?',
'Explain ACID properties in databases.',
'What is database normalization?'],
}

HR_QUESTIONS = [
'Tell me about yourself.',
'What are your strengths and weaknesses?',
'Why should we hire you?',
'Describe a challenging situation you handled.',
'Where do you see yourself in 5 years?'
]
```

IV. ATS SCORE CHECKER MODULE

4.1. Overview

The ATS Score Checker (Page 2) allows users to paste resume text or upload a PDF, enter a target job title or keywords, and receive a detailed ATS compatibility report. It evaluates keyword presence, section coverage, and formatting.

4.2. ATS Score Algorithm

```
def calculate_ats_score(resume_text, target_keywords, job_description=''):
resume_lower = resume_text.lower()
keywords = [k.strip().lower() for k in target_keywords.split(',')]

# Keyword matching
found = [k for k in keywords if k in resume_lower]
missing = [k for k in keywords if k not in resume_lower]
keyword_score = (len(found) / len(keywords)) * 100 if keywords else 100

# Section coverage check
sections = {
'contact_info': any(w in resume_lower for w in ['email','phone','mobile','linkedin']),
'work_experience': any(w in resume_lower for w in ['experience','worked','employed']),
'education': any(w in resume_lower for w in ['education','university','degree','b.tech']),
'skills': 'skills' in resume_lower,
}
section_score = (sum(sections.values()) / len(sections)) * 100

# Weighted final score
final_score = int(0.7 * keyword_score + 0.3 * section_score)

# Assign grade
```

```
grade = 'A' if final_score >= 90 else 'B' if final_score >= 75 else 'C' if final_score >= 60 else 'D'

return {
'score': final_score,
'grade': grade,
'found_keywords': found,
'missing_keywords': missing,
'sections': sections,
'format_issues': detect_format_issues(resume_text)
}
```

4.3. Format Issue Detection

```
def detect_format_issues(text):
issues = []
if len(text) < 200:
issues.append('Resume text appears too short — may have extraction issues')
if not re.search(r'[\w.-]+@[\w.-]+', text):
issues.append('No email address detected')
if text.count('|') > 20:
issues.append('Excessive pipe characters may confuse ATS parsers')
if re.search(r'\.(jpg|png|jpeg)', text, re.IGNORECASE):
issues.append('Image references found — avoid images in ATS resumes')
return issues if issues else []
```

V. FEEDBACK MODULE

5.1. Feedback Collection

The Feedback page allows users to rate the application from 1 to 5 and optionally leave a comment. Auto-filled Name and Email fields are carried over from the User session. Submissions are stored in MySQL and displayed as a pie chart and comment table.

```
def show_feedback_page():
name = st.session_state.get('user_name', '')
email = st.session_state.get('user_email', '')

name_input = st.text_input('Name', value=name)
email_input = st.text_input('Email', value=email)
rating = st.slider('Rate Us From 1 - 5', 1, 5, 3)
comment = st.text_input('Comments')

if st.button('Submit'):
save_feedback_to_db(name_input, email_input, rating, comment)
st.success('Thanks! Your Feedback was recorded.')

# Display past ratings
df = load_feedback_from_db()
st.subheader("Past User Rating's")
fig = px.pie(df, names='Feed', title='Chart of User Rating Score From 1 - 5')
st.plotly_chart(fig)
st.subheader("User Comment's")
st.dataframe(df[['User', 'Comment']])
```

VI. ADMIN MODULE

6.1. Authentication

The admin page is protected by a hardcoded credential check. Login credentials are: username = admin, password = admin@resume-analyzer. In a production environment, these should be stored as hashed environment variables.

```
ADMIN_CREDENTIALS = {'admin': 'admin@resume-analyzer'}

def show_admin_login():
st.success('Welcome to Admin Side')
username = st.text_input('Username')
password = st.text_input('Password', type='password')
if st.button('Login'):
if ADMIN_CREDENTIALS.get(username) == password:
st.session_state['admin_logged_in'] = True
show_admin_dashboard()
else:
st.error('Invalid credentials!')
```

6.2. Admin Dashboard Features

- User's Data Table — complete log with ID, Token, IP Address, Name, Email, Mobile, Predicted Field, Timestamp
- Download Report — CSV export of all user data
- User's Feedback Data — full feedback table with ratings and comments
- User Rating's Pie Chart — visual distribution of ratings 1–5
- Predicted Field Pie Chart — breakdown of career domains detected
- User Experience Level Pie Chart — Fresher / Intermediate / Experienced distribution

6.3. Database Schema

```
-- MySQL Database: resume_analyzer

CREATE TABLE user_data (
id          INT AUTO_INCREMENT PRIMARY KEY,
token       VARCHAR (50),
ip_address VARCHAR (50),
name        VARCHAR (100),
email       VARCHAR (100),
mobile      VARCHAR (20),
predicted_field VARCHAR (50),
experience_level VARCHAR (20),
resume_score INT,
timestamp                DATETIME        DEFAULT
CURRENT_TIMESTAMP
);

CREATE TABLE feedback (
id        INT AUTO_INCREMENT PRIMARY KEY,
name      VARCHAR (100),
email     VARCHAR (100),
rating    INT,
comment   TEXT,
timestamp                DATETIME        DEFAULT
CURRENT_TIMESTAMP
);
```

VII. TECHNOLOGY STACK

Category	Library / Tool	Purpose
Web Framework	Streamlit 1.x	UI rendering, navigation, state management
PDF Parsing	pdfminer.six	Text extraction from uploaded PDF resumes
NLP / Regex	re (built-in)	Pattern matching for emails, phones, keywords
Data Processing	Pandas, NumPy	Dataframe manipulation, analytics
Visualization	Plotly Express	Pie charts, bar charts, analytics dashboard
Database	PyMySQL + MySQL	Persistent user and feedback data storage
Export	ReportLab	PDF report download for admin
Environment	Python 3.10+	Core runtime
Deployment	localhost / Streamlit Cloud	Development and production hosting

VIII. DATABASE CONNECTIVITY

```
import pymysql

def get_db_connection():
return pymysql.connect(
host='localhost',
user='root',
password='your_password',
db='resume_analyzer',
charset='utf8mb4',
cursorclass=pymysql.cursors.DictCursor
)

def save_user_data(token, ip, name, email, mobile,
field, level, score):
conn = get_db_connection()
try:
with conn.cursor() as cursor:
sql = '''INSERT INTO user_data
(Token, ip_address, name, email, mobile,
predicted_field,
experience_level, resume_score)
VALUES (%s,%s,%s,%s,%s,%s,%s,%s)'''
cursor.execute(sql, (token, ip, name, email, mobile,
field, level, score))
conn.commit()
finally:
conn.close()

def load_user_data():
conn = get_db_connection()
try:
with conn.cursor() as cursor:
cursor.execute('SELECT * FROM user_data ORDER
BY id DESC')
return cursor.fetchall()
finally:
conn.close()
```

IX. PROJECT FILE STRUCTURE

```
AI-Resume-Analyzer/
├── app.py                # Main Streamlit application
entry point
├── requirements.txt      # Python dependencies
├── logo.png              # App header logo / banner
├── Courses.py            # Course recommendation
data
├── pages/
```

```
│   ├── 01_User.py          # User resume analysis page
│   ├── 02_ATS_Score.py     # ATS Score Checker page
│   ├── 03_Cover_Letter.py # Cover Letter generator
│   ├── 04_Feedback.py      # Feedback collection page
│   ├── 05_About.py         # About/documentation page
│   └── 06_Admin.py         # Admin dashboard (password-protected)
├── utils/
│   ├── resume_parser.py    # PDF parsing & info extraction
│   ├── skill_engine.py     # Skill detection & recommendation
│   ├── ats_checker.py      # ATS score calculation
│   ├── linkedin_gen.py     # LinkedIn suggestion generator
│   └── db_utils.py         # MySQL CRUD operations
└── data/
    ├── skills_db.json      # Domain-skills mapping dictionary
    └── interview_qa.json   # Technical Q&A per skill
```

X. RESULTS & OBSERVATIONS

10.1.Live Usage Statistics

As observed in the application screenshots, the system has tracked over 37,073 visitor sessions since deployment. The admin dashboard logs show users from multiple IP addresses including both local (127.0.0.1) and remote network addresses, confirming the application operates beyond single-machine testing.

10.2.ATS Score Accuracy

In testing with a Data Scientist resume against a single keyword target ('Python'), the system correctly identified the keyword and returned a 100/100 ATS score with Grade A. Multi-keyword testing showed accurate detection of missing skills and appropriate score reduction proportional to keyword gap.

10.3.User Feedback Distribution

From collected feedback data (6 submissions at time of documentation), ratings distribution shows a balanced spread across scores 2, 3, 4, and 5 (25% each), with comments including 'Good', 'Excellent', 'Poor', demonstrating diverse user experiences and areas for improvement.

10.4.Career Field Prediction

Admin analytics reveal that the system successfully categorizes users across Data Science, Web Development, and N/A (unclassified) buckets, with each representing 25% of the tested user base — validating the multi-domain skill detection framework.

XI. FUTURE ENHANCEMENTS

- Integrate Claude/GPT API to generate personalized, job-description-aware cover lettersAI-Powered Cover Letter Generation:
- Connect to job APIs (Indeed, LinkedIn) to match resume against live job postingsReal-Time Job Matching:
- Replace regex-based parsing with a BERT/spaCy NER model for higher accuracyDeep Learning Skill Extraction:
- Allow users to save and compare multiple resume versions over timeResume Version Control:
- Extend beyond PDF to support DOCX, TXT, and LinkedIn profile URL inputsMulti-format Support:
- Replace hardcoded admin credentials with Google/GitHub OAuth2 loginOAuth2 Authentication:
- Send analysis results to users' email via SMTP integrationAutomated Email Reports:
- Port the Streamlit app to a React Native mobile applicationMobile App:

XII. CONCLUSION

The AI Resume Analyzer demonstrates a complete, production-ready implementation of NLP-based resume intelligence built with Python and Streamlit. The system successfully integrates resume parsing, career prediction, skill gap analysis, ATS scoring, LinkedIn optimization, interview preparation, and administrative analytics into a cohesive, user-friendly platform.

The project has proven its real-world utility with over 37,000 tracked visitor interactions and active multi-user feedback collection. The modular architecture, clean code separation across pages, and MySQL-

backed persistence make it a strong portfolio piece for ML/AI engineering roles, particularly demonstrating practical NLP application development, full-stack Python development with Streamlit, database design and integration, and data analytics/visualization with Plotly.

The application aligns strongly with industry needs in the EdTech and HR-Tech spaces, and represents a viable foundation for a commercial resume optimization product with the enhancements outlined in Section 11.

REFERENCES

[1] S. Streamlit Inc., *Streamlit Documentation*. [Online]. Available: https://docs.streamlit.io. [Accessed: Jun. 18, 2026].

[2] E. A. Wiedemann et al., pdfminer.six Documentation. [Online]. Available: https://pdfminersix.readthedocs.io. [Accessed: Jun. 18, 2026].

[3] PyMySQL Developers, PyMySQL Documentation. [Online]. Available: https://pymysql.readthedocs.io. [Accessed: Jun. 18, 2026].

[4] Plotly Technologies Inc., Plotly Express Documentation. [Online]. Available: https://plotly.com/python/plotly-express/. [Accessed: Jun. 18, 2026].

[5] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python. Sebastopol, CA, USA: O'Reilly Media, 2009.

[6] A. Rajaraman and J. D. Ullman, Mining of Massive Datasets. Cambridge, U.K.: Cambridge University Press, 2011.

[7] ReportLab Inc., ReportLab PDF Generation Library Documentation. [Online]. Available: https://www.reportlab.com/docs/. [Accessed: Jun. 18, 2026].

[8] Oracle Corporation, MySQL 8.0 Reference Manual. [Online]. Available: https://dev.mysql.com/doc/. [Accessed: Jun. 18, 2026.].