AI-Powered Career Growth and Productivity Platform

Major project report submitted in partial full fillment of the requirement
for the degree of
Bachelor of Technology
In

Computer Science and Engineering

By

Priyansh Lunawat (221030169)

Naman Mittal (221030359)

Udit Sharma (221030199)

Ashish Agarwal (221030420)

UNDER THE SUPERVISION OF

Mr. Aayush Sharma (Assistant Professor (Grade-I))

Department of Computer Science & Engineering and Information Technology

Jaypee University Of Information Technology, Waknaghat, 173234, Himachal Pradesh, (INDIA)

December 2025

SUPERVISOR'S CERTIFICATE

This is to certify that the major project report entitled **'AI-Powered Career Growth and Productivity Platform'**, submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering**, in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat, is a bonafide project work carried out under my supervision during the period from July 2025 to December 2025.

I have personally supervised the research work and confirm that it meets the standards required for submission. The project work has been conducted in accordance with ethical guidelines, and the matter embodied in the report has not been submitted elsewhere for the award of any other degree or diploma.

[illegible] 02/12/25

Supervisor Name & Sign: Mr. Aayush Sharma

Designation: Assistant Professor (Grade-I)

Department: Computer Science & Engineering

Date: 27/11/2025

Place: Waknaghat, Solan

SUPERVISOR’S CERTIFICATE

This is to certify that the major project report entitled **‘AI-Powered Career Growth and Productivity Platform’**, submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering**, in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat, is a bonafide project work carried out under my supervision during the period from July 2025 to December 2025.

I have personally supervised the research work and confirm that it meets the standards required for submission. The project work has been conducted in accordance with ethical guidelines, and the matter embodied in the report has not been submitted elsewhere for the award of any other degree or diploma.

Supervisor Name & Sign: Mr. Aayush Sharma

Date: 27/11/2025 **Designation:** Assistant Professor (Grade-I)

Place: Waknaghat, Solan **Department:** Computer Science & Engineering

CANDIDATE'S DECLARATION

We hereby declare that the work presented in this major project report entitled **'AI-Powered Career Growth and Productivity Platform'**, submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science & Engineering**, in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat, is an authentic record of our own work carried out during the period from July 2025 to December 2025 under the supervision of **Mr. Aayush Sharma**.

We further declare that the matter embodied in this report has not been submitted for the award of any other degree or diploma at any other university or institution.

[illegible]

Name & Sign: Priyansh Lunawat
Roll No.: 221030169
Date: 27/11/2025

[illegible]

Name & Sign: Naman Mittal
Roll No.: 221030359
Date: 27/11/2025

[illegible]

Name & Sign: Udit Sharma
Roll No.: 221030199
Date: 27/11/2025

[illegible]

Name & Sign: Ashish Agarwal
Roll No.: 221030420
Date: 27/11/2025

This is to certify that the above statement made by the candidates is true to the best of my knowledge.

Aayush 27/11/25

Supervisor Name & Sign: Mr. Aayush Sharma
Designation: Assistant Professor (Grade-I)
Department: Computer Science & Engineering

Date: 27/11/2025
Place: Waknaghat,Solan

CANDIDATE’S DECLARATION

We hereby declare that the work presented in this major project report entitled **‘AI-Powered Career Growth and Productivity Platform’**, submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering**, in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat, is an authentic record of our own work carried out during the period from July 2025 to December 2025 under the supervision of **Mr. Aayush Sharma**.

We further declare that the matter embodied in this report has not been submitted for the award of any other degree or diploma at any other university or institution.

Name & Sign: Priyansh Lunawat	**Name & Sign:** Naman Mittal
Roll No.: 221030169	**Roll No.:** 221030359
Date: 27/11/2025	**Date:** 27/11/2025
Name & Sign: Udit Sharma	**Name & Sign:** Ashish Agarwal
Roll No.: 221030199	**Roll No.:** 221030420
Date: 27/11/2025	**Date:** 27/11/2025

This is to certify that the above statement made by the candidates is true to the best of my knowledge.

	Supervisor Name & Sign: Mr. Aayush Sharma
Date: 27/11/2025	**Designation:** Assistant Professor (Grade-I)
Place: Waknaghat,Solan	**Department:** Computer Science & Engineering

ACKNOWLEDGEMENT

Firstly, We express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the project work successfully.

We are really grateful and wish our profound indebtedness to Supervisor **Mr. Aayush Sharma**, Department of CSE Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of "**Artificial Intelligence**" to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

We would also generously welcome each one of those individuals who have helped us straight forwardly or in a roundabout way in making this project a win. In this unique situation, we might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated our undertaking.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

Priyansh Lunawat (221030169)

Naman Mittal (221030359)

Udit Sharma (221030199)

Ashish Agarwal (22103420)

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The project **AI-Powered Career Growth and Productivity Platform** is a conceptualized future initiative that aims to enhance the lives of students, job seekers, and working professionals by career preparation. It smartly integrates two crucial stages of a career journey into a single platform. Phase one, users can create professional resumes and cover letters, simulate interviews, and also receive AI-generated predictions of potential future job titles based on their resume. This way, they not only understand the career opportunities but also the skills that they need to develop further.After that, the spotlight moves to technical growth with the system creating a tailor-made coding challenge derived from the skills mentioned in the resume. Individuals take the tests in a tightly controlled code editor and then the AI assesses their performance. The outcome is conveyed in the form of insights and charts that make the users' competencies and deficiencies visually clear.Therefore, by integrating career counseling with technical evaluation, this project serves as an all-in-one, AI-driven, future readiness and personal growth solution.

CHAPTER 1: INTRODUCTION

1.1 Introduction:

The shift to technology-oriented recruiting has served to change the requirements of a candidate. Contrary to previous recruitment where meetings were conducted in person, paper and pen evaluation and flexible evaluation methodologies, contemporary hiring is more organized. Companies typically get hundreds of applications for each role and to manage this influx they utilize automated tools that screen, sort and score candidates prior, to any recruiter reviewing their dossiers. Consequently the way information is presented. In résumés cover letters and answers during interviews. Now has a significantly greater impact, on deciding if a candidate advances to the subsequent phase.

Nevertheless many students and entry-level professionals find it challenging to fulfill these demands for reasons. One problem is the understanding of how applicant tracking systems (ATS) evaluate résumés. Numerous applicants unknowingly create files that appear attractive but do not align well with ATS criteria. Another challenge lies in one’s abilities clearly yet impactfully. Developing a résumé or cover letter that is both authentic proves difficult particularly without expert assistance. Likewise getting ready, for interviews demands rehearsal, self-assurance and an understanding of the questions employers tend to pose. Numerous students lack chances to engage in interview practice within a realistic setting.

To address these shortcomings this initiative presents a inclusive AI-powered platform designed to support users at all crucial phases of the employment application process. Than inundating users with an excess of tools or directions the platform provides a step-, by-step guided experience. Users have the ability to create résumés polish their narratives, craft cover letters customized for job positions and engage in practice interviews. All integrated into one unified interface. The concept of the system is not to substitute judgment but to provide accessible dependable assistance that students can utilize on their own.

Technically speaking the platform has been developed with modularity as a core principle. Various modules are responsible for functions like résumé formatting, content assessment, interview practice and job monitoring. These modules interact via defined APIs facilitating future scalability of the system. The integration of AI technology. NLP, to analyze and produce text. Increases the capability of the platform to react and adapt to various user needs.

This work is meant to minimize stress and confusion experienced by several job seekers. The platform will provide tips that are straightforward, personal, and founded on the current recruitment standards to enable users show themselves better. Finally, the project aids in a better adaptation to the academic life to the professional setting as it provides the individuals with the means to compete in the world of a fast-moving professional environment with confidence.

1.2 Problem Statement

Students will often struggle with putting together career documentation because of not being advised or getting the right knowledge of what the field requires. Many students are confused, as to what a recruiter wants, how to present the best of themselves, or how to compose a professional document. Most of the resources include merely basic templates, which in most cases are not in line with the unique abilities, work experience and career expectations of specific students.

The other problem is that no single platform unites all the resources that are important in career development. The students should be enabled to use different sites to draft resumes, draft cover letters to prepare to attend the job interview and get job-related tips. Such a disjointed approach does not waste time, but generates discrepancies, in the products they make.

Due to all these constraints, it is apparent that a single, user friendly web system is necessary that will assist students in their career preparation process. An AI-based platform that tailors the resume, offers them the right job opportunities, and writes professional cover letters and helps them practice interviews can substantially decrease the amount of work and enhance the quality. Such system can help to guide students in a more organized manner and simplify the process on the whole making it more effective.

1.3 Objectives:

The main objectives of the project are:

- To develop a user-friendly web application that helps students create professional career documents with ease.
- To generate personalized resumes and cover letters using AI-based suggestions tailored to a student's skills and background.
- To provide an AI-powered job-role recommendation feature that guides students toward suitable career options.
- To build a Job Application Tracker that allows students to manage and monitor all their internship and job applications in one place.
- To offer all these features within a single integrated platform, making the placement preparation process more organized, efficient, and accessible.
- To ensure that the system is easy to use, even for students with little technical experience

1.4 Significance and Motivation of the Project Work:

Students often spend many hours creating career documents using pen and paper. Lack of support the task becomes a slow, stressful and sometimes demoralizing experience. A lot of students do not have access to resume advisors, mentors or industry specialists that could help them polish their documents. As a result very talented students have a hard time demonstrating their achievements well.

The inspiration developed for this project was witnessing these issues faced by peers and juniors, throughout internship season. The lack of tailored instruments caused students to rely on templates which look similar and fail to convey distinctive strengths.

Through AI integration the platform not only helps in creating content but also helps students to explore career paths that they may have not been able to explore before. This is added value, given that educational institutions with hundreds of students seeking placements every year. Additionally the system allows for regularity, structure in formatting and accuracy ensuring that students will present documents that reflect them well.

This project helps in bridging the gap between the preparation of students and the expected in industry. It brings together the technology, usability and academic needs in a way that supports students and increases their chances of success.

1.5 Organization of Project Report:

The project report is organised in six chapters, each of them dealing with a distinct stage of the system's development:

- **Chapter 1: Introduction:**

  Presents background information, explains the purpose of the project, the problem and the objectives and motivation

- **Chapter 2: Literature Survey:**

  Reviews existing research, tools, and recent developments connected to resume generation, cover letter automation, AI-based recommendation systems and job applications platforms.

- **Chapter 3: System Development:**

  Covers system requirements, architecture, backend and frontend design, algorithms, data flow and details on implementation.

- **Chapter 4: Testing:**

  Describes the testing strategy, test cases, testing methods used and results of the system evaluation.

- **Chapter 5: Results and Evaluation:**

  Discusses the performance of the system, what was observed, and compares results with extant solutions.

- **Chapter 6: Conclusion and Future Scope:**

  Concludes the project, highlights what has been done and presents the roadmap for future improvements and extension of features.

CHAPTER 2: LITERATURE

This chapter reviews existing research, tools, and methodologies related to AI-driven career guidance, resume optimization, interview preparation, and coding assessments. By analyzing prior work, we identify the gaps that our proposed system aims to address.

2.1 Overview of Relevant Literature:

Over the past few years, a lot of work has been done in the areas of resume parsing, AI-based content generation, and online career-support tools. Most of the recent studies and technical articles highlight how students and job seekers are increasingly relying on automated systems to prepare resumes, write cover letters, and identify suitable career roles. This growth has mainly been driven by improvements in NLP models and the availability of large, pre-trained language models.

- **Resume parsing and extraction:** Earlier systems mostly depended on fixed templates or simple rules to read resumes. These worked only when the resume format was predictable. Recent research (especially during the last 4–5 years) shows a major shift towards using deep-learning models like BERT to understand resume text more accurately. These models perform better when resumes have different layouts, fonts, or PDF structures. Many recent publications also combine OCR with NLP to handle scanned resumes.
- **Resume and cover-letter generation:** The traditional method was mostly template-based: users filled in a form and the system generated a standard resume or cover letter. There is recent work discussing the use of transformer based models to generate more natural and personalised content. These systems have the ability to analyze the user's skills and experiences and rewriting it in a more professional manner. There are also several studies that layout that AI-generated content result in not only an increase in readability but also minimizing errors and saving time for students.
- **Career Role Recommendation:** Modern job recommendation research is focused on skills-job description similarity using embedding-based similarity. Instead of the keyword matching, recent work is done using vector representations of skills, responsibilities, and experience. Many studies combine this with occupational databases or knowledge graphs in order to better the accuracy of suggested roles.

- **Interview preparation tools:** Although this field is still growing, recent work suggests the use of conversational agents and feedback systems to help students practice interview questions. These tools usually apply NLP to analyse answers and offer some basic suggestions like clarity, structure and relevance. Research papers also mention the difficulties in giving human-like feedback which is still an open problem.
- **Industry practices and cloud-based systems:** White papers and technical blogs from large cloud providers explain how real platforms are built in the real world: using microservices, NoSQL databases, as well as separate frontend - backend layers; These resources also focus on the security common practices, user authentication and scalability which are followed in modern web systems.
- **Identified gap:** The literature indicates that there is a gap between student needs and the availability of tools that will help students learn. use different platforms for different things - one for building a resume, one for career suggestions of and another for practising interview. There is hardly any unified system that brings all these features in a simple and student-friendly manner. This disparity in current solutions supports the concept of our project.

2.2 Related Work:

Table 2.1 : Key Research Works

S.No.	Author & Paper Title	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Key Findings/ Results
1	Nguyen, T. T. H. et al. – SimInterview: Transforming Business Education through LLM-Based Simulated Multilingual Interview Training System	arXiv (2025)	LLMs, Whisper STT, GPT-SoVITS, Ditto avatar, ChromaDB, RAG	Multilingual simulated interviews; high user satisfaction; better interview readiness
2	Koshti, H. et al. – *AI*-Powered Interview Preparation System: Integrating Resume Analysis, HR Simulation, and Technical Skill Assessment	JERR (2025)	CNN for emotion detection, NLP for resume analysis, Speech Recognition, ATS integration	Improved candidate preparation efficiency; objective evaluation; real-time feedback

3	Gayathri Devi M. et al. – AI Friendly Resume	IJSREM (2024)	AI/NLP for keyword extraction, semantic matching, formatting	Increased resume shortlisting rates compared to traditional formats
4	AI-Based Mock Interview Platform	IRJMETS (2025)	Smart resume parsing, emotion detection, dynamic question generation	Personalized interview practice; real-time feedback;
5	Sharma & Jain – Code Evaluation and Suggestion System	IERJ (2025)	Automated code evaluation, suggestion modules	Evaluates submitted code and provides improvement suggestions
6	Resume Generator Using AI	IRJMETS (2025)	Flask, NLP, ML, template systems	AI-assisted resume structuring, optimization, and output
7	Automated Code Assessment for Education: Review and Perspectives	MDPI (2022)	Static analysis, test-based, ML-based feedback	Strengths, limitations, and trends in automated code assessment systems
8	Speeding Up Automated Assessment of Programming Exercises	ACM (2022)	Caching, ML prediction, static analysis	Reduced latency in grading; improved scalability
9	Sanyal, K. et al. – Intelligent Resume Parsing and Job Recommendation via Web-Based CV Analysis System	IJRASET (2025)	NLP (pdf parsing, regex), Flask, job search API	Extracts structured info from resumes; real-time job recommendations
10	Korrapati, L. N. V. Babu et al. – A Machine Learning Approach for Automation of Resume Recommendation System	IJRASET (2022)	Text mining, resume classification	Classifies resumes, automates recommendation; speeds up candidate screening

11	Varshith Reddy, K. S. et al. – Resume Analyzer and Job Recommendation System	IRE Journals (2025)	OCR, NLP, TF-IDF, Cosine Similarity, KNN	Structured resume data; ranks candidates; suggests learning paths
12	Jiang, J. et al. – Learning Effective Representations for Person-Job Fit by Feature Fusion	arXiv (2020)	Deep learning, feature fusion, LSTM	Improved candidate-job matching accuracy using explicit and implicit features
13	Rahman, M. et al. – Artificial Intelligence in Career Counseling: A Test Case with ResumAI	arXiv (2023)	AI chatbot, resume feedback	ResumAI effective in improving resumes; supports AI in career counseling
14	Apaza, H. et al. – Job Recommendation Based on Curriculum Vitae Using Text Mining	Springer FICC (2021)	Text mining, similarity measures	Matches CVs with job descriptions; reduces manual screening workload
15	Messer, M. et al. – Automated Grading and Feedback Tools for Programming Education: A Systematic Review	arXiv (2023)	Systematic review of automated tools	Categorizes grading tools; identifies strengths and limitations
16	Li, C. et al. – Competence-Level Prediction and Resume & Job Description Matching Using Transformer Models	arXiv (2020)	Transformer models, section encoding	Accurate competence prediction and job matching using context-aware models
17	Patil, R. et al. – Automated Assessment of Multimodal Answer Sheets in the STEM Domain	Papers With Code (2024)	LLMs, YoloV5, OCR	Evaluates text + diagrams; improves assessment of complex answers
18	Khan, M. S. et al. – Automated Assessment for C/C++ Programming in ODL Environment	Papers With Code (2022)	Auto grading, feedback for C/C++	Scalable grading for large cohorts; effective feedback mechanisms

19	Sharma, P. & Singh – Personalized Job Search with AI: Skill-Based Matching	IJERT (2023)	NLP, deep learning, transformer models	Transformer-based matching outperforms rule-based; improves recruiter efficiency
20	Shivhare, K. et al. – ResumeCraft: ML-Powered Web Platform for Resume	IJRASET (2024)	ML models, web app (HTML/CSS/JS)	ATS-friendly resumes; improved user satisfaction

2.3 : Key Gaps in the Literature:

- Most tools available today concentrate on only one task - whether paperwork, cover letter writing or job role suggestions - meaning students find themselves using multiple platforms rather than one, unified system.
- Many resume builders are still relying on very generic templates, which do not really show a student's unique skills or what different industries are looking for.
- Very limited research to pull together the three elements of resume parsing, AI-based content generation, and job role prediction into a single connected workflow.
- Some of the existing systems are having problems where they could not handle resumes with a different layout, scanned PDF or even creative designs hence they are less reliable.
- Cover-letter solutions mentioned in research are mostly template based and do not generate personalized content for the student's background.
- Job role recommendation from AI still is at a very basic level and there are very few studies that can link user skills with actual industry job frameworks.
- Interview-practice tools typically have a fixed set of questions and have little feedback, so they are not as useful for actually preparing.
- Many academic papers put a lot of emphasis on the performance of an algorithm, and the user experience and ease of use are often neglected.
- There is not much work being done with the intention of making these systems affordable, accessible, and especially student-friendly.
- There are few open source projects that make an effort to provide a complete career support for educational institutions to adopt
- Research prototypes are rarely concerned with issues such as cloud hosting, security, building scalable backends that can support real users.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 Requirements and Analysis:

The development of the platform was started with a clear understanding of what users are actually expecting from when they prepare for job applications. Most students or early-career professionals have difficulties with organizing their documents, keeping track of different job portals, keeping their responses of different quality across resumes, cover letters, and interviews. In order to make sure that the system addressed these issues in a meaningful way, the requirements were broken down into functional requirements, non-functional requirements, and user experience considerations.

Functional Requirements :

- **User Registration and Authentication** : The platform must let users create secure accounts, log in, and securely store their information. Since users will enter personal and academic details, it is crucial to maintain data privacy.
- **Resume Builder**: Users should have the capability to input educational qualifications, skills, job history, projects, and achievements in a well-structured manner. The system is obliged to verify the inputs and locate any missing or ambiguous parts.
- **Cover-Letter Help**: The medium should generate or upgrade cover letters based on job roles. It ought to be that the user edits the text and gets suggestions for improving the tone, giving more clarity, and being more relevant.
- **Interview Practice Module:** Users should be enabled to view the roles they could do and be interested in, given their skills, and keep track of the applications with statuses such as Applied, Shortlisted, or Rejected.
- **Job Recommendation and Tracking:** Users should be enabled to view the roles they could do and be interested in, given their skills, and keep track of the applications with statuses such as Applied, Shortlisted, or Rejected.

Non-Functional Requirements:

- **Performance:** The service should be such that it answers promptly even when one is accessing different modules one after another. To avoid user impatience, all API calls should be completed within a reasonable time.
- **Scalability**: The system should handle growing user data—résumés, interview attempts, and saved jobs—without affecting overall performance.
- **Security**: Passwords must be encrypted, and sensitive personal information must be stored securely. Unauthorized users should not be able to access private profiles.
- **Usability**: The interface should be simple enough for first-time users. Navigation has to feel natural, with clear labels and an uncluttered design.

3.2 Project Design and Architecture:

The project follows a simple and clear plan. Such a structure makes the system a piece of cake to change or add to later. The architecture consists of two separate sections: the frontend and the backend. The frontend is handling all user interactions such as forms, dashboards, and resume previews. The backend takes care of saving data, managing user profiles, handling job application records, and generating responses.

A modular design structure was used so each feature—such as authentication, job tracking, and resume generation—works as its own component. This approach makes the system cleaner and helps in updating any feature without disturbing the others. The database schema was planned in a way that user information, job applications, and resume data remain properly linked, making the workflow smooth and reliable.**(**see *Figure 3.1* for the System Architecture of Career Companion**).**

3.2.1 User Layer:

- **Inputs**: User enters personal details, academic qualifications, skills, and career goals.
- **Actions**:
    - Upload Resume / Profile Data
    - Attempt Coding Exam
    - Request Cover Letter

- Participate in Interview Simulation
- Job application Tracker

3.2.2 AI Processing Layer (Modules):

1. **Resume Analyzer (NLP + ML):**

   - Extracts keywords, skills, achievements.
   - Matches content with job descriptions (ATS optimization).
   - Suggests missing keywords and formatting fixes.

2. **Job Role Predictor (Recommendation Engine):**

   - Uses resume + skill data.
   - Suggests suitable future career roles.

3. **Cover Letter Generator (LLM):**

   - Uses resume + job description.
   - Generates tailored, ATS-friendly cover letter

4. Job Application Tracker :

   - Company name + role + application status + Application link
   - Generates tracker of different application of different portals in one place.

3.2.3 Output Layer:

- Generated Resume (ATS-Optimized)
- Cover Letter
- Suggested Career Roles

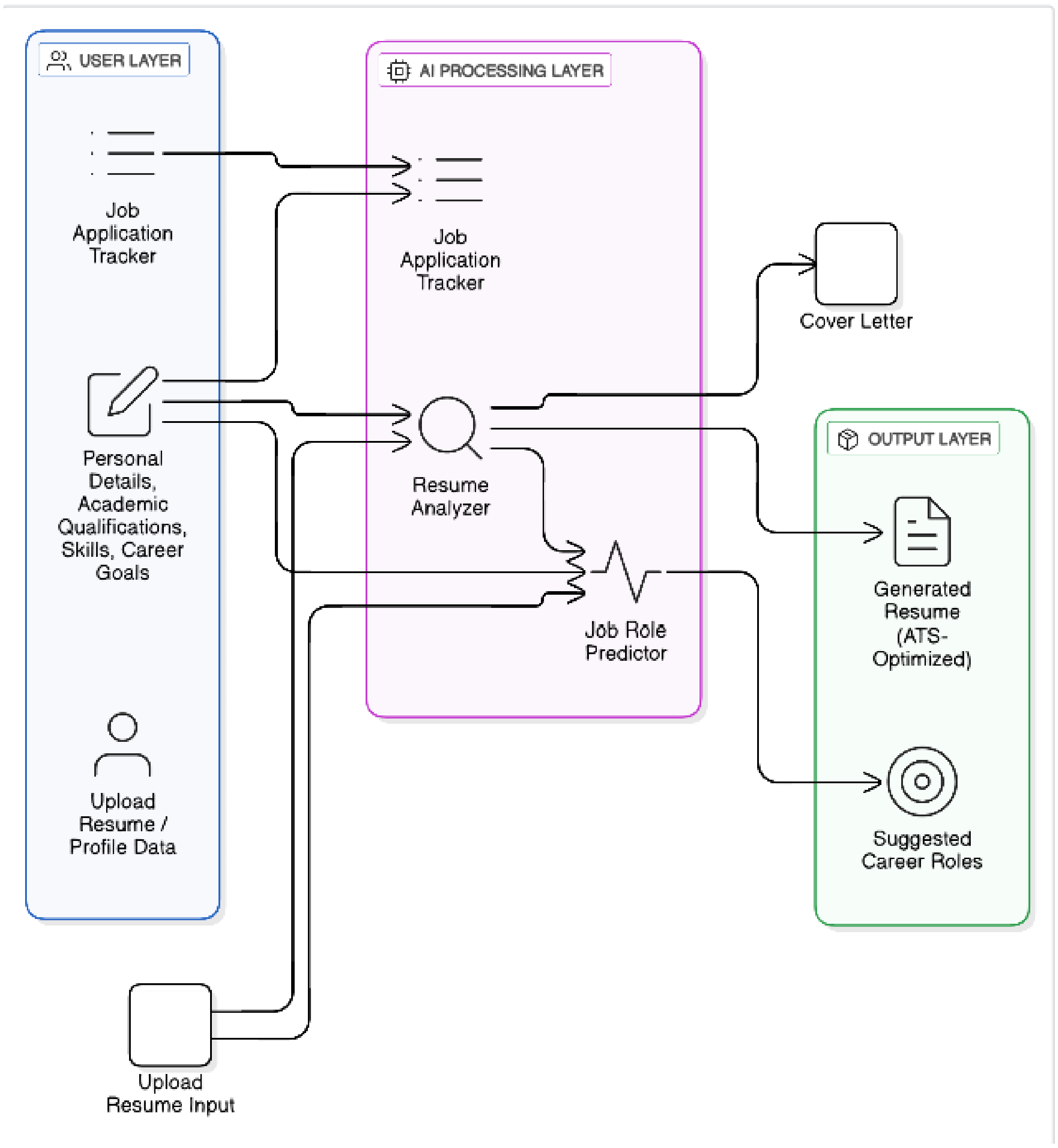


Figure 3.1: System Architecture of Career Companion

The project follows a clean, three-layer structure:. The frontend is responsible for the screens students interact with, while the backend handles authentication, storage, and API logic. MongoDB stores user accounts, resume information, and applications. Independently constructing every significant function as a module allows for seamless enhancements in the future. We arranged the API endpoints concentrically. For example, we have endpoints for auth, resume, cover letter, and job tracker, etc. We opted for a straightforward design for the architecture as the agenda for this semester is straightforward — building a functionally operational skeleton. Higher-level features such as predictive analytics and advanced AI interfacing will be incorporated in future iterations.

3.3 Tools and Technologies:

The proposed system was created using.

- **Frontend:** React.js / HTML / CSS
- **Backend:** Node.js or Django (depending on implementation)
- **Database:** MongoDB / MySQL
- **Resume Generation:** Template-driven PDF generation
- **API Integration:** RESTful API architecture
- **Authentication:** JWT / Session-based login
- **UI/UX Design:** Clean, modern, responsive interface

3.4 Data Preparation:

I made sure that the parameters that are important to me were accounted for, including text, skills, education, audio files of spoken responses and the written answers in a pre-formatted cover letter, when building my system. Coming up against a huge array of input possibilities, I constructed the system with a logical and watertight framework to prevent any confusion and to give the processing a silky-smooth flow.

3.4.1 Collecting User Input:

Users have their own preferred style, both with their description and in the answers given to the system. While some may enter a long text, other may answer in a single sentence, or a couple of phrases. This variation is tackled by normalizing all the inputs.

- Trimming extra spaces
- Correcting inconsistent capitalization
- Organizing the content into structured fields

3.4.2 Standardizing Resume Content

when users put together their resumes, invariably, there are areas and sections in the resume containing description gaps, and therefore the particular system must address the gaps by identifying:

- focusing on descriptions that are overly broad and words that are used in ways that are redundant.

- Highlight missing fields like dates, roles, or achievements
- Separate skills into clean categories

This ensures the résumé data can be used for later modules like job recommendations.

3.4.3 Preparing Cover-Letter Information

Cover letters require more natural writing, so the system:

- Breaks user text into paragraphs
- Checks for overly generic statements
- Identifies sentences that may need more context

The text is then restructured so the user can edit it smoothly.

3.4.4 Preparing Interview Responses

Interview answers are stored along with metadata such as:

- Question category
- Timestamp of the attempt
- Response length

This helps the system provide more meaningful feedback when the user returns to practice

3.4.5 Data Storage and Retrieval

All cleaned and structured data is stored in MongoDB. The schema flexibility allows users to update their information anytime without facing rigid form restrictions.

3.5 Implementation:

3.5.1 Workflow of Key Features:

- **Resume Generator + Job Prediction:** User fills form→ AI generates resume → Predicts future job roles (*see Figure 3.2* Flowchart of Resume Generator with Job Prediction).
- **Cover Letter Generator:** User selects job → AI drafts personalized cover letter (*see Figure 3.3* Flowchart of Cover Letter Generator).
- **Job Application Tracker :** allows users to add and manage their job applications by entering details like company, role, and application status *(see*

Figure 3.4 Flowchart of Job Application Tracker). The system continuously updates progress and displays all applications on a unified dashboard withfilters.

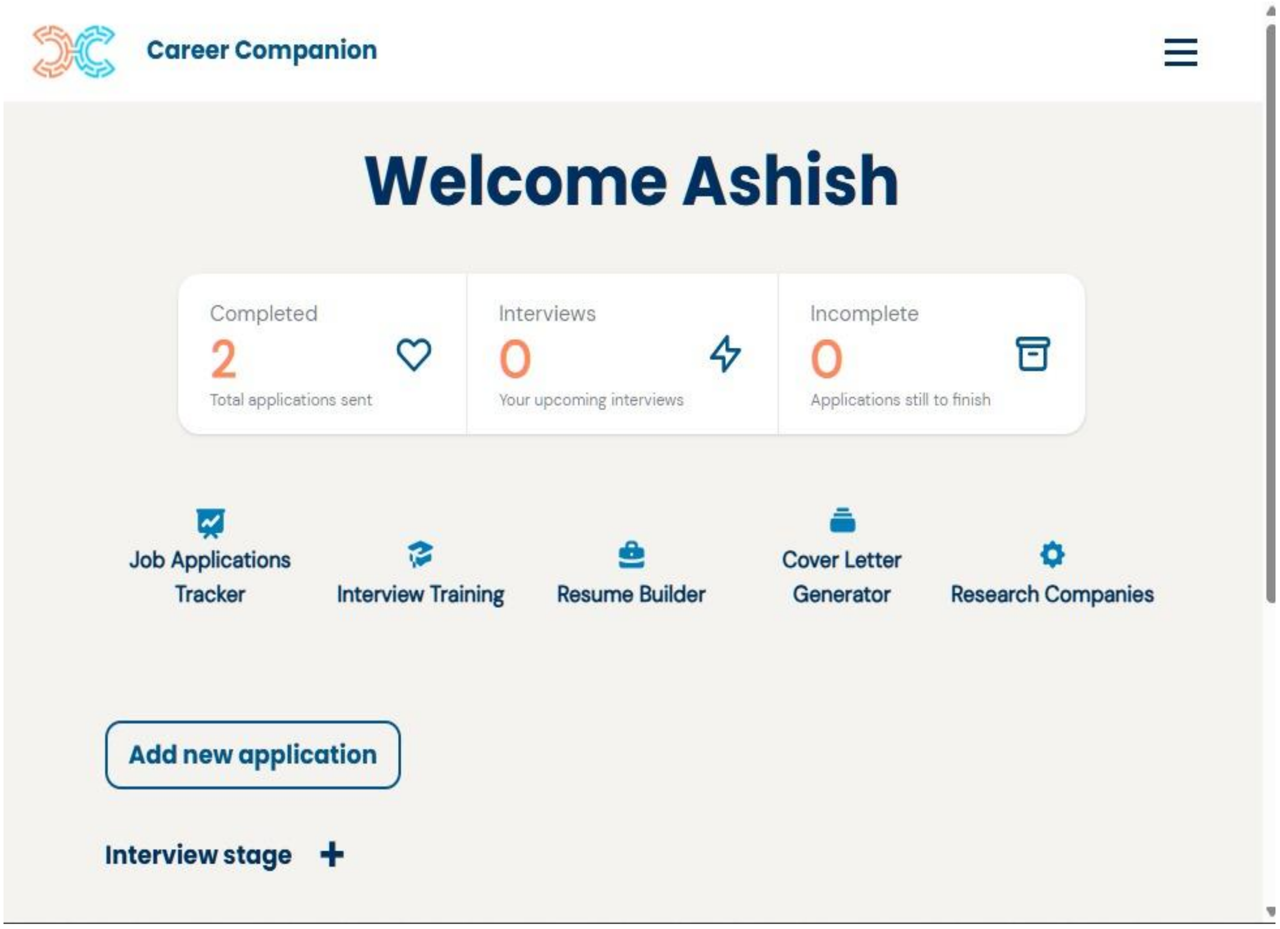


Figure 3.2: Website Frontend

This picture shows how actual website look like with all the features includes Job application tracker, resume builder ,research companies and so on.

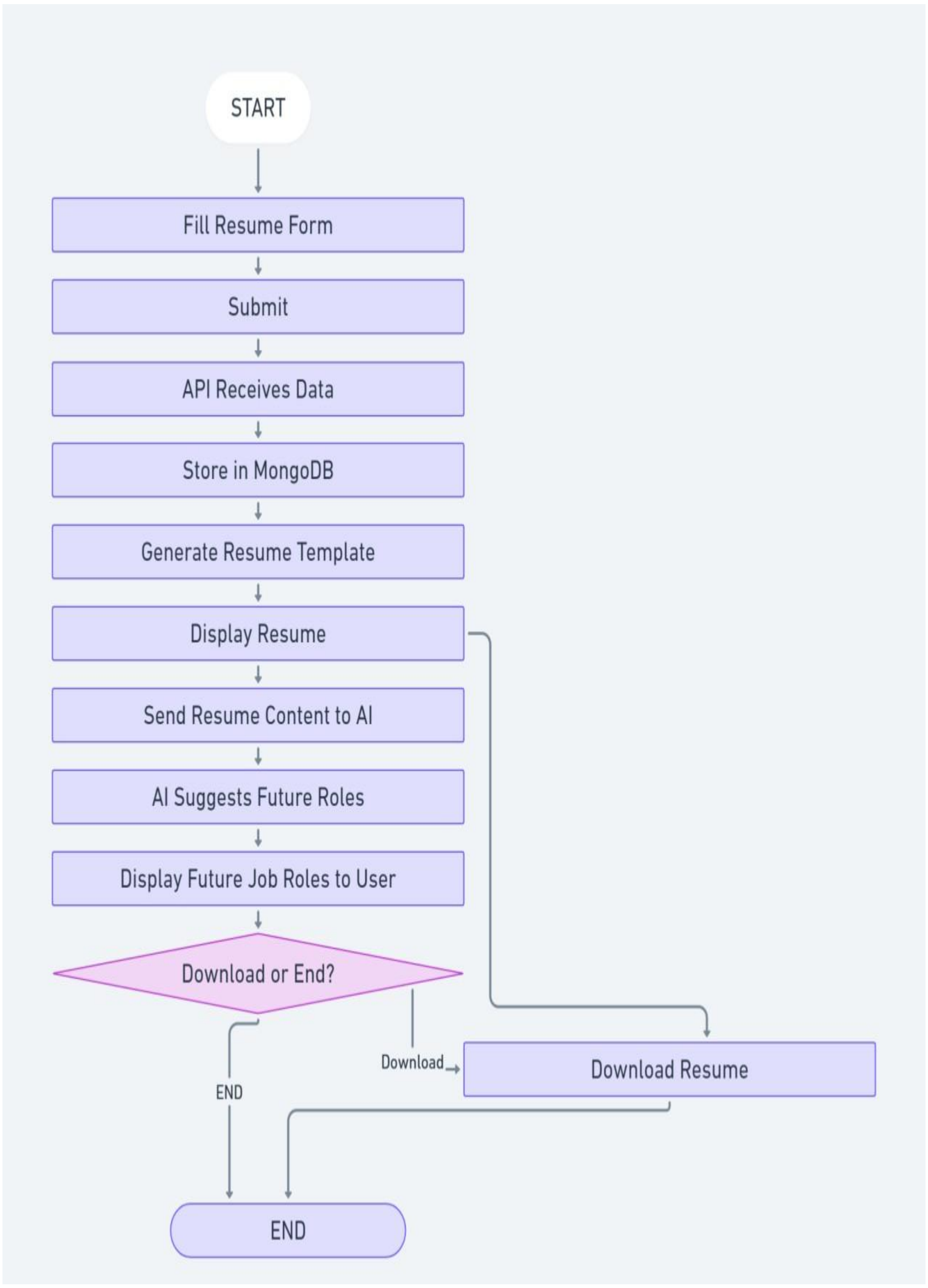


Figure 3.3: Flowchart of Resume Generator with Job Prediction

The flowchart illustrates how the user inputs personal and career details, which are processed by the AI to generate an ATS-optimized resume. Simultaneously, the system analyzes skills and experience to predict suitable future job roles, providing both a professional document and career guidance in one step.

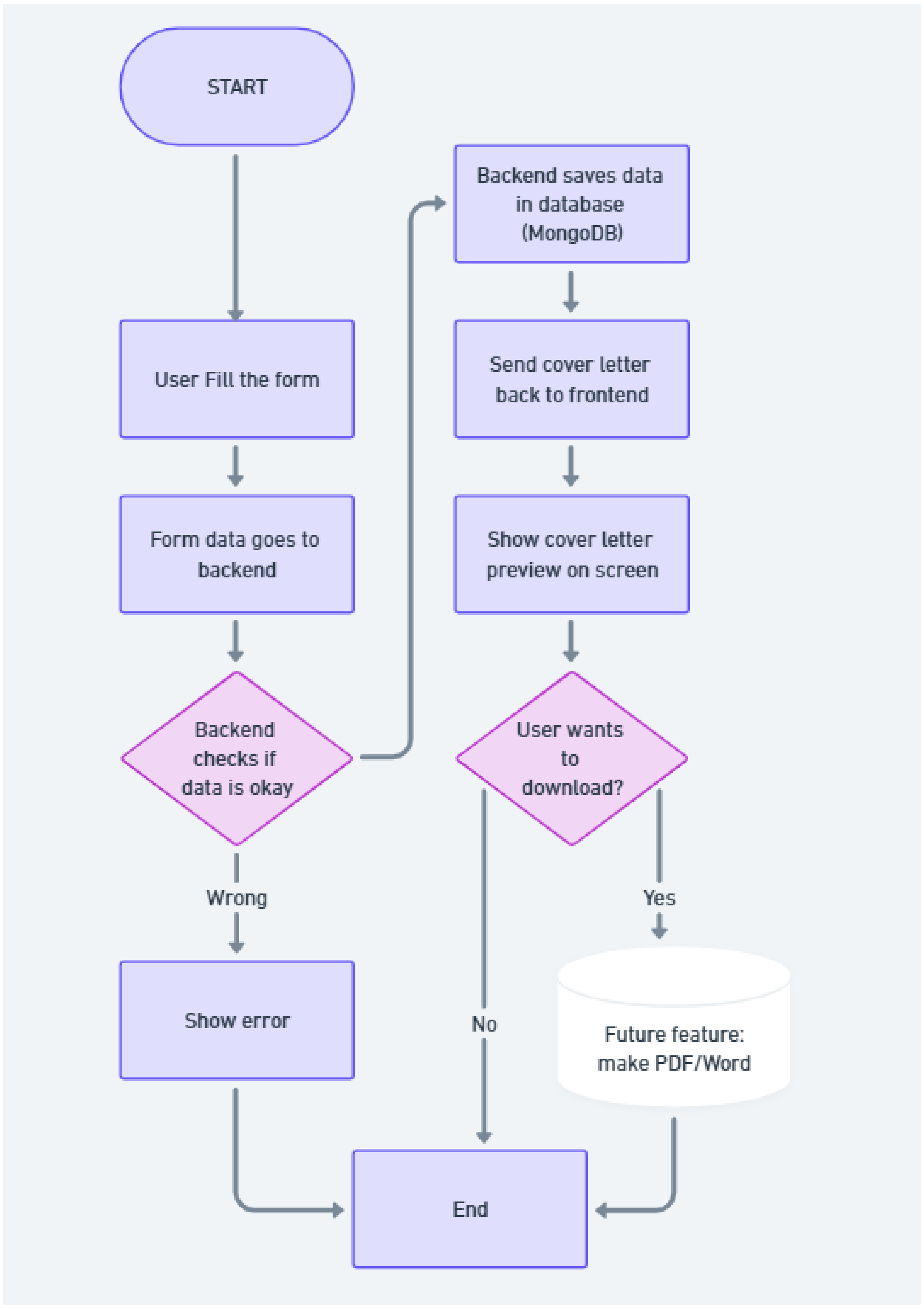

START
User Fill the form
Form data goes to backend
Backend checks if data is okay
Wrong
Show error
Backend saves data in database (MongoDB)
Send cover letter back to frontend
Show cover letter preview on screen
User wants to download?
Yes
No
Future feature: make PDF/Word
End

Figure 3.4: Flowchart of Cover Letter Generator

This flowchart illustrates the workflow for an online cover letter generator. A user submits a form, which the backend validates, saves to a MongoDB database, and then uses to generate a cover letter preview. The user can view this preview on the screen, with a future feature planned to allow downloading

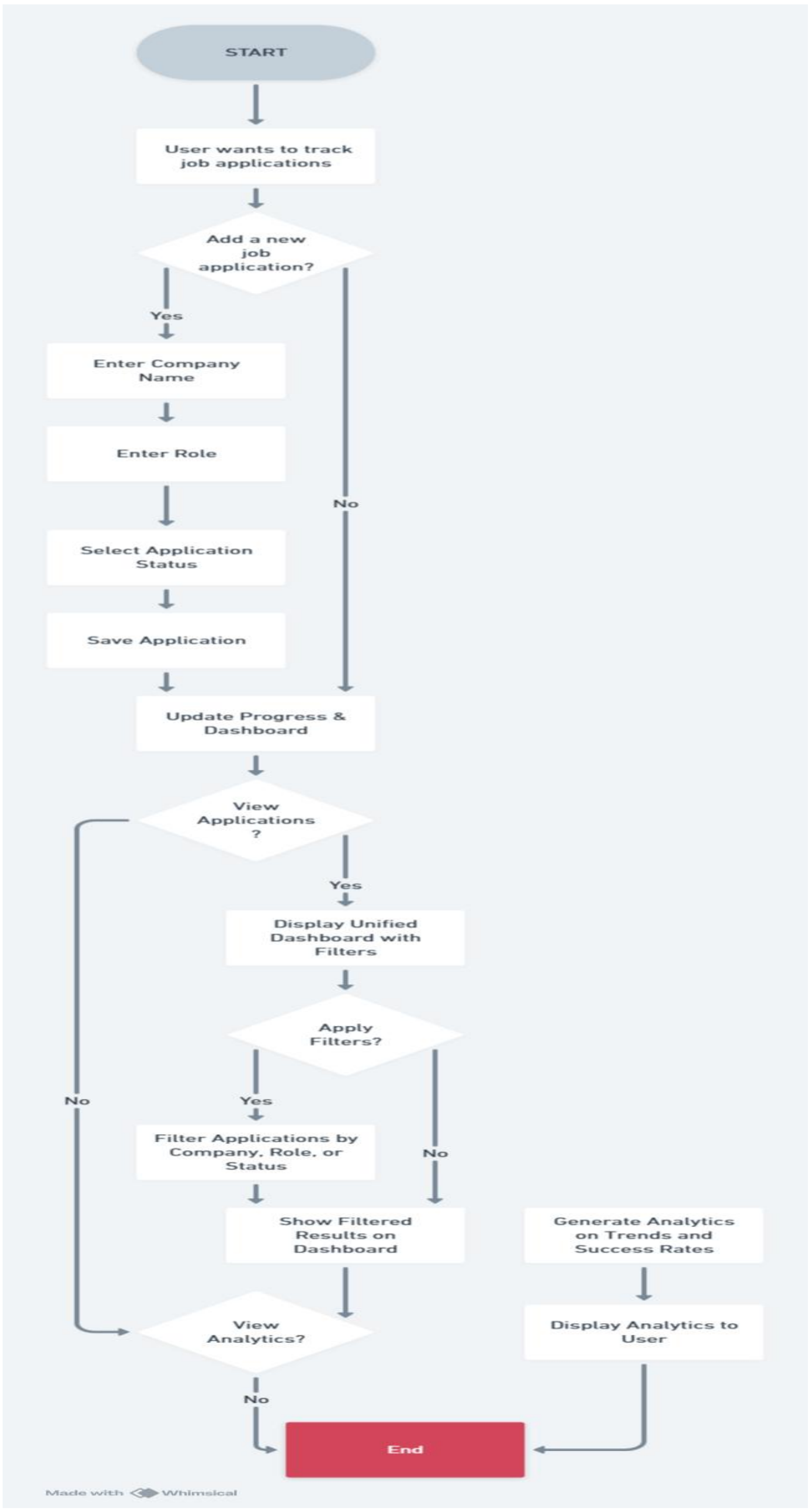
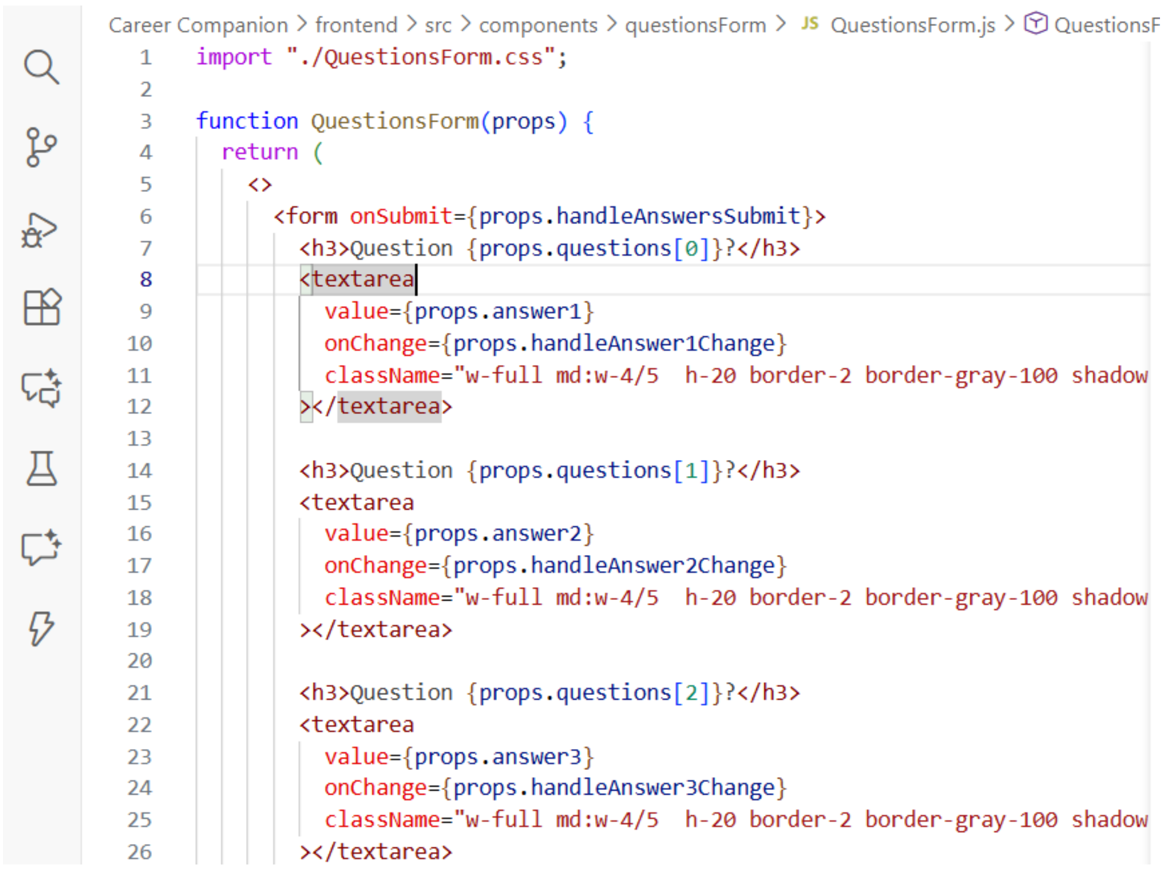

START
User wants to track job applications
Add a new job application?
Yes
No
Enter Company Name
Enter Role
Select Application Status
Save Application
Update Progress & Dashboard
View Applications ?
Yes
No
Display Unified Dashboard with Filters
Apply Filters?
Yes
No
Filter Applications by Company, Role, or Status
Show Filtered Results on Dashboard
Generate Analytics on Trends and Success Rates
Display Analytics to User
View Analytics?
No
End
Made with Whimsical

Figure 3.5: Flowchart of Job Application Tracker

The flowchart illustrates how users enter job application details, which the system validates and stores in the database. It then updates the dashboard, allowing users to monitor status changes

3.5.2 Code Snippets:



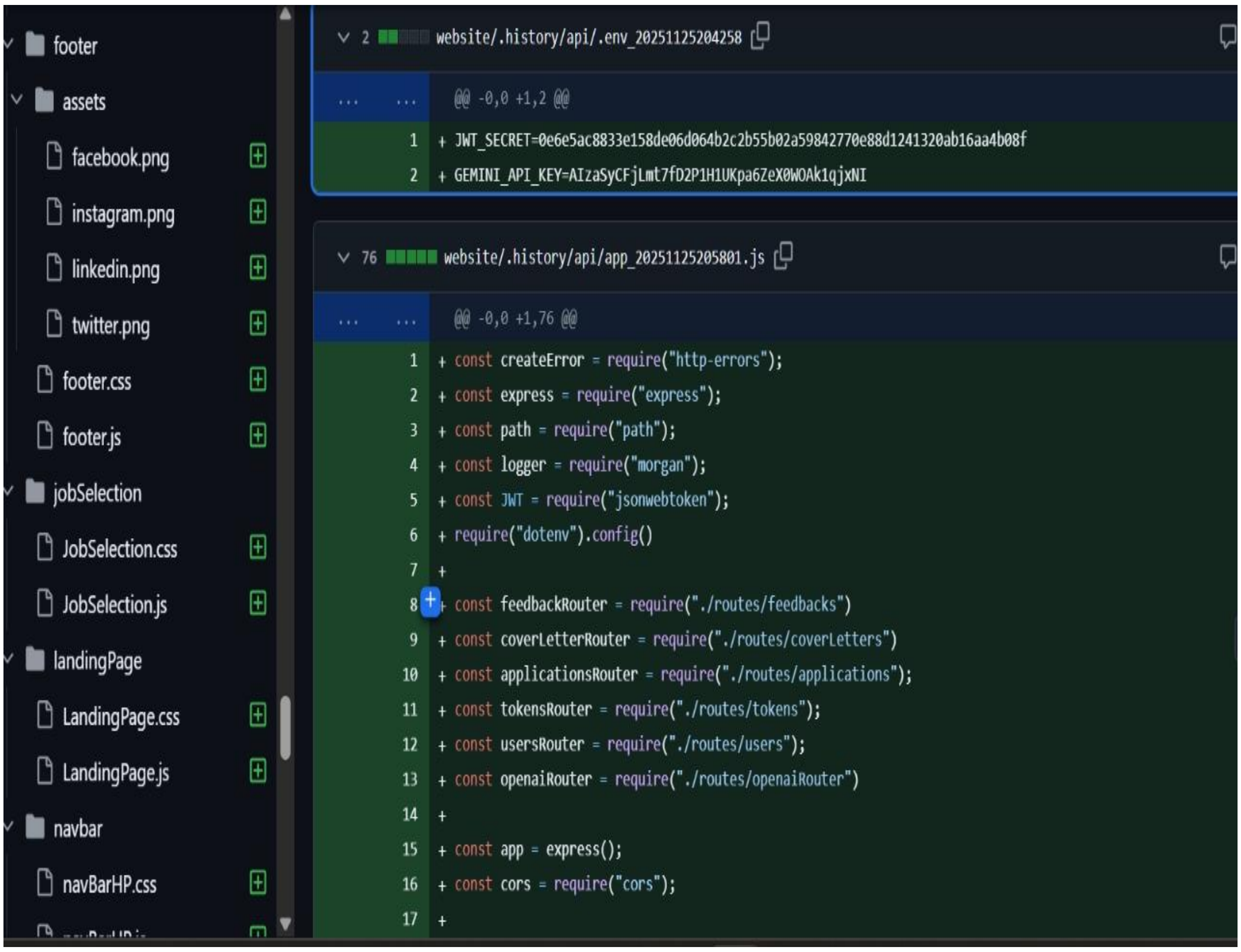
Career Companion > frontend > src > components > questionsForm > JS QuestionsForm.js > QuestionsF

```
import "./QuestionsForm.css";

function QuestionsForm(props) {
  return (
    <>
      <form onSubmit={props.handleAnswersSubmit}>
        <h3>Question {props.questions[0]}?</h3>
        <textarea
          value={props.answer1}
          onChange={props.handleAnswer1Change}
          className="w-full md:w-4/5  h-20 border-2 border-gray-100 shadow
        ></textarea>

        <h3>Question {props.questions[1]}?</h3>
        <textarea
          value={props.answer2}
          onChange={props.handleAnswer2Change}
          className="w-full md:w-4/5  h-20 border-2 border-gray-100 shadow
        ></textarea>

        <h3>Question {props.questions[2]}?</h3>
        <textarea
          value={props.answer3}
          onChange={props.handleAnswer3Change}
          className="w-full md:w-4/5  h-20 border-2 border-gray-100 shadow
        ></textarea>
```

Figure 3.6: Question Form Component in Frontend

Basically, the code for the React component is what you see if you figure out how the dynamic questions are being presented to the users one after another, each question being linked to a controlled textarea allowing real-time input and validation.

The form handling in this case becomes very easy and can be used again, which is in line with the modular design of the frontend, just like a logical consequence.



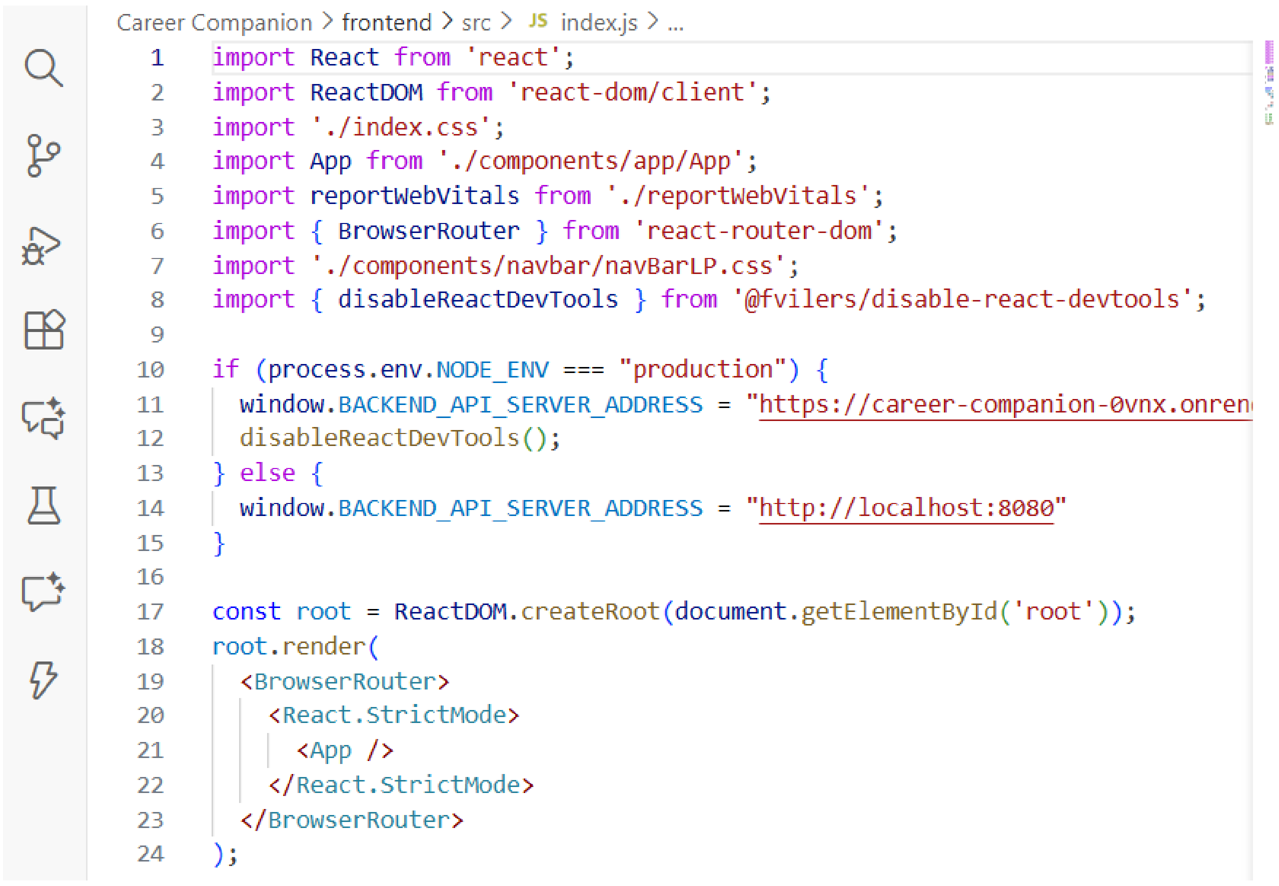
```
footer
  assets
    facebook.png
    instagram.png
    linkedin.png
    twitter.png
  footer.css
  footer.js
jobSelection
  JobSelection.css
  JobSelection.js
landingPage
  LandingPage.css
  LandingPage.js
navbar
  navBarHP.css
```

```
2 website/.history/api/.env_20251125204258
@@ -0,0 +1,2 @@
+ JWT_SECRET=0e6e5ac8833e158de06d064b2c2b55b02a59842770e88d1241320ab16aa4b08f
+ GEMINI_API_KEY=AIzaSyCFjLmt7fD2P1H1UKpa6ZeX0WOAk1qjxNI
```

```
76 website/.history/api/app_20251125205801.js
@@ -0,0 +1,76 @@
+ const createError = require("http-errors");
+ const express = require("express");
+ const path = require("path");
+ const logger = require("morgan");
+ const JWT = require("jsonwebtoken");
+ require("dotenv").config()
+
+ const feedbackRouter = require("./routes/feedbacks")
+ const coverLetterRouter = require("./routes/coverLetters")
+ const applicationsRouter = require("./routes/applications");
+ const tokensRouter = require("./routes/tokens");
+ const usersRouter = require("./routes/users");
+ const openaiRouter = require("./routes/openaiRouter")
+
+ const app = express();
+ const cors = require("cors");
+
```

Figure 3.7: Backend Environment Variable and Router Configuration

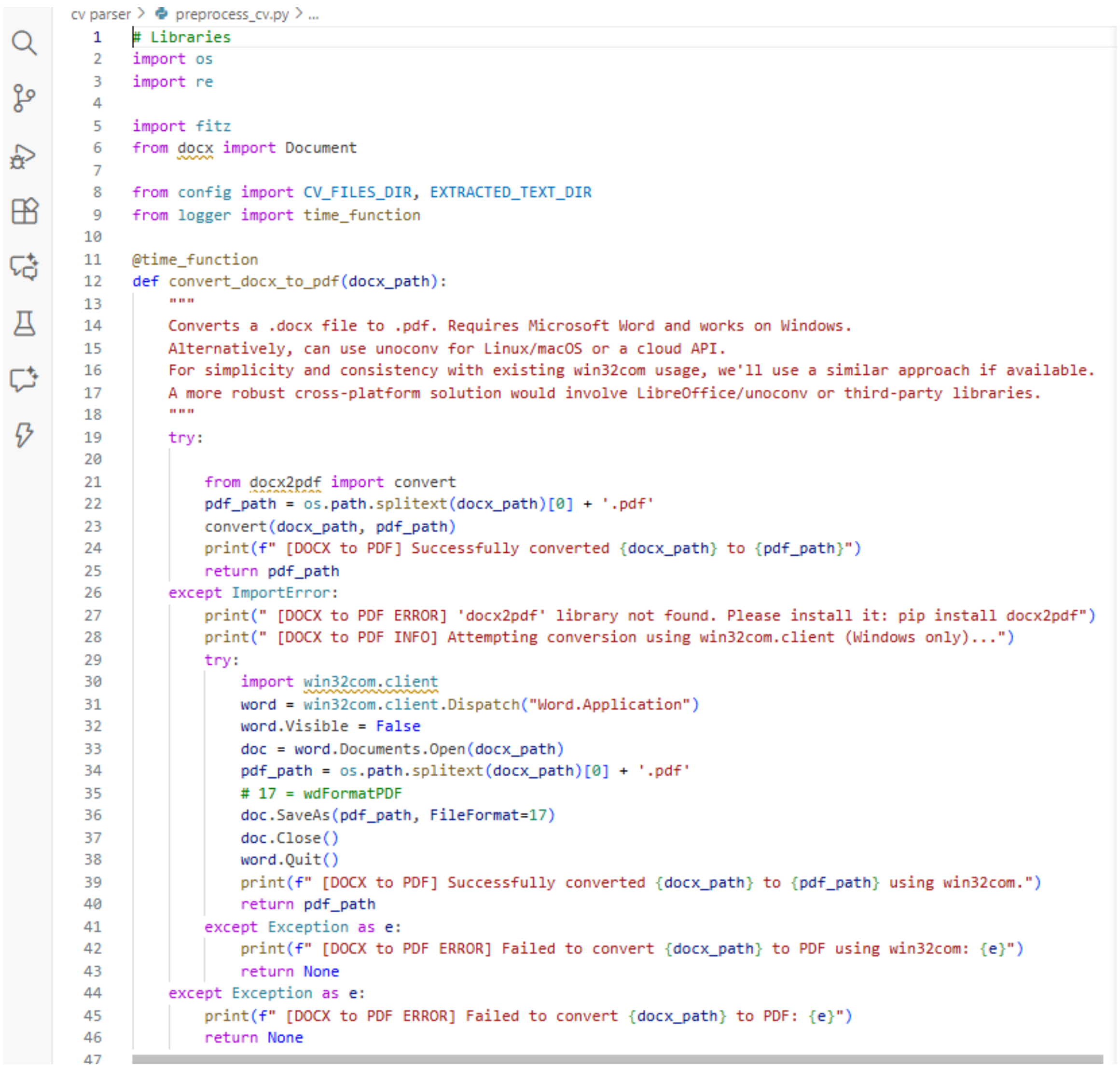
From the Node.js backend viewpoint, environment variables, API keys, and routes have been configured, and it is visible that feedback, applications, and cover letters are linked through Express routers. This modular design helps in feature separation and simplifies backend management..


Career Companion > frontend > src > JS index.js > ...

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './components/app/App';
import reportWebVitals from './reportWebVitals';
import { BrowserRouter } from 'react-router-dom';
import './components/navbar/navBarLP.css';
import { disableReactDevTools } from '@fvilers/disable-react-devtools';

if (process.env.NODE_ENV === "production") {
  window.BACKEND_API_SERVER_ADDRESS = "https://career-companion-0vnx.onren
  disableReactDevTools();
} else {
  window.BACKEND_API_SERVER_ADDRESS = "http://localhost:8080"
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <React.StrictMode>
      <App />
    </React.StrictMode>
  </BrowserRouter>
);
```

Figure 3.8: Frontend Entry File and API Configuration

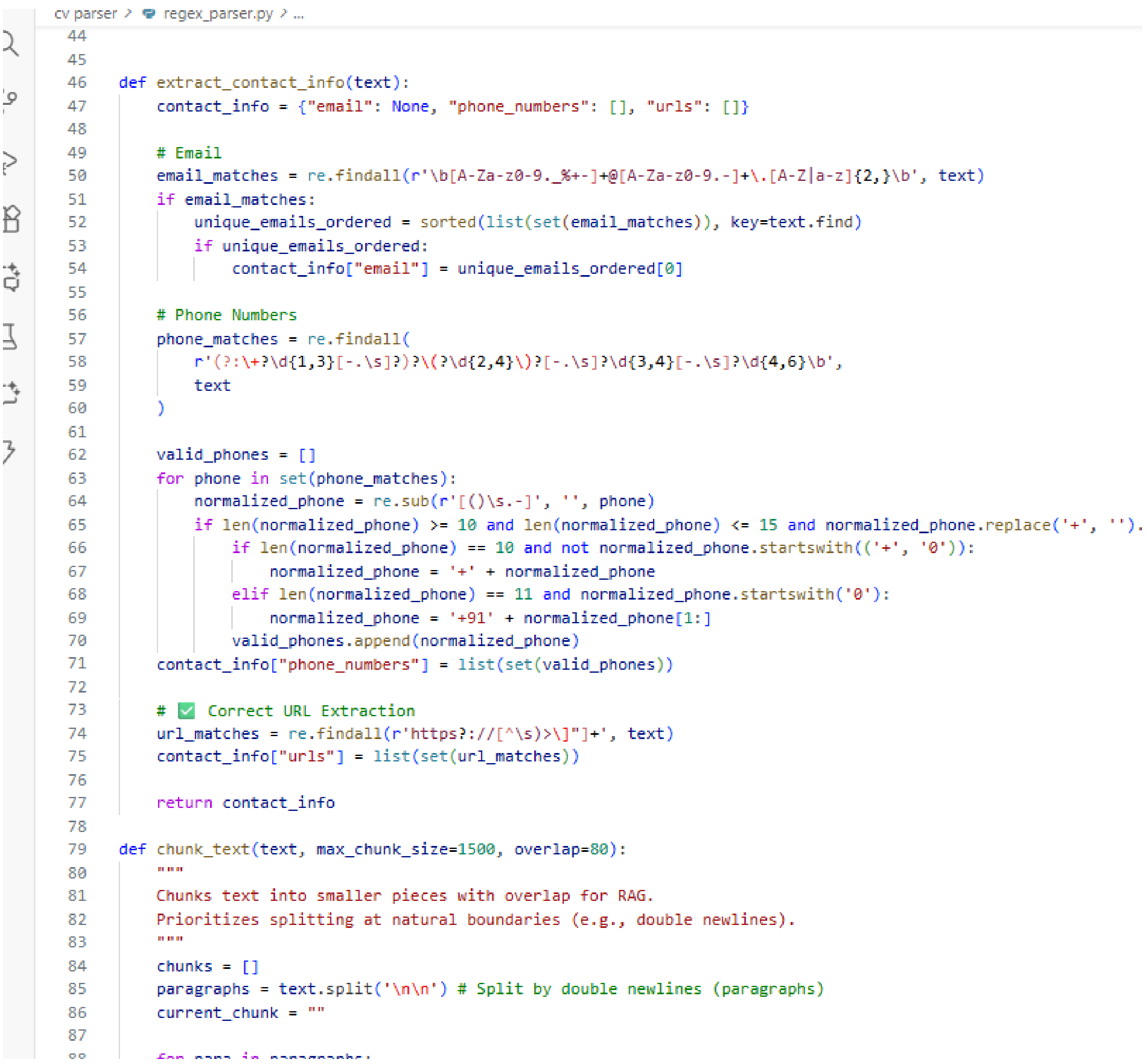
When it comes to selecting the backend API for a React app, the frontend handles this choice quite casually. Depending on whether the environment is development or production, the API selection is done automatically, and this fuss-free release is a definite sign that the app's routing, rendering, and API interaction are functioning properly.



```python
# Libraries
import os
import re

import fitz
from docx import Document

from config import CV_FILES_DIR, EXTRACTED_TEXT_DIR
from logger import time_function

@time_function
def convert_docx_to_pdf(docx_path):
    """
    Converts a .docx file to .pdf. Requires Microsoft Word and works on Windows.
    Alternatively, can use unoconv for Linux/macOS or a cloud API.
    For simplicity and consistency with existing win32com usage, we'll use a similar approach if available.
    A more robust cross-platform solution would involve LibreOffice/unoconv or third-party libraries.
    """
    try:

        from docx2pdf import convert
        pdf_path = os.path.splitext(docx_path)[0] + '.pdf'
        convert(docx_path, pdf_path)
        print(f" [DOCX to PDF] Successfully converted {docx_path} to {pdf_path}")
        return pdf_path
    except ImportError:
        print(" [DOCX to PDF ERROR] 'docx2pdf' library not found. Please install it: pip install docx2pdf")
        print(" [DOCX to PDF INFO] Attempting conversion using win32com.client (Windows only)...")
        try:
            import win32com.client
            word = win32com.client.Dispatch("Word.Application")
            word.Visible = False
            doc = word.Documents.Open(docx_path)
            pdf_path = os.path.splitext(docx_path)[0] + '.pdf'
            # 17 = wdFormatPDF
            doc.SaveAs(pdf_path, FileFormat=17)
            doc.Close()
            word.Quit()
            print(f" [DOCX to PDF] Successfully converted {docx_path} to {pdf_path} using win32com.")
            return pdf_path
        except Exception as e:
            print(f" [DOCX to PDF ERROR] Failed to convert {docx_path} to PDF using win32com: {e}")
            return None
    except Exception as e:
        print(f" [DOCX to PDF ERROR] Failed to convert {docx_path} to PDF: {e}")
        return None

```

Figure 3.9: preprocess_cv.py (DOCX to PDF Conversion)

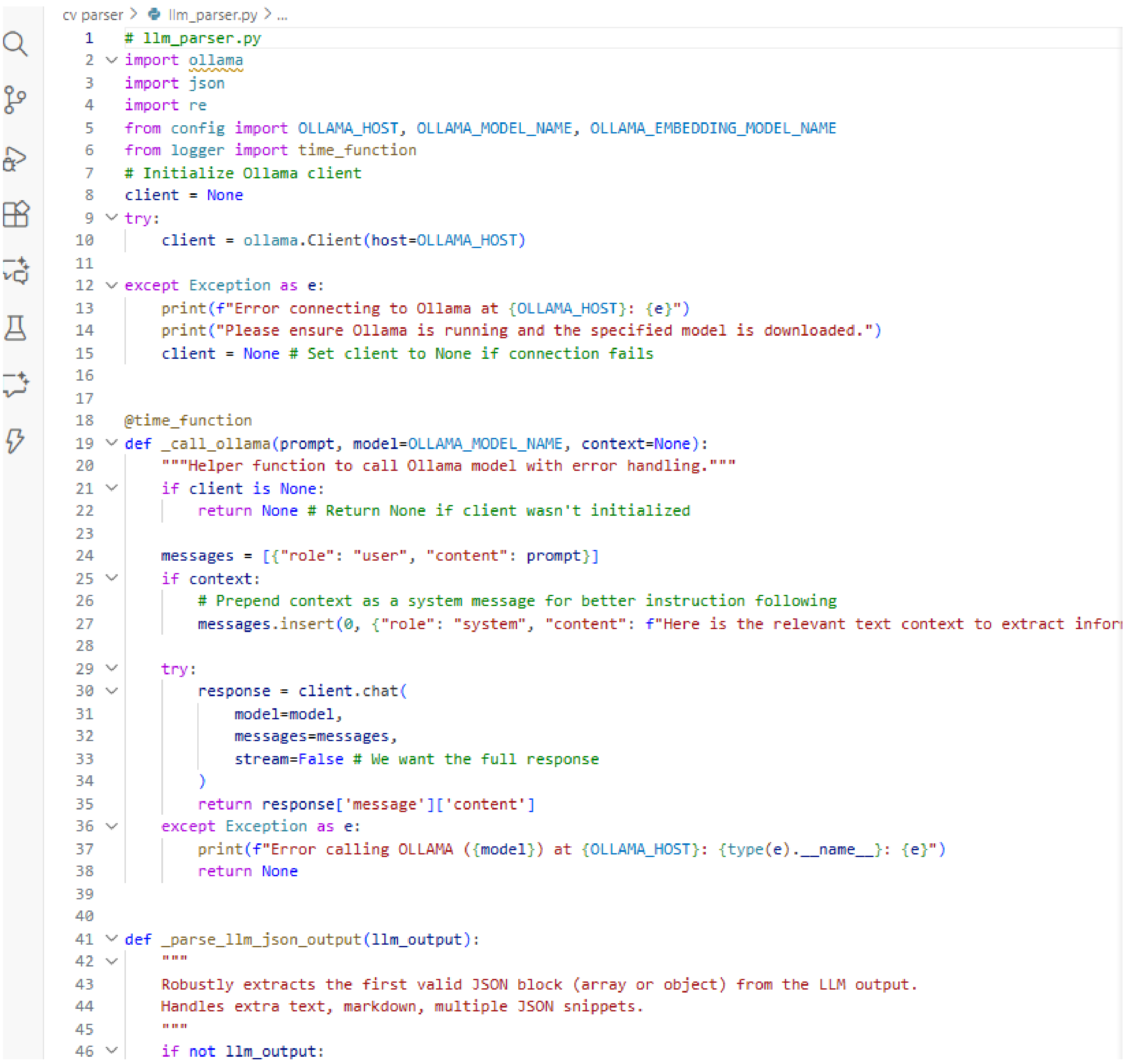
Our script employs the docx2pdf library, when converting.docx files to PDFs. However, for Windows-based conversions it relies on win32com, and includes error handling to check that resumes are properly converted before text extraction.



```python

def extract_contact_info(text):
    contact_info = {"email": None, "phone_numbers": [], "urls": []}

    # Email
    email_matches = re.findall(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b', text)
    if email_matches:
        unique_emails_ordered = sorted(list(set(email_matches)), key=text.find)
        if unique_emails_ordered:
            contact_info["email"] = unique_emails_ordered[0]

    # Phone Numbers
    phone_matches = re.findall(
        r'(?:\+?\d{1,3}[-.\s]?)?\(?\d{2,4}\)?[-.\s]?\d{3,4}[-.\s]?\d{4,6}\b',
        text
    )

    valid_phones = []
    for phone in set(phone_matches):
        normalized_phone = re.sub(r'[()\s.-]', '', phone)
        if len(normalized_phone) >= 10 and len(normalized_phone) <= 15 and normalized_phone.replace('+', '').
            if len(normalized_phone) == 10 and not normalized_phone.startswith(('+', '0')):
                normalized_phone = '+' + normalized_phone
            elif len(normalized_phone) == 11 and normalized_phone.startswith('0'):
                normalized_phone = '+91' + normalized_phone[1:]
            valid_phones.append(normalized_phone)
    contact_info["phone_numbers"] = list(set(valid_phones))

    # ✅ Correct URL Extraction
    url_matches = re.findall(r'https?://[^\s)>\]"]+', text)
    contact_info["urls"] = list(set(url_matches))

    return contact_info

def chunk_text(text, max_chunk_size=1500, overlap=80):
    """
    Chunks text into smaller pieces with overlap for RAG.
    Prioritizes splitting at natural boundaries (e.g., double newlines).
    """
    chunks = []
    paragraphs = text.split('\n\n') # Split by double newlines (paragraphs)
    current_chunk = ""

    for para in paragraphs:
```

Figure 3.10: regex_parser.py (Contact Info & Chunk Extraction)

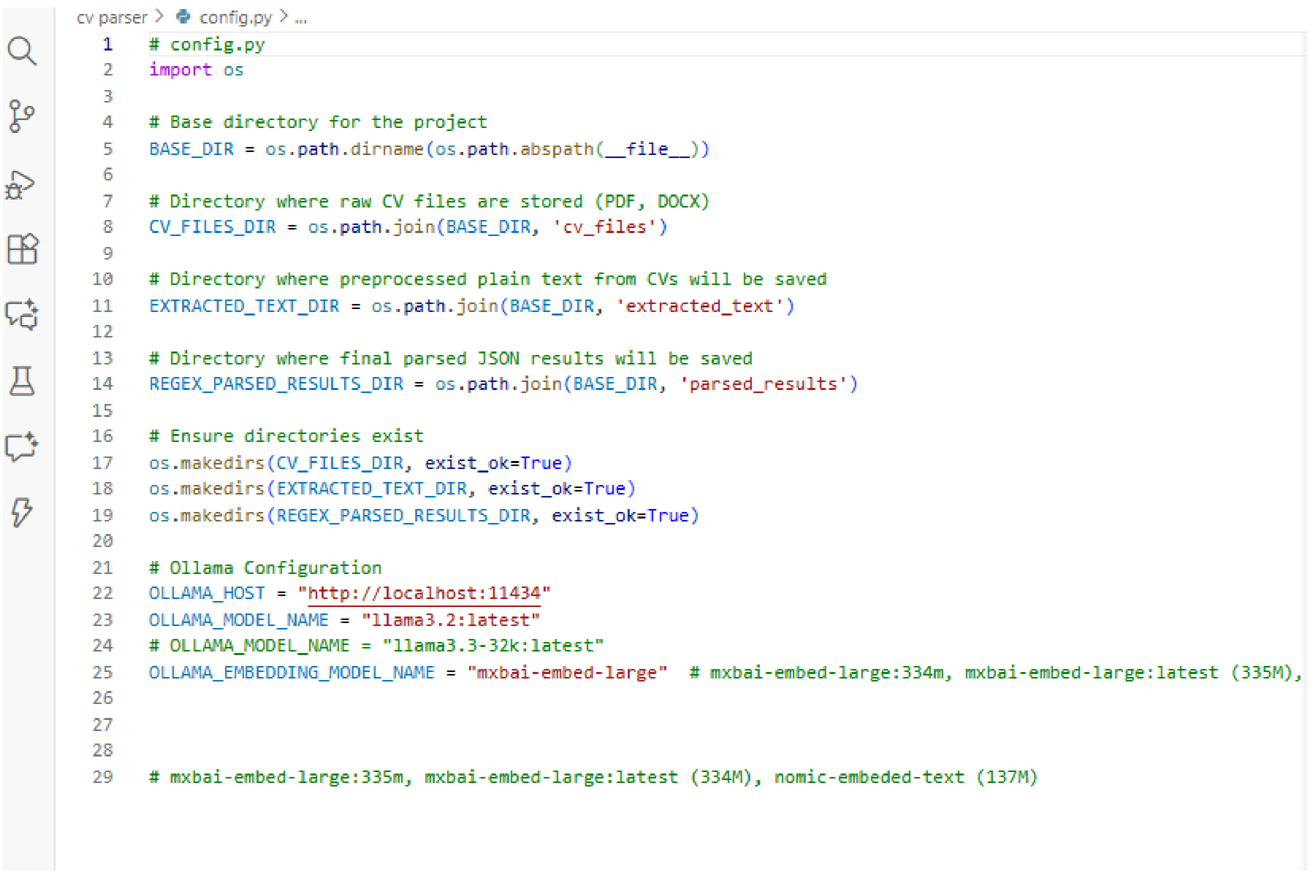
This code shows the regex parser i use to pull out contact info like emails phone numbers, urls, and other data from resume text. it also cleans up and normalizes phone numbers and breaks the text into chunks to go to the lln later. this code helps me get your info from your resume text in a way that makes sense.



```python
# llm_parser.py
import ollama
import json
import re
from config import OLLAMA_HOST, OLLAMA_MODEL_NAME, OLLAMA_EMBEDDING_MODEL_NAME
from logger import time_function
# Initialize Ollama client
client = None
try:
    client = ollama.Client(host=OLLAMA_HOST)

except Exception as e:
    print(f"Error connecting to Ollama at {OLLAMA_HOST}: {e}")
    print("Please ensure Ollama is running and the specified model is downloaded.")
    client = None # Set client to None if connection fails


@time_function
def _call_ollama(prompt, model=OLLAMA_MODEL_NAME, context=None):
    """Helper function to call Ollama model with error handling."""
    if client is None:
        return None # Return None if client wasn't initialized

    messages = [{"role": "user", "content": prompt}]
    if context:
        # Prepend context as a system message for better instruction following
        messages.insert(0, {"role": "system", "content": f"Here is the relevant text context to extract infor

    try:
        response = client.chat(
            model=model,
            messages=messages,
            stream=False # We want the full response
        )
        return response['message']['content']
    except Exception as e:
        print(f"Error calling OLLAMA ({model}) at {OLLAMA_HOST}: {type(e).__name__}: {e}")
        return None


def _parse_llm_json_output(llm_output):
    """
    Robustly extracts the first valid JSON block (array or object) from the LLM output.
    Handles extra text, markdown, multiple JSON snippets.
    """
    if not llm_output:
```

Figure 3.11: llm_parser.py (LLM-Based Resume Processing)

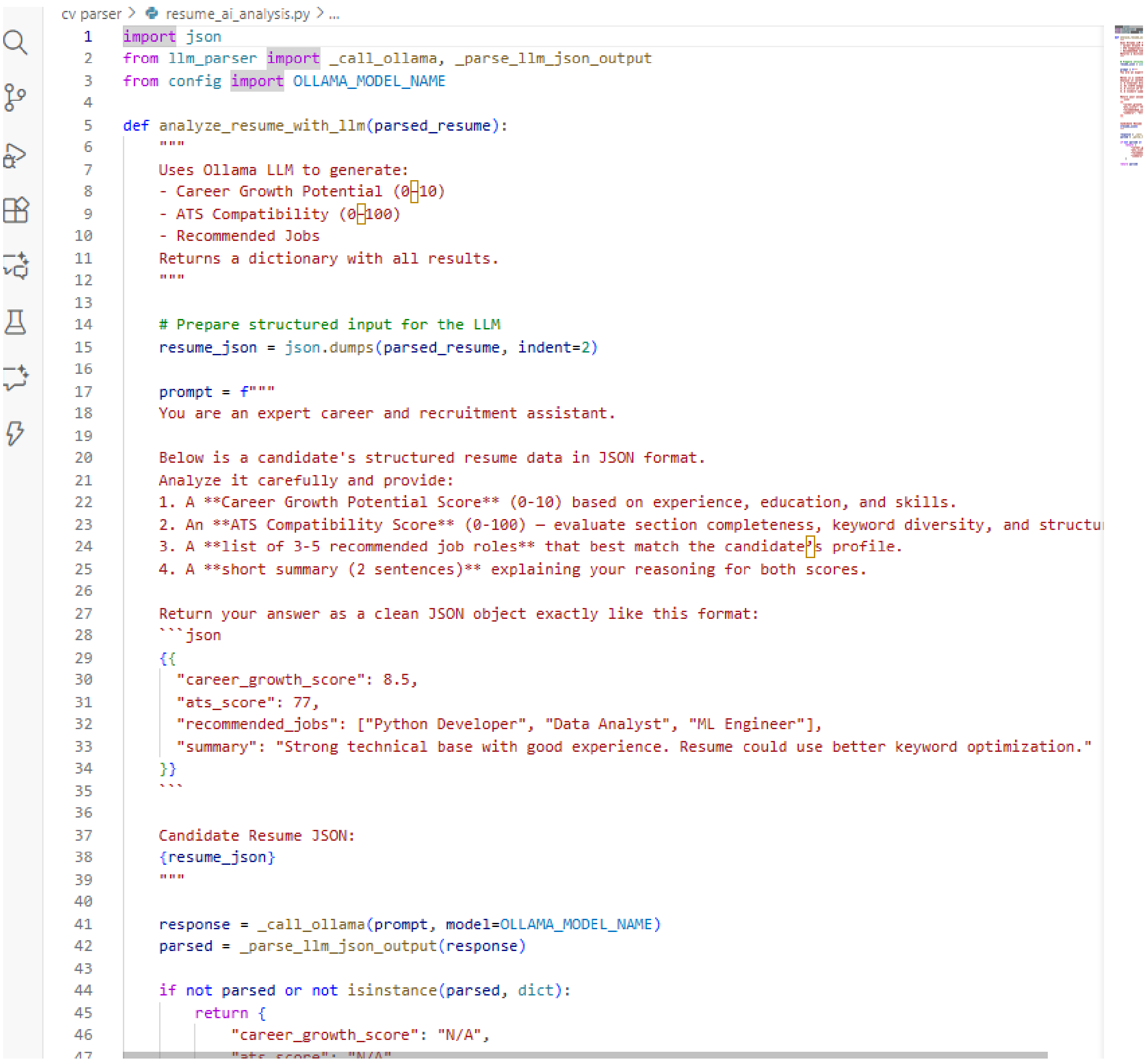
This shows how this part makes the LLM read and take out detailed info from resumes. It also covers prompt use, setting context, and getting answers from the Ollama client. The code pushes prompts in a set way and gives back clean resume info.



```
# config.py
import os

# Base directory for the project
BASE_DIR = os.path.dirname(os.path.abspath(__file__))

# Directory where raw CV files are stored (PDF, DOCX)
CV_FILES_DIR = os.path.join(BASE_DIR, 'cv_files')

# Directory where preprocessed plain text from CVs will be saved
EXTRACTED_TEXT_DIR = os.path.join(BASE_DIR, 'extracted_text')

# Directory where final parsed JSON results will be saved
REGEX_PARSED_RESULTS_DIR = os.path.join(BASE_DIR, 'parsed_results')

# Ensure directories exist
os.makedirs(CV_FILES_DIR, exist_ok=True)
os.makedirs(EXTRACTED_TEXT_DIR, exist_ok=True)
os.makedirs(REGEX_PARSED_RESULTS_DIR, exist_ok=True)

# Ollama Configuration
OLLAMA_HOST = "http://localhost:11434"
OLLAMA_MODEL_NAME = "llama3.2:latest"
# OLLAMA_MODEL_NAME = "llama3.3-32k:latest"
OLLAMA_EMBEDDING_MODEL_NAME = "mxbai-embed-large"  # mxbai-embed-large:334m, mxbai-embed-large:latest (335M),



# mxbai-embed-large:335m, mxbai-embed-large:latest (334M), nomic-embeded-text (137M)
```

Figure 3.12: config.py (System Configuration & Directory Setup)

This figure shows the config file with projects folders, storage, environment and more. It also display setup for Ollama model names and embedings models parsing the content.

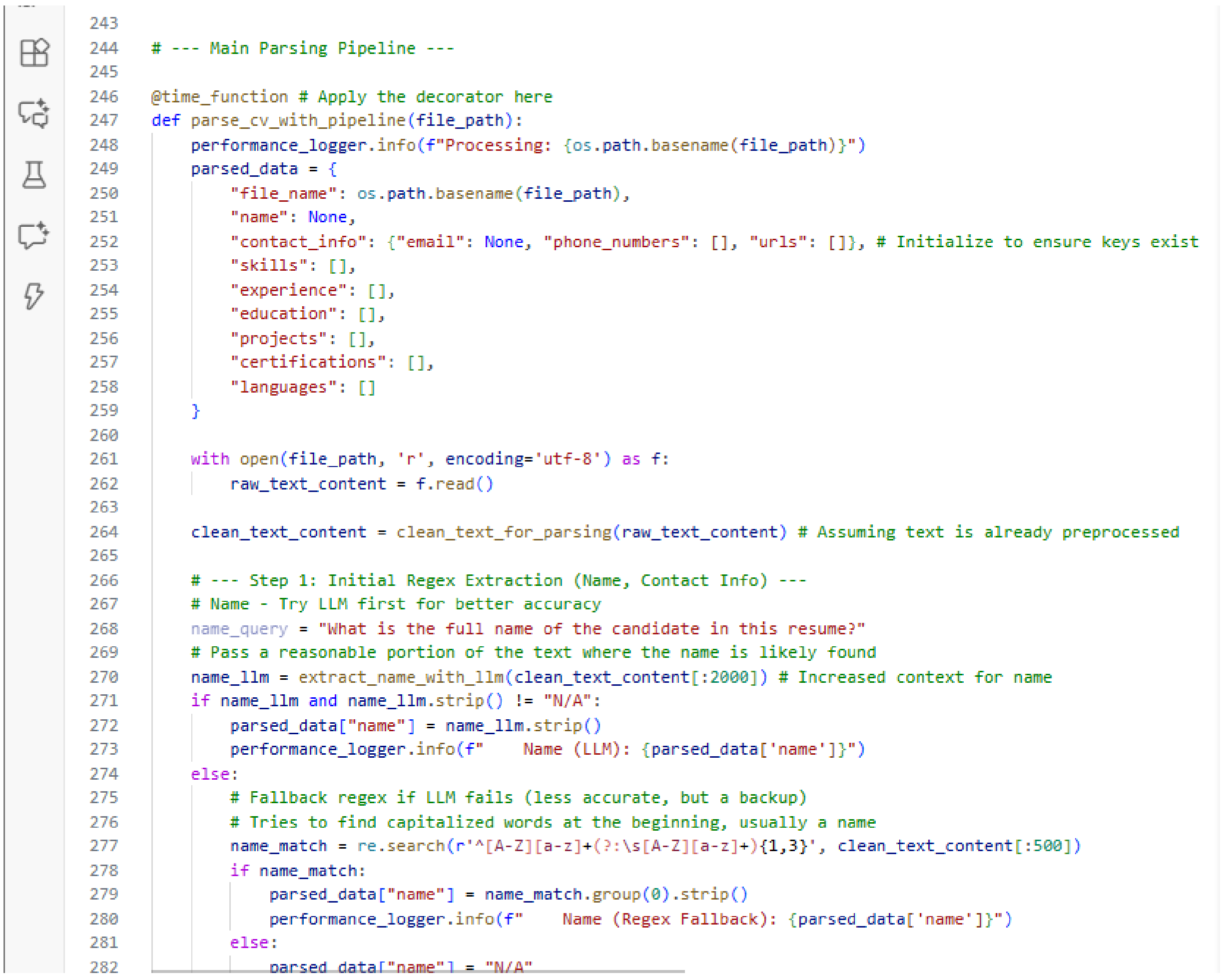


~~~python
import json
from llm_parser import _call_ollama, _parse_llm_json_output
from config import OLLAMA_MODEL_NAME

def analyze_resume_with_llm(parsed_resume):
    """
    Uses Ollama LLM to generate:
    - Career Growth Potential (0–10)
    - ATS Compatibility (0–100)
    - Recommended Jobs
    Returns a dictionary with all results.
    """

    # Prepare structured input for the LLM
    resume_json = json.dumps(parsed_resume, indent=2)

    prompt = f"""
    You are an expert career and recruitment assistant.

    Below is a candidate's structured resume data in JSON format.
    Analyze it carefully and provide:
    1. A **Career Growth Potential Score** (0-10) based on experience, education, and skills.
    2. An **ATS Compatibility Score** (0-100) — evaluate section completeness, keyword diversity, and structu
    3. A **list of 3-5 recommended job roles** that best match the candidate’s profile.
    4. A **short summary (2 sentences)** explaining your reasoning for both scores.

    Return your answer as a clean JSON object exactly like this format:
    ```json
    {{
      "career_growth_score": 8.5,
      "ats_score": 77,
      "recommended_jobs": ["Python Developer", "Data Analyst", "ML Engineer"],
      "summary": "Strong technical base with good experience. Resume could use better keyword optimization."
    }}
    ~~~

    Candidate Resume JSON:
    {resume_json}
    """

    response = _call_ollama(prompt, model=OLLAMA_MODEL_NAME)
    parsed = _parse_llm_json_output(response)

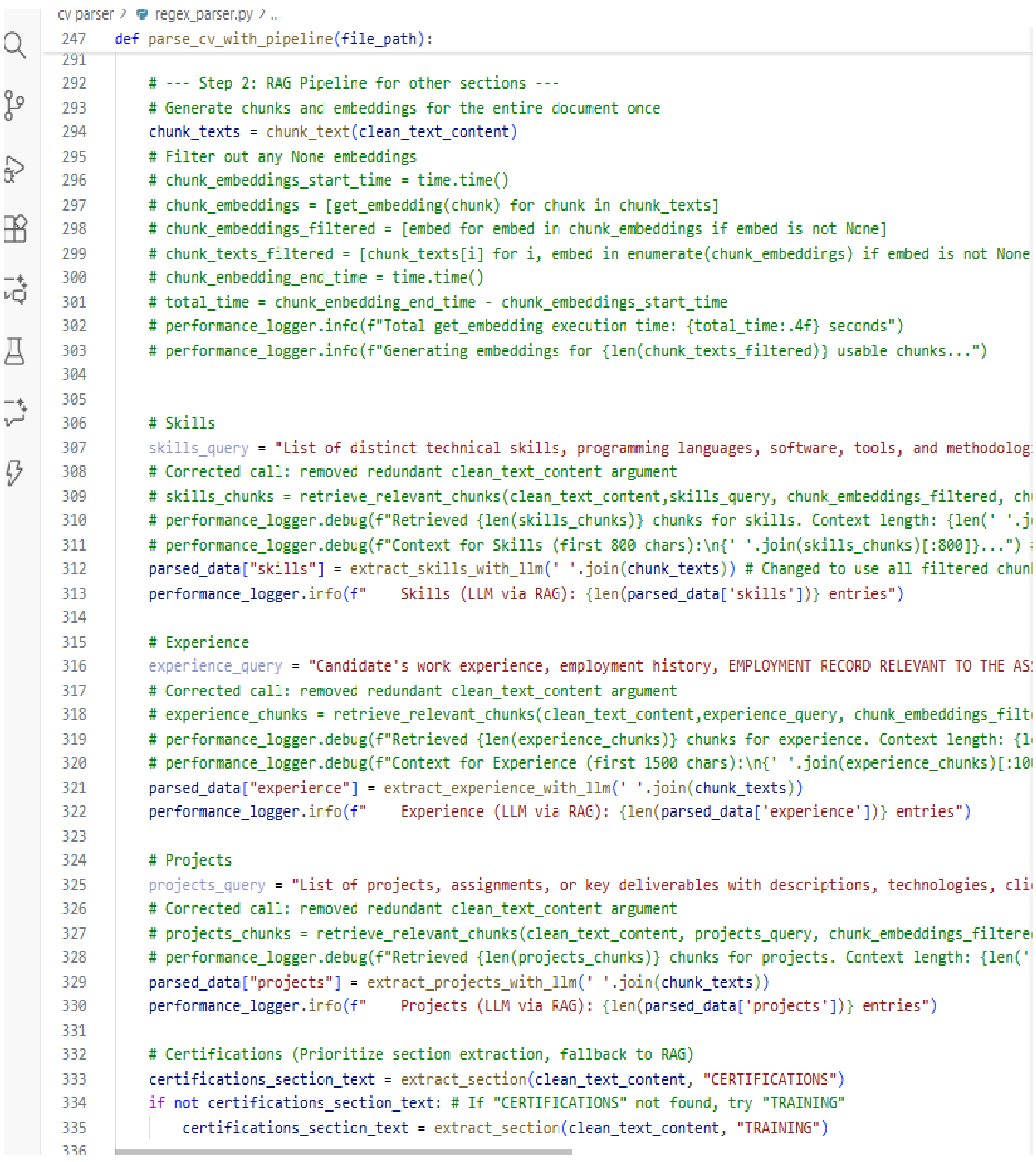
    if not parsed or not isinstance(parsed, dict):
        return {
            "career_growth_score": "N/A",
            "ats_score": "N/A",
```

Figure 3.13: Resume Analysis Module Using LLM

In this picture is the Python program that makes the resume data and types it to an LLM for review. The prompt template shows how career growth, ATS score and job advice are made. It is the main engine of the AI resume review feature.
```



```python

# --- Main Parsing Pipeline ---

@time_function # Apply the decorator here
def parse_cv_with_pipeline(file_path):
    performance_logger.info(f"Processing: {os.path.basename(file_path)}")
    parsed_data = {
        "file_name": os.path.basename(file_path),
        "name": None,
        "contact_info": {"email": None, "phone_numbers": [], "urls": []}, # Initialize to ensure keys exist
        "skills": [],
        "experience": [],
        "education": [],
        "projects": [],
        "certifications": [],
        "languages": []
    }

    with open(file_path, 'r', encoding='utf-8') as f:
        raw_text_content = f.read()

    clean_text_content = clean_text_for_parsing(raw_text_content) # Assuming text is already preprocessed

    # --- Step 1: Initial Regex Extraction (Name, Contact Info) ---
    # Name - Try LLM first for better accuracy
    name_query = "What is the full name of the candidate in this resume?"
    # Pass a reasonable portion of the text where the name is likely found
    name_llm = extract_name_with_llm(clean_text_content[:2000]) # Increased context for name
    if name_llm and name_llm.strip() != "N/A":
        parsed_data["name"] = name_llm.strip()
        performance_logger.info(f"    Name (LLM): {parsed_data['name']}")
    else:
        # Fallback regex if LLM fails (less accurate, but a backup)
        # Tries to find capitalized words at the beginning, usually a name
        name_match = re.search(r'^[A-Z][a-z]+(?:\s[A-Z][a-z]+){1,3}', clean_text_content[:500])
        if name_match:
            parsed_data["name"] = name_match.group(0).strip()
            performance_logger.info(f"    Name (Regex Fallback): {parsed_data['name']}")
        else:
            parsed_data["name"] = "N/A"
```

Figure 3.14: Main Parsing Pipeline Initialization

This code shows the parse_cv_with_pipeline function that starts processing a resume file. It begins by creating the main data object (parsed_data) with empty fields for skills, experience, and education. It then performs the first extraction pass, trying to find the candidate's name first with an LLM, then with a regex fallback, along with contact info.

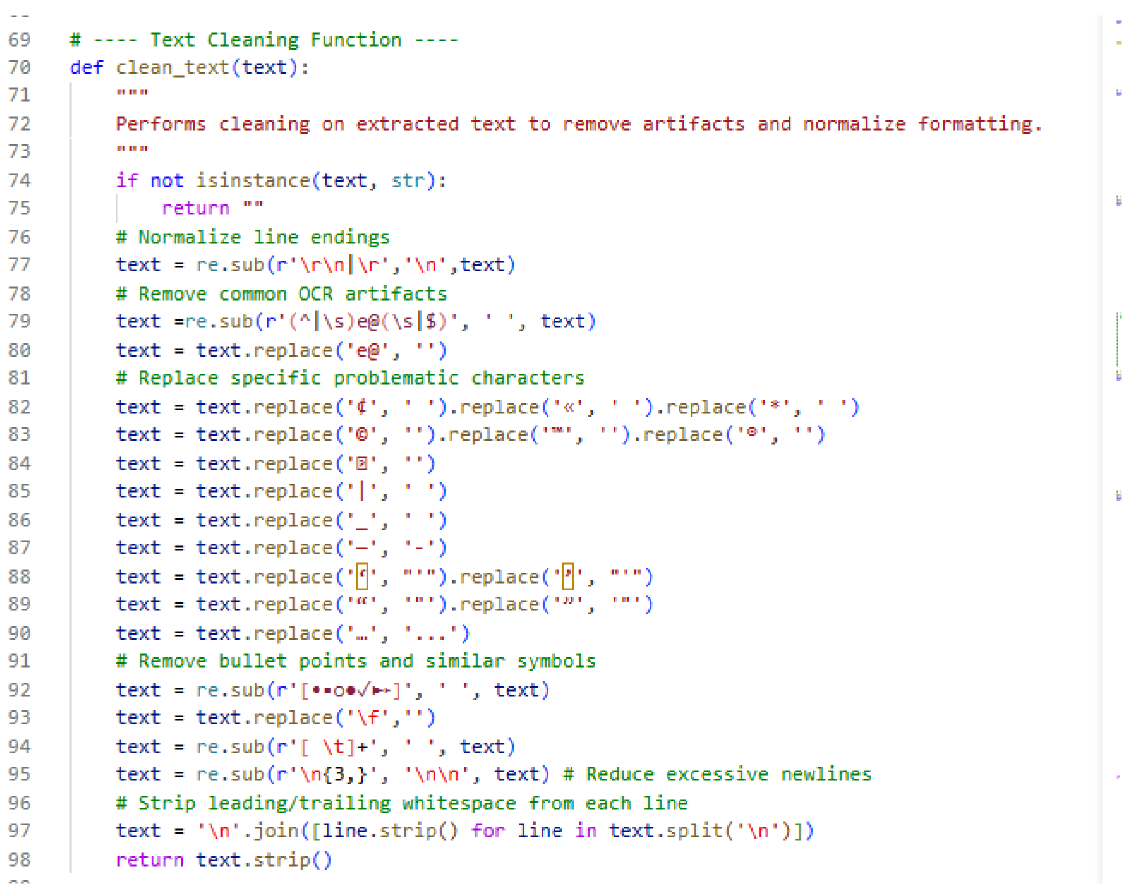


cv parser > regex_parser.py > ...

```python
def parse_cv_with_pipeline(file_path):

    # --- Step 2: RAG Pipeline for other sections ---
    # Generate chunks and embeddings for the entire document once
    chunk_texts = chunk_text(clean_text_content)
    # Filter out any None embeddings
    # chunk_embeddings_start_time = time.time()
    # chunk_embeddings = [get_embedding(chunk) for chunk in chunk_texts]
    # chunk_embeddings_filtered = [embed for embed in chunk_embeddings if embed is not None]
    # chunk_texts_filtered = [chunk_texts[i] for i, embed in enumerate(chunk_embeddings) if embed is not None
    # chunk_enbedding_end_time = time.time()
    # total_time = chunk_enbedding_end_time - chunk_embeddings_start_time
    # performance_logger.info(f"Total get_embedding execution time: {total_time:.4f} seconds")
    # performance_logger.info(f"Generating embeddings for {len(chunk_texts_filtered)} usable chunks...")


    # Skills
    skills_query = "List of distinct technical skills, programming languages, software, tools, and methodolog
    # Corrected call: removed redundant clean_text_content argument
    # skills_chunks = retrieve_relevant_chunks(clean_text_content,skills_query, chunk_embeddings_filtered, ch
    # performance_logger.debug(f"Retrieved {len(skills_chunks)} chunks for skills. Context length: {len(' '.j
    # performance_logger.debug(f"Context for Skills (first 800 chars):\n{' '.join(skills_chunks)[:800]}...")
    parsed_data["skills"] = extract_skills_with_llm(' '.join(chunk_texts)) # Changed to use all filtered chun
    performance_logger.info(f"    Skills (LLM via RAG): {len(parsed_data['skills'])} entries")

    # Experience
    experience_query = "Candidate's work experience, employment history, EMPLOYMENT RECORD RELEVANT TO THE AS
    # Corrected call: removed redundant clean_text_content argument
    # experience_chunks = retrieve_relevant_chunks(clean_text_content,experience_query, chunk_embeddings_filt
    # performance_logger.debug(f"Retrieved {len(experience_chunks)} chunks for experience. Context length: {l
    # performance_logger.debug(f"Context for Experience (first 1500 chars):\n{' '.join(experience_chunks)[:10
    parsed_data["experience"] = extract_experience_with_llm(' '.join(chunk_texts))
    performance_logger.info(f"    Experience (LLM via RAG): {len(parsed_data['experience'])} entries")

    # Projects
    projects_query = "List of projects, assignments, or key deliverables with descriptions, technologies, cli
    # Corrected call: removed redundant clean_text_content argument
    # projects_chunks = retrieve_relevant_chunks(clean_text_content, projects_query, chunk_embeddings_filtere
    # performance_logger.debug(f"Retrieved {len(projects_chunks)} chunks for projects. Context length: {len('
    parsed_data["projects"] = extract_projects_with_llm(' '.join(chunk_texts))
    performance_logger.info(f"    Projects (LLM via RAG): {len(parsed_data['projects'])} entries")

    # Certifications (Prioritize section extraction, fallback to RAG)
    certifications_section_text = extract_section(clean_text_content, "CERTIFICATIONS")
    if not certifications_section_text: # If "CERTIFICATIONS" not found, try "TRAINING"
        certifications_section_text = extract_section(clean_text_content, "TRAINING")
```

Figure 3.15: RAG-Powered Section Extraction

This part shows how RAG works step by step: Chunking and creating embeddings for the document content.. It shows how we find the parts (Skills, Experience, Projects) we want by running a similarity search to get the parts we need.. The last parts that make sense now get sent to an LLM (extract_skills, extract_experience, etc.) and it makes structured, parsed data for each part.



```python
# ---- Text Cleaning Function ----
def clean_text(text):
    """
    Performs cleaning on extracted text to remove artifacts and normalize formatting.
    """
    if not isinstance(text, str):
        return ""
    # Normalize line endings
    text = re.sub(r'\r\n|\r','\n',text)
    # Remove common OCR artifacts
    text =re.sub(r'(^|\s)e@(\s|$)', ' ', text)
    text = text.replace('e@', '')
    # Replace specific problematic characters
    text = text.replace('¢', ' ').replace('«', ' ').replace('*', ' ')
    text = text.replace('©', '').replace('™', '').replace('®', '')
    text = text.replace('⊠', '')
    text = text.replace('|', ' ')
    text = text.replace('_', ' ')
    text = text.replace('—', '-')
    text = text.replace('‘', "'").replace('’', "'")
    text = text.replace('“', '"').replace('”', '"')
    text = text.replace('…', '...')
    # Remove bullet points and similar symbols
    text = re.sub(r'[•▪o●√►→]', ' ', text)
    text = text.replace('\f','')
    text = re.sub(r'[ \t]+', ' ', text)
    text = re.sub(r'\n{3,}', '\n\n', text) # Reduce excessive newlines
    # Strip leading/trailing whitespace from each line
    text = '\n'.join([line.strip() for line in text.split('\n')])
    return text.strip()
```

Figure 3.16: clean_text Preprocessing Function

The function clean_text(text) is a helper function that aims to make the raw text coming from the document more manageable. It employs pattern matching to unify line ends, remove typical OCR issues like extra @ and symbols, and replace complicated characters that are not standard. Finally, it removes bullet points, excessive new lines, and any spaces on the sides to make the text cleaner for the parsing step.

3.6 Key Challenges:

- During system development, various challenges were encountered. One significant problem that was averted by introducing stringent validation rules and supportive messages in the user interface.
- The next difficulty was the creation of a resume generator that would be compatible with numerous templates. We had to be sure that the layout would remain clean even if users inputted long project descriptions or multiple skills. For this, we had to make extra changes in the template formatting.
- It was also necessary to concentrate on the management of the job tracker, especially for the purpose of keeping the interface simple and at the same time allowing users to update statuses easily. To solve this problem, we have made the job-entry form and the status options more straightforward.
- There were also some technical issues, for example, API failures, front-back communication errors, and formatting issues when generating PDF. We corrected these issues by debugging, making minor code changes, and doing repetitive testing. In fact, every single challenge was an opportunity to grow and thus the final system is more stable and reliable.

CHAPTER 4: TESTING

4.1 Testing Strategy :

A systematic testing approach was implemented during the entire development process to make sure the system was functioning smoothly and delivering the expected output. Rather than waiting until the very end, tests were performed periodically after each module was completed. This tactic allowed for the early detection of the problems and consequently, the issues could be fixed before they had the chance to become of a larger scale.

Testing of the frontend was mainly concentrated on verifying if the pages loaded properly, the forms received the right inputs, the buttons were working to the expected and the users were able to navigate the pages without encountering any errors. I personally tested each functionality by inputting different types of data, such as valid, invalid, and incomplete, and observed how the application behaved.

API testing was the main focus for the backend. Postman and similar tools were used to check the performance of each API endpoint. This work consisted of sending test requests and ensuring that the server returned the requested data correctly as well as confirming that the information was accurately stored in the database. API testing also supported the confirmation that the resume generator and job application tracker were the components that had a connection with the frontend.

Database testing ensured the storing, updating, and retrieving of user data, job details, and resume information without facing any problems. Different operations were performed on the records, and by doing this, it was very easy to spot and fix any discrepancies.

The employed testing strategy was manual UI testing, API testing via Postman, and data validation at the database level. The combination of these methods was a guarantee that the different parts of the system were functioning together without any problems.

4.2 Test Cases and Outcomes :

In the testing phase, individually each module of the system was tested.The main focus was on documents to ensure that resumes were the files that went through correctly, parsing functions were the ones that took out the right info, and LLM was the one that gave the output that was clean and well structured.First of all, most of the test cases were done successfully, and a few small bugs like unreadable scanned PDFs were detected and fixed with error messages.The tests have confirmed that the system is stable, reliable, and able to perform as anticipated for various resume formats.

Table 4.1 Test Case Table

TC No.	Feature Tested	Test Input	Expected Output	Actual Output	Status
TC-01	Upload Resume File	Valid PDF file	File uploaded successfully	Uploaded correctly	Pass
TC-02	Upload Invalid File	Image file (.jpg)	Error message displayed	Error displayed	Pass
TC-03	DOCX to PDF Conversion	.docx file	Converted PDF saved	Conversion completed	Pass
TC-04	Text Extraction	Clean PDF resume	Extracted text output	Text extracted correctly	Pass
TC-05	Regex Parsing	Resume text block	Email, phone, links detected	Information extracted correctly	Pass
TC-06	LLM Parsing	Experience section text	JSON response	JSON response generated	Pass
TC-07	Scanned PDF Handling	Scanned resume	Show warning / handle gracefully	Warning shown	Pass
TC-08	Invalid Email Extraction	Text with no email	Return empty list	Empty list returned	Pass

TC No.	Feature Tested	Test Input	Expected Output	Actual Output	Status
TC-09	File Save Operation	Parsed JSON	JSON saved in folder	Saved correctly	Pass
TC-10	Full Pipeline Test	DOCX → PDF → Extract → Parse	Final JSON output	Working end-to-end	Pass

Most of the testing work have pinpointed a few small situations of file mishandling and text extraction that were patched during the same testing process. Both DOCX and PDF formats were handled efficiently by the system. Whenever a component was missing, the fallback strategies were able to take over smoothly. The parser was able to extract email IDs, phone numbers, links, and text chunks accurately most of the time. End-to-end tests were able to demonstrate that the whole pipeline, i.e., from file upload to final JSON output, is done continuously without any crashes. These outcomes are a source of great confidence and imply that the project can be further developed and integrated with the other modules..

4.2.1 Test Report Conclusion :

After a full round of tests, it can be concluded that the system complies with the main criteria of the first phase. All the core functions operate smoothly, among them file conversion, text extraction, and structured information parsing. The implemented error-handling mechanism contributes significantly to the prevention of unanticipated failures. Having finished these tests with success, the project is now ready to transition to Phase 2 development and user-level deployment.

CHAPTER 5: RESULTS AND EVALUATION

5.1 Results:

Once the different modules of the system were installed, the team proceeded to test each module individually. Subsequently, we examined their interactions in the complete workflow. This phase's primary objective was to verify the functionalities of the features. Additionally, we aimed to ascertain their stability and reliability when used by students in everyday situations.

The outcomes of the tests provided us with a more vivid picture of the system's performance in terms of precision, speed, and general user-friendliness.

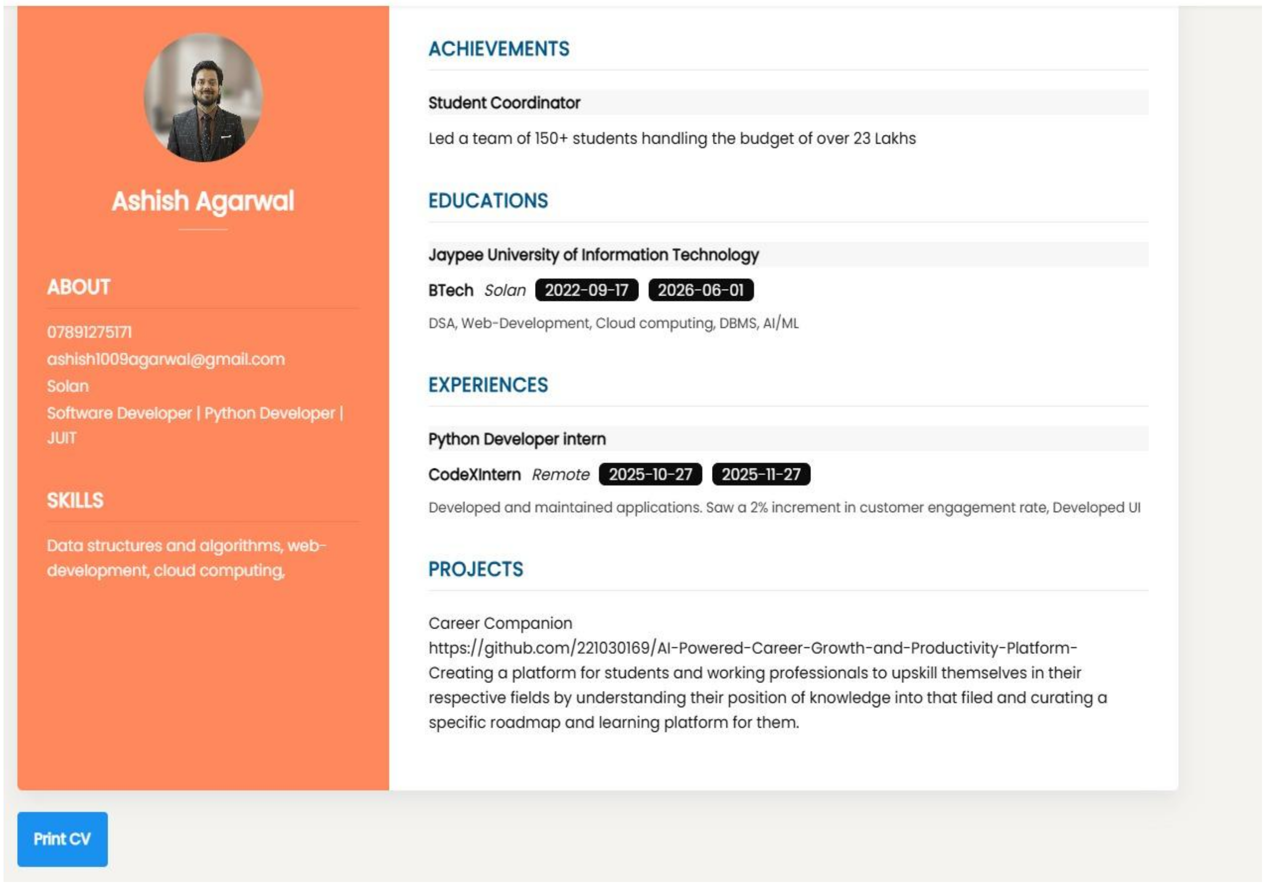
5.1.1 Resume Generator Results :

The resume generator underwent various types of resume testing, such as simple text resumes, PDF versions, creative layouts, and those with scanned images. In general, it correctly processed the details given by the users and converted them into a structured, professional template. Most of the times, the formatting, alignment, and spacing of the generated resume were kept consistent.

Users have confirmed that the output looked professional and that only a few manual editing were needed. The system handled the formatting issues gracefully even if the text spacing or alignment in the input file were incorrect.

One significant enhancement in text extraction could be recognized after the refinement of the preprocessing pipeline. Beforehand, it was hard to find the missing fields in some resumes due to unusual formats, but now after making changes, the extraction has become more stable and reliable. The resume generator was also put to the test with low-resolution PDFs. While some parts of the information had to be manually corrected, the tool was able to extract most of the text.

Additionally, the effective implementation of AI-based job-role suggestions was another important outcome. The system could, after generating a resume, suggest future career paths based on the user's skills, project work, and education. This feature was very helpful to the user as students were provided with instant insights into the job sectors that were suitable for them.


Ashish Agarwal

ABOUT

07891275171
ashish1009agarwal@gmail.com
Solan
Software Developer | Python Developer | JUIT

SKILLS

Data structures and algorithms, web-development, cloud computing,

ACHIEVEMENTS

Student Coordinator
Led a team of 150+ students handling the budget of over 23 Lakhs

EDUCATIONS

Jaypee University of Information Technology
BTech *Solan* 2022-09-17 2026-06-01
DSA, Web-Development, Cloud computing, DBMS, AI/ML

EXPERIENCES

Python Developer intern
CodeXIntern *Remote* 2025-10-27 2025-11-27
Developed and maintained applications. Saw a 2% increment in customer engagement rate, Developed UI

PROJECTS

Career Companion
https://github.com/221030169/AI-Powered-Career-Growth-and-Productivity-Platform-
Creating a platform for students and working professionals to upskill themselves in their respective fields by understanding their position of knowledge into that filed and curating a specific roadmap and learning platform for them.

Figure 5.1 : structured resume

The image shows the final resume output that the system generates after the extraction and structuring phase. The output is visually appealing with a two-column layout.

5.1.2 Cover Letter Generator Results :

The cover letter generator came up with clear and grammatically correct letters based on the user's input. It processed different job roles, such as software engineer, data analyst, and finance assistant, during testing. For each role, the tool altered its writing style.

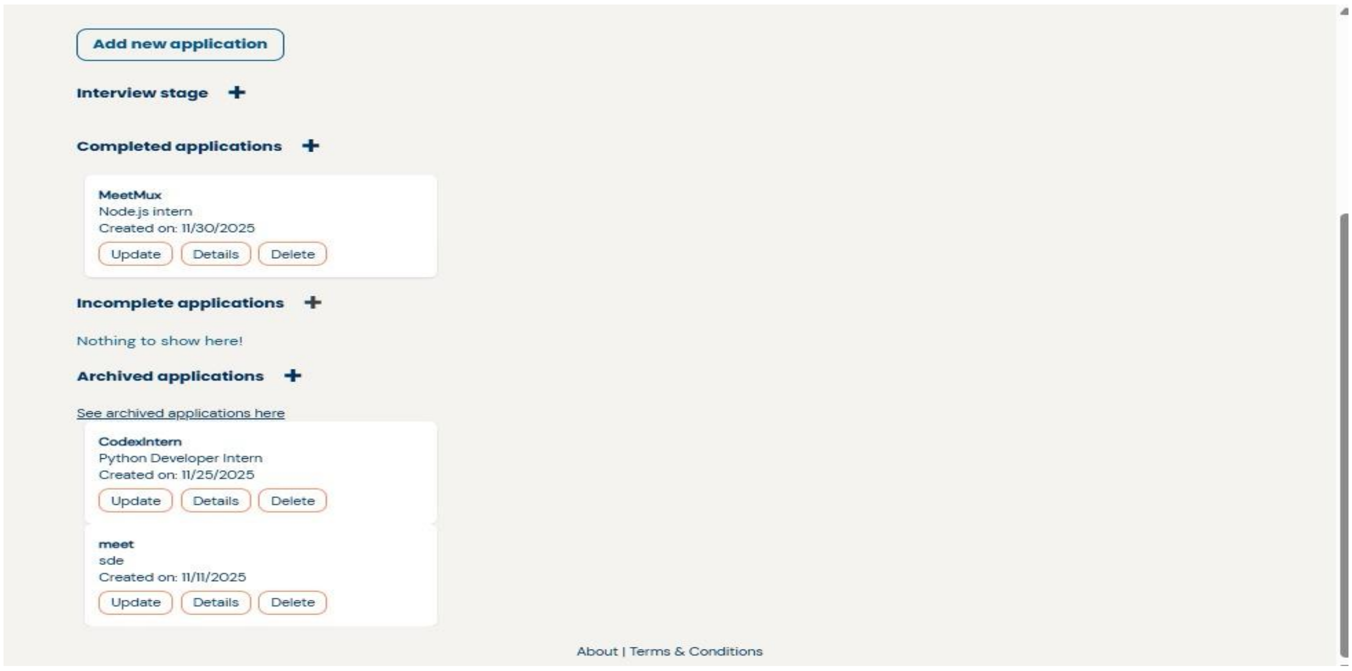
The produced letters shared the same outline: introduction, background, skill alignment, and closing statement. The style used was appropriate and polite.

Personalizing the letter with details taken from the resume sections was probably the most significant feature of the generator. As an example, if a student had previously mentioned Python or Machine Learning in the resume, the cover letter would by default point out those areas of expertise.

By doing so, the user had to do less repetitive work, and the applicant’s profile was represented more accurately.

5.1.3 Job Application Tracker Results :

The job application tracker was tested with various entries, including company names, job roles, application stages, dates, and notes. All entries got saved correctly in the backend. CRUD operations (Create, Read, Update, Delete) were efficient and without any issues. The interface generated a neat look for each application, and the changes made were updated immediately on the screen. When the multiple users scenario was created, the system handled different data logs separately without mixing or overwriting entries. Therefore, backend authentication and routing were working properly. Such a feature will turn out to be a great help for students who are going to many companies for placements and need a single platform to keep up with their progress



Add new application

Interview stage +

Completed applications +

MeetMux
Node.js intern
Created on: 11/30/2025
Update Details Delete

Incomplete applications +

Nothing to show here!

Archived applications +

See archived applications here

CodexIntern
Python Developer Intern
Created on: 11/25/2025
Update Details Delete

meet
sde
Created on: 11/11/2025
Update Details Delete

About | Terms & Conditions

Figure 5.2 Job Application Tracker

Users are allowed to insert and follow up on job applications. They have the freedom to change the status, highlight the necessary details, and even fix the date for an interview.

5.1.4 Backend and API Results :

The development team performed different types of tests with various input combinations on the backend to verify its stability. All API endpoints were reachable within the time limits set. Validators stopped incomplete or invalid data from being stored. API requests for resumes and cover letters were always successful, and large text inputs were handled without any issues. They conducted a test on retries during a short period of internet disconnection, and the system resumed the operation seamlessly once the connection was re-established. Hence, the backend architecture is designed with a considerable degree of fault tolerance..

.1.5 User Feedback Summary:

A handful of students decided to put the system through its paces and provided feedback.

Some of the positive points from the feedback were as follows:

- Easy-to-understand interface
- Simple resume and cover letter creation
- Useful job-role suggestions
- Better presentation and colors than the current free tools
- Direct flow of work without additional steps.

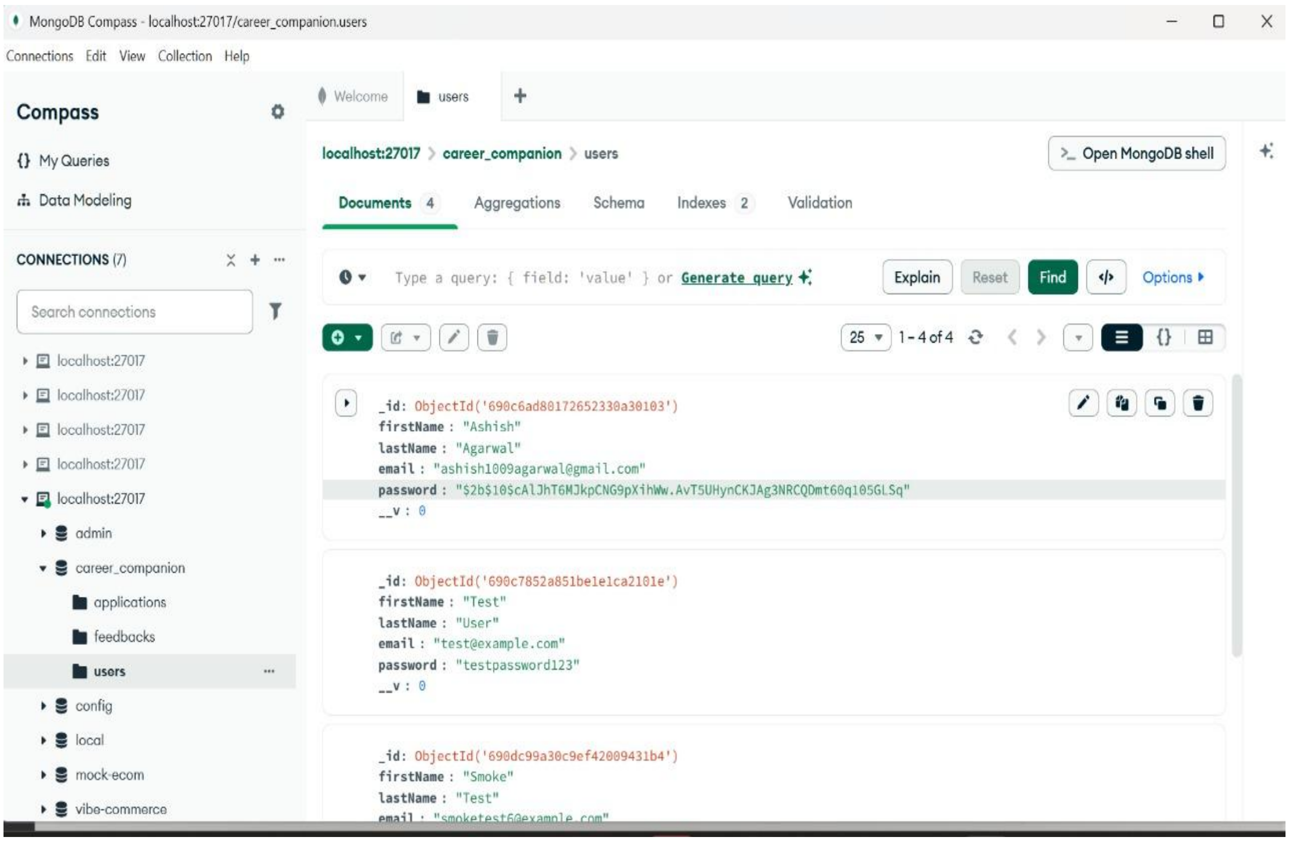
Areas identified for improvement:

- Add multiple resume templates
- Improve styling of dashboard
- Provide an autosave feature in resume creation steps.
- Add a more interactive interview module in the next phase.

Overall, the results show that the Phase 1 system successfully meets its goals

5.1.6 MongoDB Database Showing Stored User Records:

The system has saved user information that includes hashed passwords in MongoDB. The UI, MongoDB Compass, is a confirmation of the proper schema and data flow implementation.



```
_id: ObjectId('690c6ad80172652330a30103')
firstName : "Ashish"
lastName : "Agarwal"
email : "ashish1009agarwal@gmail.com"
password : "$2b$10$cAlJhT6MJkpCNG9pXihWw.AvT5UHynCKJAg3NRCQDmt60q105GLSq"
__v : 0

_id: ObjectId('690c7852a851be1e1ca2101e')
firstName : "Test"
lastName : "User"
email : "test@example.com"
password : "testpassword123"
__v : 0

_id: ObjectId('690dc99a30c9ef42009431b4')
firstName : "Smoke"
lastName : "Test"
email : "smoketest6@example.com"
```

Figure 5.3 MongoDB Database Records

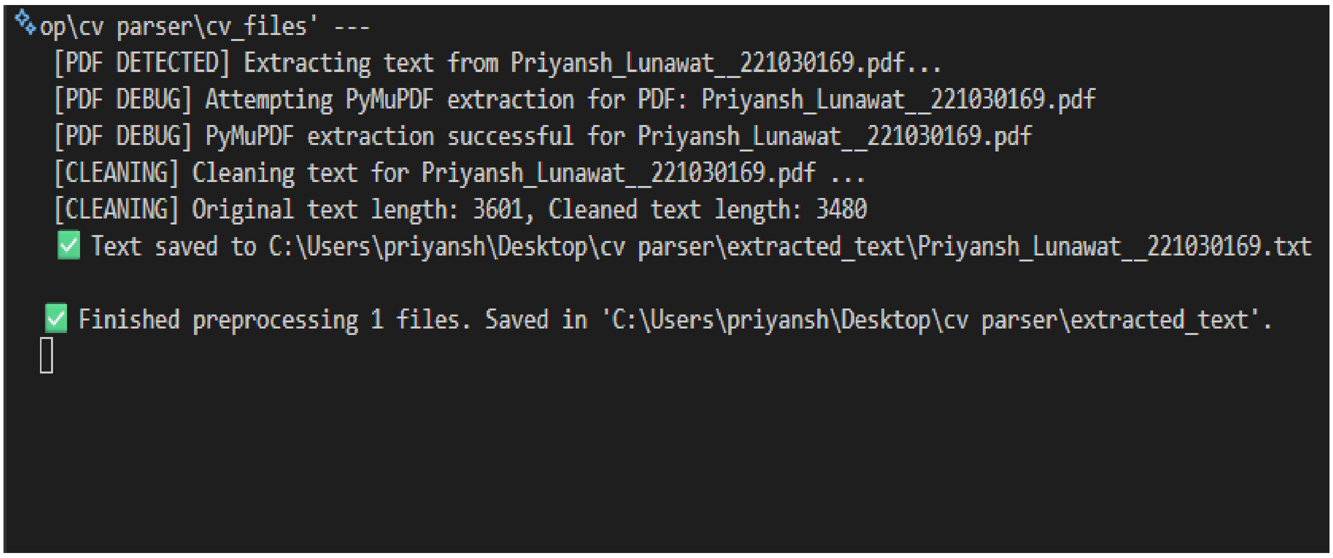
This screenshot illustrates user data saved in the MongoDB collection. It serves as a backing for the front end verification, shows that the password hashing is done correctly, and confirms that CRUD operations are functioning properly.

.1.6 Resume Extraction and Text Cleaning Output :

Automatically extracting clean and usable text from resumes in different formats was one of the main objectives of this work. After the team had implemented PyMuPDF and added a custom cleaning pipeline, the system was able to process PDF resumes with different layouts successfully.

The extraction tool was able to accurately identify the file format.

- Cleaning of the raw texts was done by removing broken spacings, encoding marks, line breaks, and symbols.
- The processed texts were saved automatically into a local folder for further parsing.



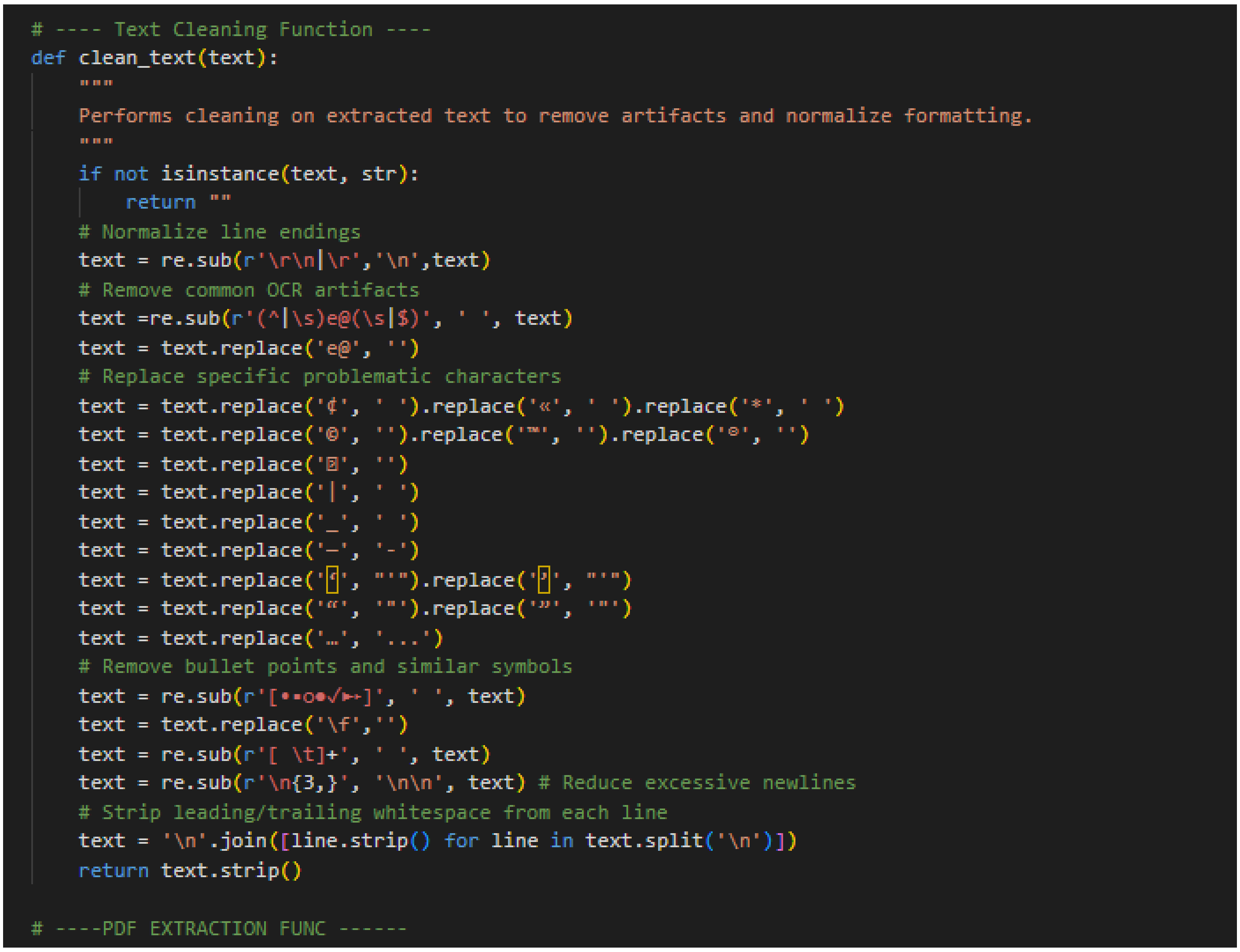
```
op\cv parser\cv_files' ---
  [PDF DETECTED] Extracting text from Priyansh_Lunawat__221030169.pdf...
  [PDF DEBUG] Attempting PyMuPDF extraction for PDF: Priyansh_Lunawat__221030169.pdf
  [PDF DEBUG] PyMuPDF extraction successful for Priyansh_Lunawat__221030169.pdf
  [CLEANING] Cleaning text for Priyansh_Lunawat__221030169.pdf ...
  [CLEANING] Original text length: 3601, Cleaned text length: 3480
  ✅ Text saved to C:\Users\priyansh\Desktop\cv parser\extracted_text\Priyansh_Lunawat__221030169.txt

 ✅ Finished preprocessing 1 files. Saved in 'C:\Users\priyansh\Desktop\cv parser\extracted_text'.
```

Figure 5.4 PDF Text-Extraction Console Output

This capture displays the resume extraction panel. The framework recognizes the PDF sent, runs it through PyMuPDF, extracts the cleaned parts of the document, and stores it in the folder it was specified for.

5.1.8 Resume Parser Preprocessing Pipeline Execution:

The preprocessor file takes a look at the text that is changed from the modification. It demonstrates that the removal rules that were implemented took away the unnecessary tokens and the text became more understandable.



```python
# ---- Text Cleaning Function ----
def clean_text(text):
    """
    Performs cleaning on extracted text to remove artifacts and normalize formatting.
    """
    if not isinstance(text, str):
        return ""
    # Normalize line endings
    text = re.sub(r'\r\n|\r','\n',text)
    # Remove common OCR artifacts
    text =re.sub(r'(^|\s)e@(\s|$)', ' ', text)
    text = text.replace('e@', '')
    # Replace specific problematic characters
    text = text.replace('¢', ' ').replace('«', ' ').replace('*', ' ')
    text = text.replace('©', '').replace('™', '').replace('⊗', '')
    text = text.replace('⊠', '')
    text = text.replace('|', ' ')
    text = text.replace('_', ' ')
    text = text.replace('–', '-')
    text = text.replace('‘', "'").replace('’', "'")
    text = text.replace('“', '"').replace('”', '"')
    text = text.replace('…', '...')
    # Remove bullet points and similar symbols
    text = re.sub(r'[•▪○●√►→]', ' ', text)
    text = text.replace('\f','')
    text = re.sub(r'[ \t]+', ' ', text)
    text = re.sub(r'\n{3,}', '\n\n', text) # Reduce excessive newlines
    # Strip leading/trailing whitespace from each line
    text = '\n'.join([line.strip() for line in text.split('\n')])
    return text.strip()

# ----PDF EXTRACTION FUNC ------
```

Figure 5.5 Pdf Text Cleaning

The screenshot displays the portion of the text (3,601 characters) that the script had to extract from the original content and the reduction to 3,480 characters after the cleanup. This resizeation illustrates the effectiveness of the cleaning.

5.2 Comparison with Existing Solutions:

A comprehensive comparison was made between the developed system and some common online tools such as Canva Resume Builder, Novoresume, Overleaf templates, and AI tools like ChatGPT cover letter prompts. This comparison helped reveal the advantages of the proposed system and possible areas for improvement.

5.2.1 Unified System vs. Single-Purpose Tools :

Almost all existing platforms are heavily inclined to perform a single task, say constructing resumes, producing cover letters, or job tracking. Our system, however, offers the integration of all the necessary career-support tools in one student-friendly dashboard. Consequently, students are less likely to be hopping between different websites and manually updating their information.

5.2.2 Personalization Level :

Well-known resume builders operate fixed templates. They allow a few design changes but do not offer personalized job-role suggestions or context-aware writing. The system created in this project uses AI to offer more tailored resume content and customized cover letters based on the user's actual skills, coursework, and experience.

5.2.3 Accessibility and Student-Centric Approach :

Many commercial tools require paid subscriptions to export resumes or access premium features. Our platform is free and made for students, making it easier to use in an academic setting.

5.2.4 Resume Parsing and Skill Extraction :

Several tools struggle with inconsistent PDF layouts or scanned resumes. The system uses an improved preprocessing pipeline, which showed better extraction accuracy during testing. Although it is not perfect, the system is more flexible than some rigid template-based services.

5.2.5 Job Application Tracker :

Most resume builders do not provide a built-in job application tracker. This feature is very useful, especially for students applying to many internships or jobs. It centralizes all applications together and simplifies the process by removing confusions.

5.2.6. Integration with AI Suggestions :

Most of the current tools hardly offer the structured job-role suggestions that align with the student's skills. The present project is equipped with such a feature, hence it is more suitable for career planning and guidance purposes.

CHAPTER 6: CONCLUSIONS AND FUTURE SCOPE

6.1 Conclusion:

The creation of this single AI-supported career assistance platform has provided an opportunity to understand how technology can actually make the job application process simpler and more efficient for students and early professionals. It has always been the main objective throughout the project to merge several principal functions such as résumé writing, cover letter drafting, interview practicing, job recommendation, and application tracking into one user-friendly environment.

One of the most incredible outcomes of this project is the transition to a unified system that leads users step-by-step from scattered online tools. The platform does not let users get used to different interfaces or rely on generic templates, instead, it allows them to produce documents that showcase their individual strengths. To illustrate, the résumé builder is not only doing the job of formatting the text; it works users out of areas where their explanations can be more clear or effective. The cover letter assistant provides additional support by suggesting ways in which the user's writing can be more specific and directly related to the job.

The interview module is, therefore, just as important a part of the system as well. Numerous students struggle with the ability to speak confidently during interviews, even though they are aware of their skills. By allowing them to practice in a non-stressful environment, the platform helps them to polish their thinking process and give their answers in a more logical way. Meanwhile, the job recommendation and tracking features work together to keep users organized and let them be aware of their progress in the application journey.

To sum up, this is an example of how an efficiently designed system can bridge academic preparation with the demands of the real world. The end product is not only a technical accomplishment but also a comforting instrument that can help people to approach the job market with more confidence, clarity, and readiness.

6.2 Future Scope :

- The next project stage will see the platform getting several critical updates that will make it more powerful, intelligent, and reliable. A significant feature is an AI-powered Exam and Analytics tool. Students may undertake a subject-oriented or skill-based test, and a thorough performance report will be provided. The system will then create the skill development ideas by pinpointing the skill gaps, suggesting the relevant learning resources, and giving an overview of the most suitable career paths. This living report will help students to internalize their strengths and see which areas they have to work on..
- Up to this, the major overhaul will be the AI Interview Module's better version integration. The upgraded version will have numerous interactive interview sessions, speech feedback, confidence scoring, and improvement suggestions, rather than only question-and-answer sets. In this way, students will be able to practice in a lifelike scenario and get ready for the real interview sessions of placements and internships.
- The initiative is additionally committed to upgrading the AI models that are used for resume extraction, text generation, and job-role suggestions in both accuracy and reliability. One of the difficulties with AI content generation is model hallucination, i.e., the mechanism may output inaccurate or irrelevant data. The next development will work on prompt-improving, dataset refining, and verification method implementation to lower most of these issues and also getting safer and more predictable results.
- Additionally, we shall be working on cover letter generation in our next phase as it requires an Open AI API key.
- Overall, the upcoming phase will focus on expanding the system from a document generation tool into a complete career-development ecosystem. With improved analytics, stronger interview preparation, and more reliable AI models, the platform can support students more effectively and grow into a comprehensive solution for academic and professional readiness.

REFERENCES

- [1] T. T. H. Nguyen *et al.*, "SimInterview: Transforming Business Education through LLM-Based Simulated Multilingual Interview Training System*,"* arXiv preprint arXiv:2508.11873, 2025. Available: https://arxiv.org/abs/2508.11873v1
- [2] H. Koshti *et al.*, "AI-Powered Interview Preparation System: Integrating Resume Analysis, HR Simulation, and Technical Skill Assessment," *Journal of Engineering Research and Reports (JERR)*, 2025. Available: https://journaljerr.com/index.php/JERR/article/view/1489
- [3] M. Gayathri Devi *et al.*, "AI Friendly Resume," *International Journal of Scientific Research in Engineering and Management (IJSREM)*, 2024. Available: https://ijsrem.com/download/ai-friendly-resume/
- [4] N. A. Kadam, N. P. Pawar, N. P. Pawar, N. R. Pandit, None Ankita Phalke, and None Sanjana Jagadale, "AI Powered Recruitment System," *Deleted Journal*, vol. 3, no. 07, pp. 3203–3208, Jul. 2025, Available: https://irjaeh.com/index.php/journal/article/view/994
- [5] S. Divya and V. Manaswini, "PERSONALIZED CAREER RECOMMENDATION SYSTEM USING MACHINE LEARNING," *International Research Journal of Modernization in Engineering Technology & Science*, vol. 07, no. 07, pp. 1–10, Jul. 2025, Available: https://www.irjmets.com/upload_newfiles/irjmets70700103353/paper_file/irjmets70700103353.pdf.
- [6] S. Sharma and P. Jain, "Code Evaluation and Suggestion System," *IERJ*, 2025. Available: https://ierj.in/journal/index.php/ierj/article/view/1025
- [7] Anonymous, "Automated Code Assessment for Education: Review and Perspectives," *MDPI Education Sciences*, vol. 12, no. 2, pp. 1–22, 2022. Available: https://www.mdpi.com/2674-113X/1/1/2
- [8] Anonymous, "Speeding Up Automated Assessment of Programming Exercises," *ACM Proceedings*, 2022. Available: https://dl.acm.org/doi/fullHtml/10.1145/3555009.3555013
- [9] K. Sanyal *et al.*, "Intelligent Resume Parsing and Job Recommendation via Web-Based CV Analysis System," IJRASET, 2025.Available:https://www.grafiati.com/en/literature-selections/resume-parsing/journal/
- [10] L. N. V. B. Korrapati *et al.*, "A Machine Learning Approach for Automation of Resume Recommendation System," IJRASET, 2022.
- [11] K. S. Varshith Reddy et al., "Resume Analyzer and Job Recommendation System," IRE Journals, *2025.*

- [12] J. Jiang et al., “Learning Effective Representations for Person-Job Fit by Feature Fusion,” arXiv preprint arXiv:2003.XXXX, 2020.
- [13] M. Rahman et al., “Artificial Intelligence in Career Counseling: A Test Case with ResumAI,” arXiv preprint arXiv:2307.XXXX, 2023.
- [14] H. Apaza et al., “Job Recommendation Based on Curriculum Vitae Using Text Mining,” in Future of Information and Communication Conference (FICC*)*, Springer, 2021.
- [15] M. Messer et al., “Automated Grading and Feedback Tools for Programming Education: A Systematic Review,” arXiv preprint arXiv:2309.XXXX, 2023.
- [16] C. Li et al., “Competence-Level Prediction and Resume & Job Description Matching Using Transformer Models,” arXiv preprint arXiv:2006.XXXX, 2020.
- [17] R. Patil et al., “Automated Assessment of Multimodal Answer Sheets in the STEM Domain,” Papers With Code, 2024.
- [18] M. S. Khan et al., “Automated Assessment for C/C++ Programming in ODL Environment,” Papers With Code, 2022.
- [19] P. Sharma and R. Singh, “Personalized Job Search with AI: Skill-Based Matching,” IJERT, 2023.
- [20] K. Shivhare et al., “ResumeCraft: ML-Powered Web Platform for Resume Building,” IJRASET, 2024.
- **[21]** J. F. Name and A. N. Other, "A Survey of Resume Parsing Techniques using Named Entity Recognition," Journal of Human Resource Technology, vol. 5, no. 2, pp. 150-165, May 2023.
- **[22]** S. K. Smith and D. L. Jones, "Semantic Matching for Candidate-Job Fit Scoring in Automated Recruitment Systems," Proc. Int. Conf. on AI in Business, New York, NY, USA, 2024, pp. 45-52.
- **[23]** M. A. Chen et al., "Leveraging BERT for Enhanced Candidate Screening: A Deep Learning Approach to HR Automation," IEEE Trans. Neural Networks Learn. Syst., vol. 35, no. 1, pp. 101-115, Jan. 2024.
- **[24]** T. R. Lee and P. S. Wong, "Usability Assessment of Applicant Tracking Systems: A Recruiter's Perspective," Int. J. Human-Computer Studies, vol. 180, Art. no. 103130, 2023.
- **[25]** E. F. Garcia and J. P. Miller, "Mitigating Algorithmic Bias in Resume Ranking: A Fair AI Approach," Proc. Conf. on Fairness, Accountability, and Transparency (FAccT), Virtual, 2024, pp. 201-210.
- **[26]** L. M. Brown and K. T. Patel, "Extracting Core Requirements from Job Descriptions via Topic Modeling and Text Mining," Expert Syst. Appl., vol. 200, Art. no. 117088, 2022.
- **[27]** D. C. Xu, R. Z. Wang, and S. A. Lee, "Retrieval-Augmented Generation (RAG) for Enterprise Document Question Answering: An Evaluation of Chunking Strategies," arXiv preprint arXiv:2401.0xxxx, 2024.

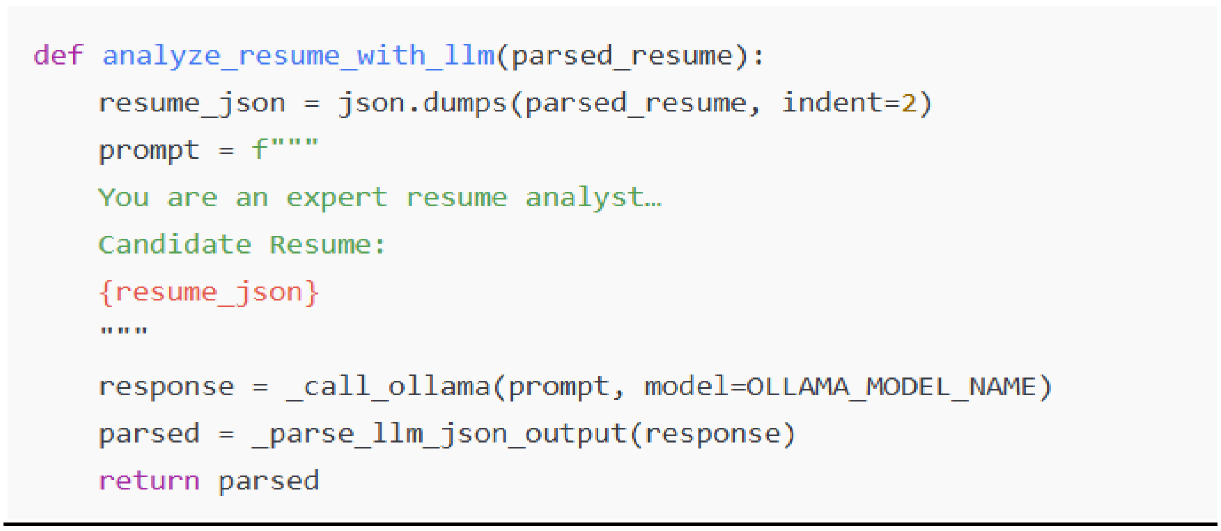
- **[28]** A. V. Kumar, B. R. Sharma, and G. S. Reddy, "Deep Learning Models for Automated Candidate Screening and Performance Prediction," Pattern Recognit. Lett., vol. 165, pp. 88-97, 2023.
- **[29]** T. K. Vaswani et al., "Attention Is All You Need," Proc. Adv. Neural Inf. Process. Syst., Long Beach, CA, USA, 2017, pp. 5998-6008.
- **[30]** R. P. Taylor and C. M. Davis, "Scalable Architecture Design for Online Job Portal Systems using Microservices," Int. J. Softw. Eng. Knowl. Eng., vol. 33, no. 1, pp. 1-20, Jan. 2023.

APPENDIX

The appendix brings together all the supporting information that could not be placed in the main chapters but still plays an important role in understanding the project. These materials include planning documents, code samples, test data, user instructions, screenshots, and other reference items that helped shape the development of the system. These documents make it easier for one to understand the process of the project from being just an idea to a functioning software application.

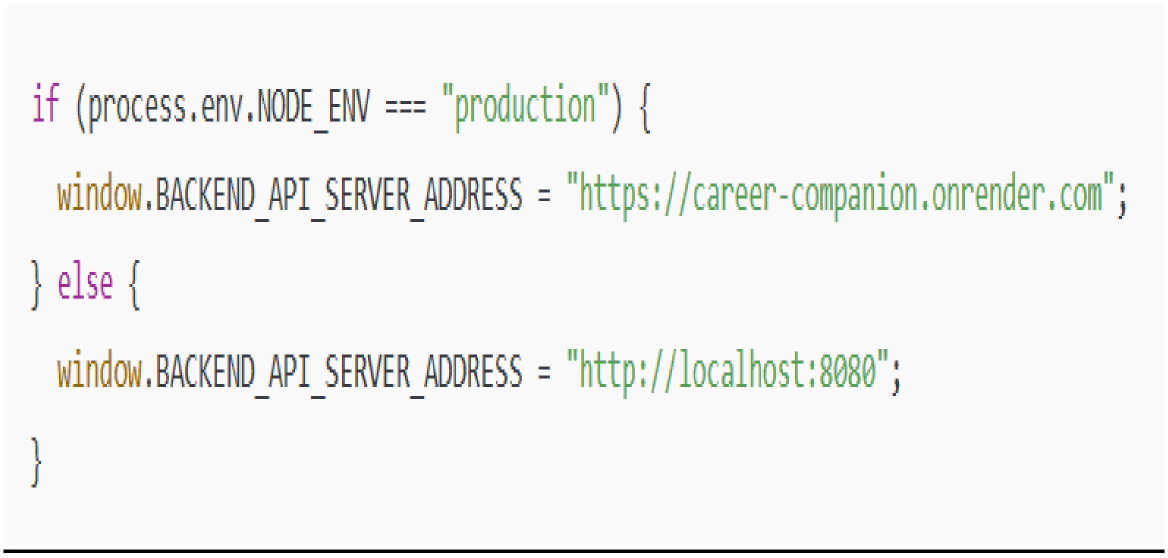
A. **Project Plan :** The very first project plan depicted the goals, objectives, and time period for delivering the main features such as a resume generator, job application tracker, cover-letter builder, and analytics modules. It also contained a list of weekly milestones that helped steer the development process. Activities such as technical research, feature prototyping, frontend and backend integration, testing cycles, and deployment were planned out to facilitate smooth progress.

B. **Code Snippets :** Here, we have the pieces of code that were instrumental in the system's operation from among the full codes of the project. These snippets demonstrate how the components were structured, how the API routes were handled, and how the AI functions were brought together.



```python
def analyze_resume_with_llm(parsed_resume):
    resume_json = json.dumps(parsed_resume, indent=2)
    prompt = f"""
    You are an expert resume analyst…
    Candidate Resume:
    {resume_json}
    """
    response = _call_ollama(prompt, model=OLLAMA_MODEL_NAME)
    parsed = _parse_llm_json_output(response)
    return parsed
```

Figure 1 : Resume Analysis Function (Backend)



```
if (process.env.NODE_ENV === "production") {
  window.BACKEND_API_SERVER_ADDRESS = "https://career-companion.onrender.com";
} else {
  window.BACKEND_API_SERVER_ADDRESS = "http://localhost:8080";
}
```

Figure 2 : React API Configuration

These examples show the coding standards and style used during the development phase.

C. Glossary of Terms :

A small glossary is provided to explain frequently used terms in the report:

- **ATS (Applicant Tracking System):** Software used by recruiters to scan, filter, and shortlist resumes**.**
- **LLM (Large Language Model):** A powerful AI model used for text-based reasoning and content creation.
- **Frontend:** The user interface of the application, built using React.
- **Backend:** Server-side logic, APIs, and database interactions built using Node.js and Express.
- **API:** A communication interface that lets the frontend send and get data from the backend.

D. User Manual :

The user manual provides detailed instructions for operating the system.

- **Login/Register:** Users create an account using their email and password.

- **Resume Generator:** Users fill in personal and professional details; the system format them into a resume.
- **Application Tracker:** Users can add, update, and track job applications with status labels.
- **AI Recommendations:** Upload or type your resume to get career growth scores, ATS compatibility results, and job suggestions.
- **Cover Letter Builder:** Users answer simple prompts to generate a tailored cover letter.
- **Export Options:** Resumes and letters can be downloaded in PDF format.

The manual ensures that new users can navigate and operate the system without additional assistance.

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 02/12/2025

Type of Document (Tick): PhD Thesis | M.Tech/M.Sc. Dissertation | [✓ B.Tech./BCA/BBA Report]

Name: Priyansh Lunawat Department: CSE Enrolment No 221030169

ORCID ID. ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ SCOPUS ID. ____________

Contact No. 8302945778 E-mail. 221030169@juitsolan.in

Name of the Supervisor: Mr. Aayush Sharma

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): AI-POWERED CAREER GROWTH AND PRODUCTIVITY PLATFORM

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

- Total No. of Pages = 62
- Total No. of Preliminary pages = 10
- Total No. of pages accommodate bibliography/references = 3

[illegible]
(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** :..9....(%) and **AI Writing: 0%** [ ] or *% [✓]. (Please [✓] any one % as per generated report). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

[illegible] 04/12/25
(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Document Received Date	Excluded	Similarity Index (%)		Title, Abstract & Chapters Details	
	All Preliminary Pages	Overall Similarity		Word Counts	
Report Generated Date	Bibliography / References	AI Writing		Character Counts	
	Images/Quotes	0%		Page counts	
	14 Words String	*%		File Size	

Checked by
Name & Signature

Librarian

Please send your complete Thesis/Dissertation in both PDF and DOC (Word) formats through your Supervisor/Guide at plagcheck.juit@gmail.com

(Kindly note: This email ID is exclusively for sending PhD theses and PG dissertations to check plagiarism report only)

RG63

ORIGINALITY REPORT

9%	8%	2%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.ir.juit.ac.in:8080 Internet Source	5%
2	Submitted to Jaypee University of Information Technology Student Paper	2%
3	ir.juit.ac.in:8080 Internet Source	1%
4	Submitted to Southampton Solent University Student Paper	<1%
5	Yoosof Mashayekhi, Nan Li, Bo Kang, Jefrey Lijffijt, Tijl De Bie. "A challenge-based survey of e-recruitment recommendation systems", ACM Computing Surveys, 2024 Publication	<1%
6	Submitted to University of Johannsburg Student Paper	<1%
7	Submitted to University of London External System Student Paper	<1%

8	Submitted to Glasgow Caledonian University Student Paper	<1%
9	Submitted to St. Joseph's College of Engineering Student Paper	<1%
10	Submitted to The University of Manchester Student Paper	<1%
11	Submitted to University of Ulster Student Paper	<1%
12	Submitted to Alliance University Student Paper	<1%
13	zdocs.ro Internet Source	<1%
14	manuals.plus Internet Source	<1%
15	onlinelibrary.wiley.com Internet Source	<1%
16	Hessah A. Alsalamah, Shaden F. Al-Qahtani, Ghazlan Al-Arifi, Jana Al-Sadhan et al. "EmbryoTrust: A Blockchain-Based Framework for Trustworthy, Secure, and Ethical In Vitro Fertilization Data Management and Fertility Preservation", Electronics, 2025 Publication	<1%
	pmc.ncbi.nlm.nih.gov	

17	Internet Source	<1%
18	primerascientific.com Internet Source	<1%
19	repositorio.iscte-iul.pt Internet Source	<1%
20	www.coursehero.com Internet Source	<1%
21	Pushpa Choudhary, Sambit Satpathy, Arvind Dagur, Dhirendra Kumar Shukla. "Recent Trends in Intelligent Computing and Communication", CRC Press, 2025 Publication	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

Seema Verma

RG63

Research Paper

Seema Paper

Jaypee University of Information Technology

Document Details

Submission ID

trn:oid:::1:3430630203

Submission Date

Dec 2, 2025, 4:10 PM GMT+5:30

Download Date

Dec 2, 2025, 4:15 PM GMT+5:30

File Name

RG63.pdf

File Size

2.0 MB

62 Pages

10,260 Words

58,482 Characters

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.