Lebanese University

Faculty of Engineering III

Electrical and Electronic Department

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Resuma

Mini-Project

by

**Rasha Hammoud, 6373**

**Faten Mortada, 6506**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Spring 2024-2025**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Supervisor: Dr. Mohamad AOUDE**

Table of Contents

[Introduction 1](#_Toc203673499)

[System Architecture 1](#_Toc203673500)

[1. Overview Diagram 1](#_Toc203673501)

[2. Technologies Used 2](#_Toc203673502)

[3. Data Flow 2](#_Toc203673503)

[AI Logic & Dual-Agent Implementation 3](#_Toc203673504)

[1. Agent Design Philosophy 3](#_Toc203673505)

[2. Agent Overview 3](#_Toc203673506)

[Testing & Evaluation Strategy 4](#_Toc203673507)

[1. Unit & Integration Testing 4](#_Toc203673508)

[2. LLM Output Validation 5](#_Toc203673509)

[Results and output samples 5](#_Toc203673510)

[1. Generated Output Files 5](#_Toc203673511)

[2. Resume Before and After 6](#_Toc203673512)

[3. Database Structure 6](#_Toc203673513)

[4. Frontend Interface 7](#_Toc203673514)

[Conclusion & Future Work 9](#_Toc203673515)

Introduction

In today’s competitive job market, tailoring resumes and cover letters to specific job descriptions is critical for applicants hoping to stand out. However, this task is time-consuming, especially for candidates applying to multiple roles. Recruiters also face challenges in quickly evaluating whether resumes align with job requirements.

To address these challenges, we developed Resuma — an AI-powered system that automates resume evaluation, enhancement, and tailored cover letter generation. Built with modular AI agents and accessible via a web interface, Resuma aims to improve job seekers' chances of landing interviews while streamlining the recruiter review process.

This report outlines the technical architecture, dual AI implementation strategy, ethical considerations, performance evaluations, and deployment strategy of our system.

System Architecture

The Resuma Agents platform follows a modular microservice-style architecture designed for scalability, reusability, and clean separation of concerns. The system is divided into three primary layers: the frontend interface, the FastAPI-based backend, and a collection of AI-powered agents that handle resume processing tasks.

1. Overview Diagram

The system consists of the following major components:

- **Frontend Web Application:** A React-based UI that allows users to upload their resume (PDF) and a job description. It provides live feedback, progress indicators, and displays the final outputs: improved resume (LaTeX), tailored cover letter (TXT), and evaluation insights (JSON).

- **Backend API Server (FastAPI):** This service handles incoming HTTP requests, validates inputs, invokes AI agents, manages file uploads, and returns the final artifacts to the user.

- **AI Agents:**  
 • Resume Parser Agent  
 • Evaluation Agent  
 • Editor Agent  
 • Cover Letter Agent

- **Template Renderer:** A Jinja2-based module that renders the final resume in LaTeX format using a predefined template and the updated JSON content.

- **Dockerized Environment:** All services and dependencies, including Python, Node.js, LaTeX, and supporting libraries, are encapsulated in a single Docker container for reproducibility and portability.

1. Technologies Used

* Frontend: React.js, Tailwind CSS, Axios, FileReader API
* Backend: FastAPI, Python 3.10
* LLM Integration: OpenAI GPT-4 via API calls
* Document Parsing: PyMuPDF (fitz), pdfplumber
* Templating Engine: Jinja2 + LaTeX
* Deployment: Docker

1. Data Flow

* User uploads a resume (PDF) and pastes a job description.
* The backend extracts structured JSON from the resume using the parser agent.
* The evaluation agent compares the resume and job description.
* If improvement is needed, the editor agent updates the resume.
* The renderer module produces a LaTeX version of the updated resume.
* The cover letter agent generates a personalized letter.
* All outputs are returned to the user and optionally saved.

AI Logic & Dual-Agent Implementation

The core intelligence of the Resuma Agents platform is built on a pipeline of specialized LLM-powered agents, each responsible for a discrete subtask in the resume tailoring process. Rather than relying on a monolithic prompt or one-size-fits-all model, our design uses a dual-agent strategy to separate evaluation from editing, improving explainability and modularity.

1. Agent Design Philosophy

Each agent follows a single-responsibility principle and is stateless, receiving all required inputs and returning structured outputs. This allows each step to be debugged, audited, and improved independently.

1. Agent Overview

The following agents are implemented:

1. **Resume Parser Agent**

* Task: Converts a raw resume PDF into structured JSON (name, work experience, skills, education, etc.)
* Method: Prompts a large language model (e.g. GPT-4) using extracted text and predefined schemas to return a JSON object compliant with the ResumeSchema standard.
* Output: JSON

1. **Evaluation Agent** (Agent 1 of the Dual System)

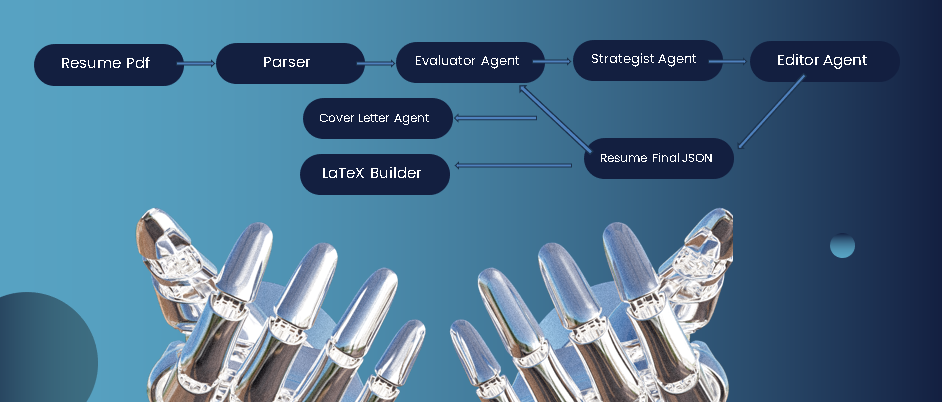
* Task: Analyzes how well the resume fits the job description.
* Input: Resume JSON and raw job description text.
* Output: JSON object containing skill gaps, matching sections, and recommendations.

1. **Editor Agent** (Agent 2 of the Dual System)

* Task: Takes the original resume and the evaluation JSON to produce an updated version.
* Logic: Uses structured evaluation feedback to selectively rewrite or augment parts of the resume while preserving overall tone and format.
* Output: Updated resume in JSON format, suitable for LaTeX rendering.

1. **Cover Letter Agent**

* Task: Generates a customized cover letter aligning with the job description and resume content.
* Output: Plain text (TXT) with sections: greeting, body, closing.



Testing & Evaluation Strategy

To ensure the correctness, reliability, and effectiveness of the Resuma Agents platform, a comprehensive testing and evaluation strategy was implemented. This included both traditional software validation (unit tests, integration tests) and qualitative evaluation of AI-generated content.

1. Unit & Integration Testing
2. API Testing:

* All FastAPI routes were tested using pytest and HTTP clients like HTTPX.
* Tests validated successful uploads, JSON generation, and response status codes (200, 422, 500).

1. File Handling Tests:

* Input PDF files were validated for proper parsing.
* Output paths (JSON, .tex, .txt) were tested for creation and formatting integrity.

1. Docker Tests:

• Containerized builds were verified with uvicorn startup and all pre-installed dependencies.

1. LLM Output Validation

Each LLM-powered agent was evaluated based on:

* JSON Schema Compliance:  
  - All agent responses (especially from the parser and editor) were validated using Pydantic models.  
  - Invalid or malformed responses were caught and logged during the pipeline.
* Error Injection Testing:  
  -Simulated edge cases (e.g., missing fields in resume, vague job descriptions) were used to evaluate robustness.

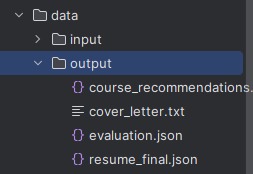
Results and output samples

1. Generated Output Files

The system produces several structured output files after processing:

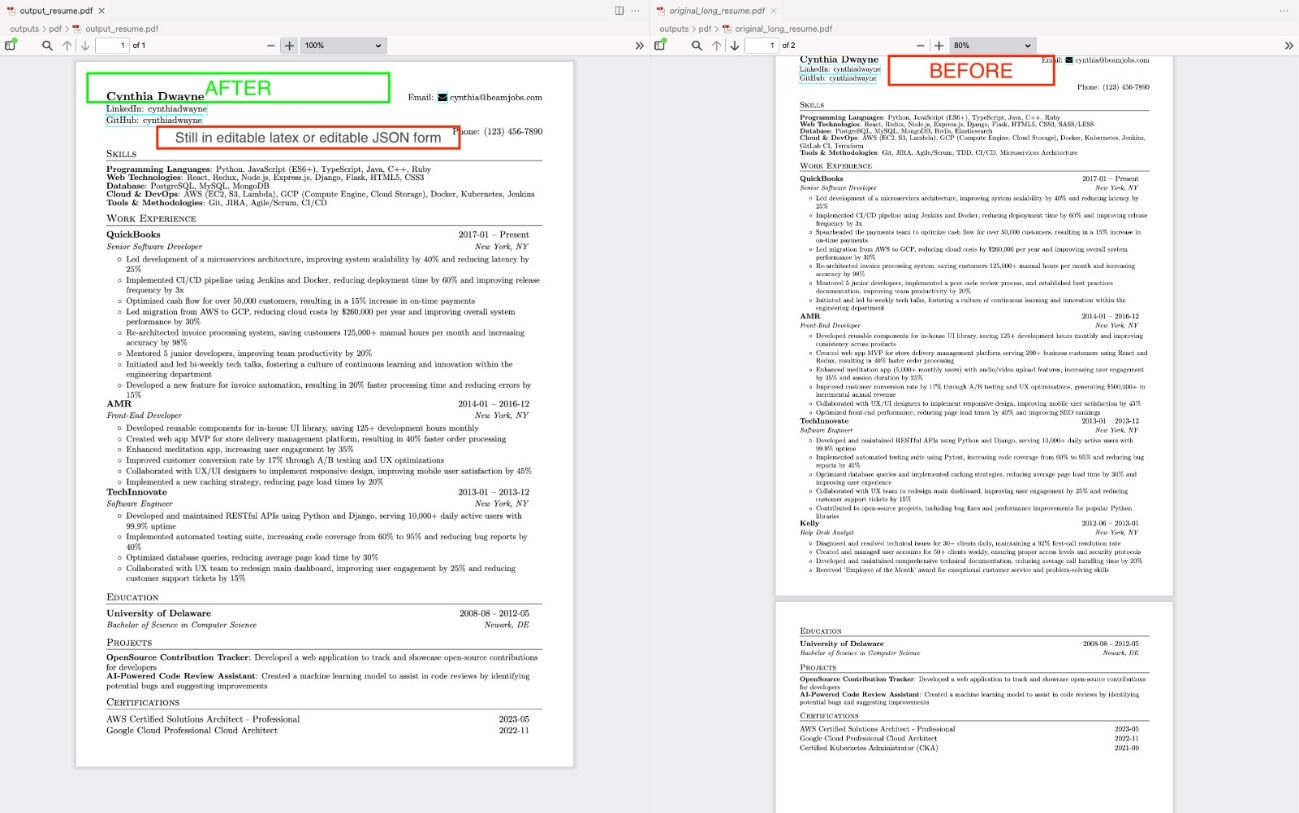
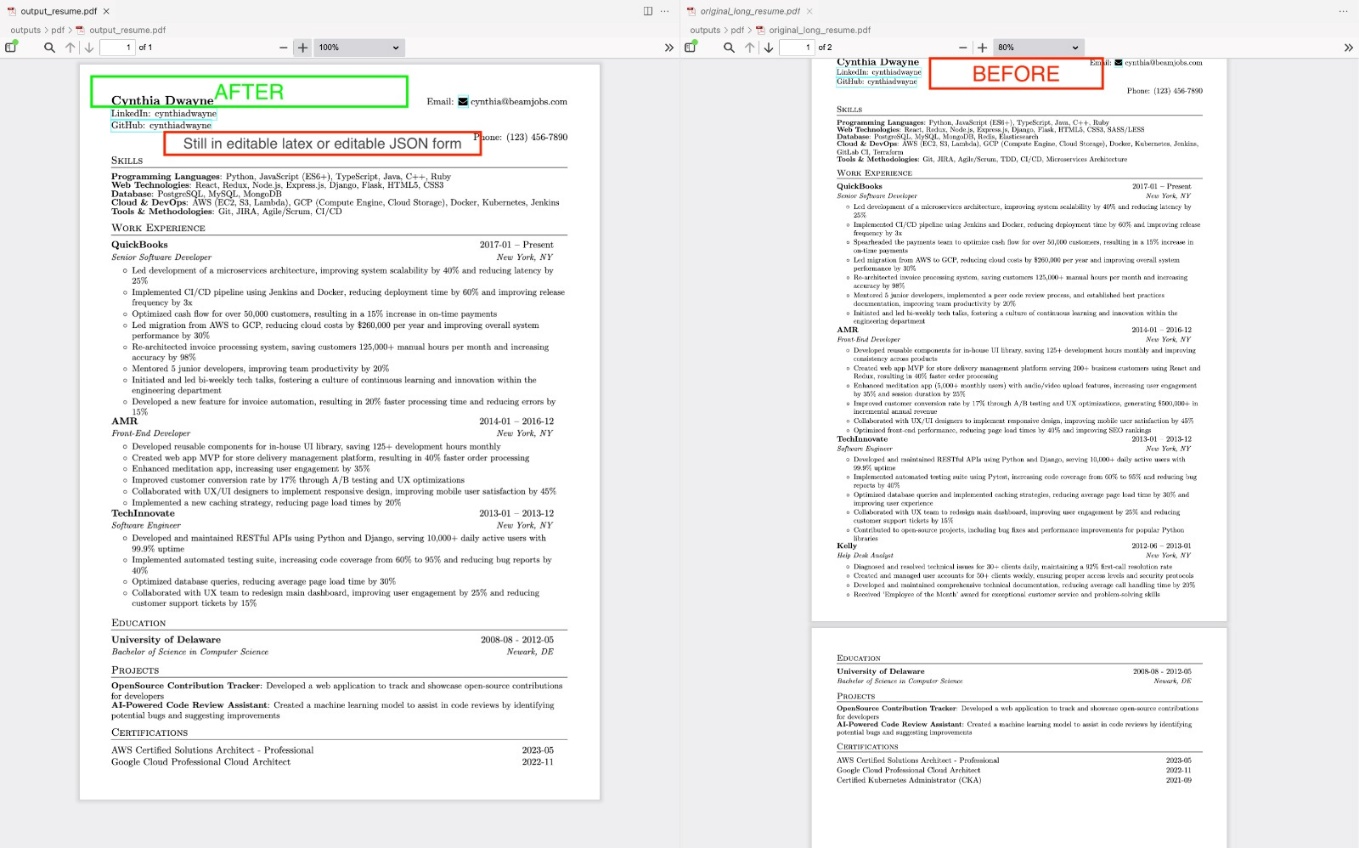
* Optimized Resume (.tex and .pdf)
* Cover Letter (.txt and .pdf)
* Resume Evaluation Report (.json)
* Parsed Resume Data (.json)

These files are available for download directly from the frontend after a successful run.



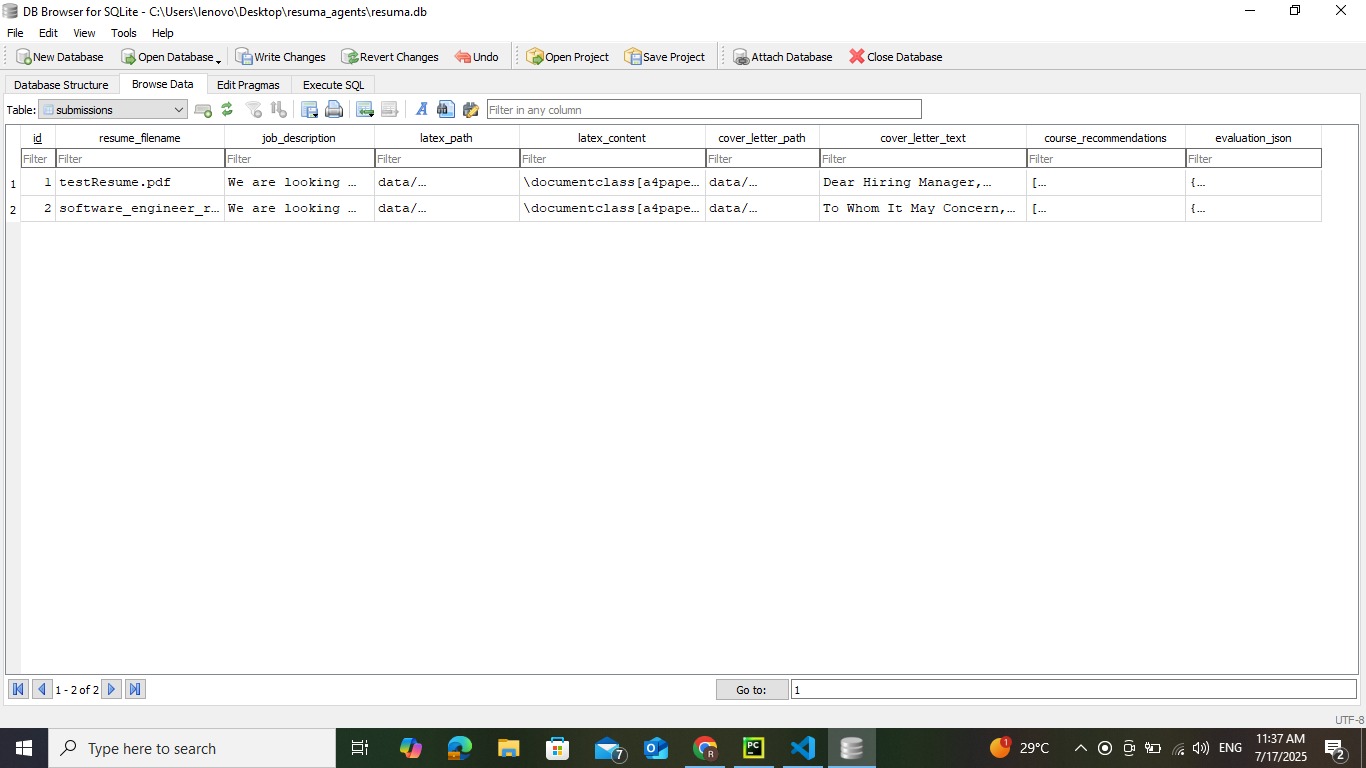
1. Resume Before and After

The system demonstrates a clear transformation between the original uploaded resume and the AI-enhanced version. The initial resume, often unstructured or lacking key job-targeted keywords, is restructured and improved with better formatting, clearer sectioning, and content tailored to the provided job description.

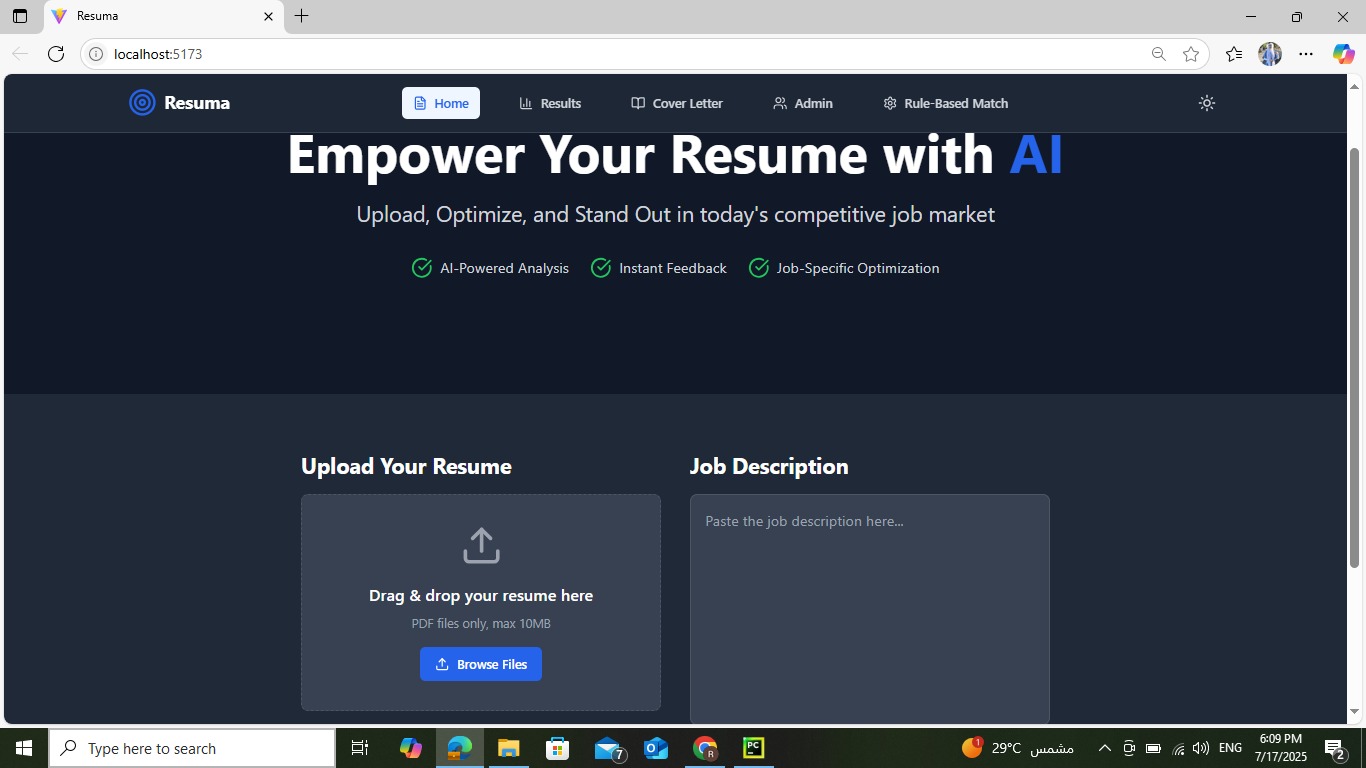
1. Database Structure

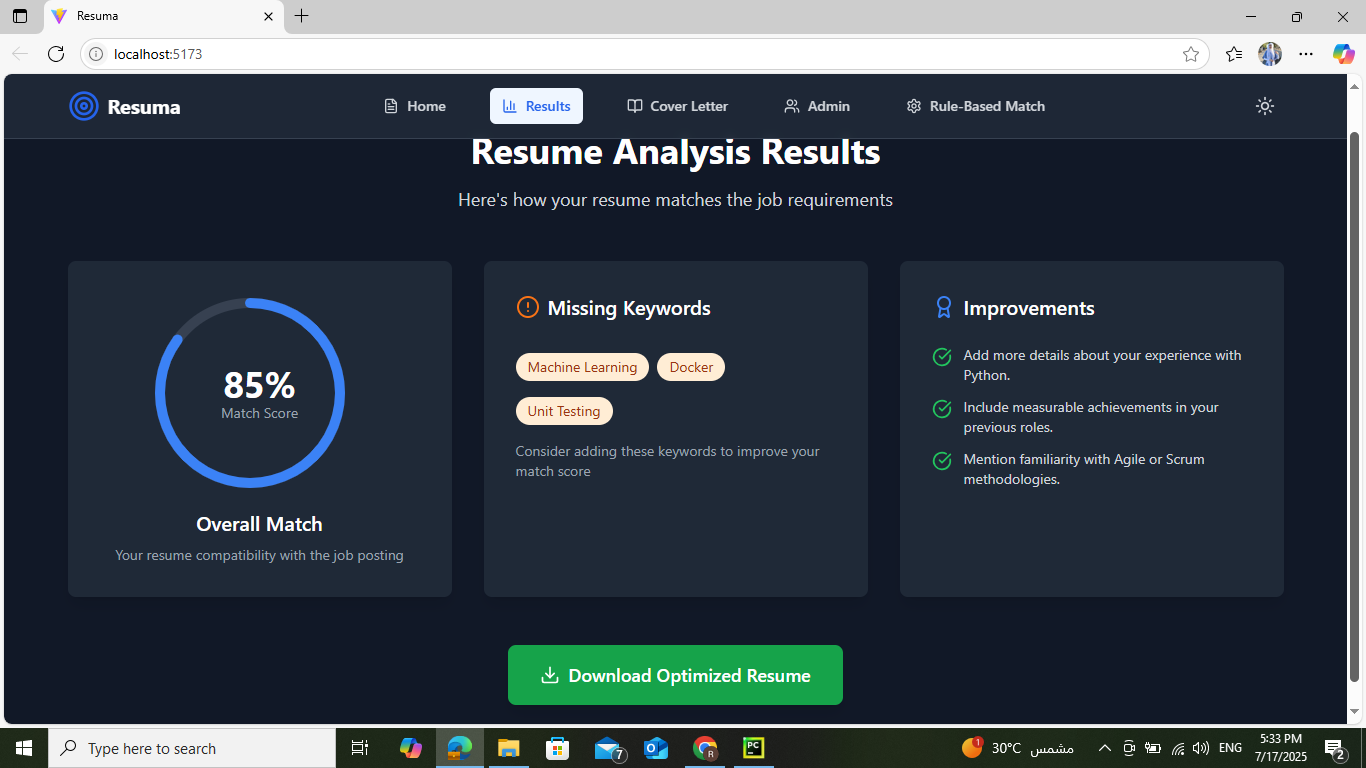
All parsed data, evaluations, and resume versions are stored in a local SQLite database. This includes JSON outputs from the parsing step, resume enhancement records, and job description data. The database provides traceability and ensures data persistence between sessions and components.

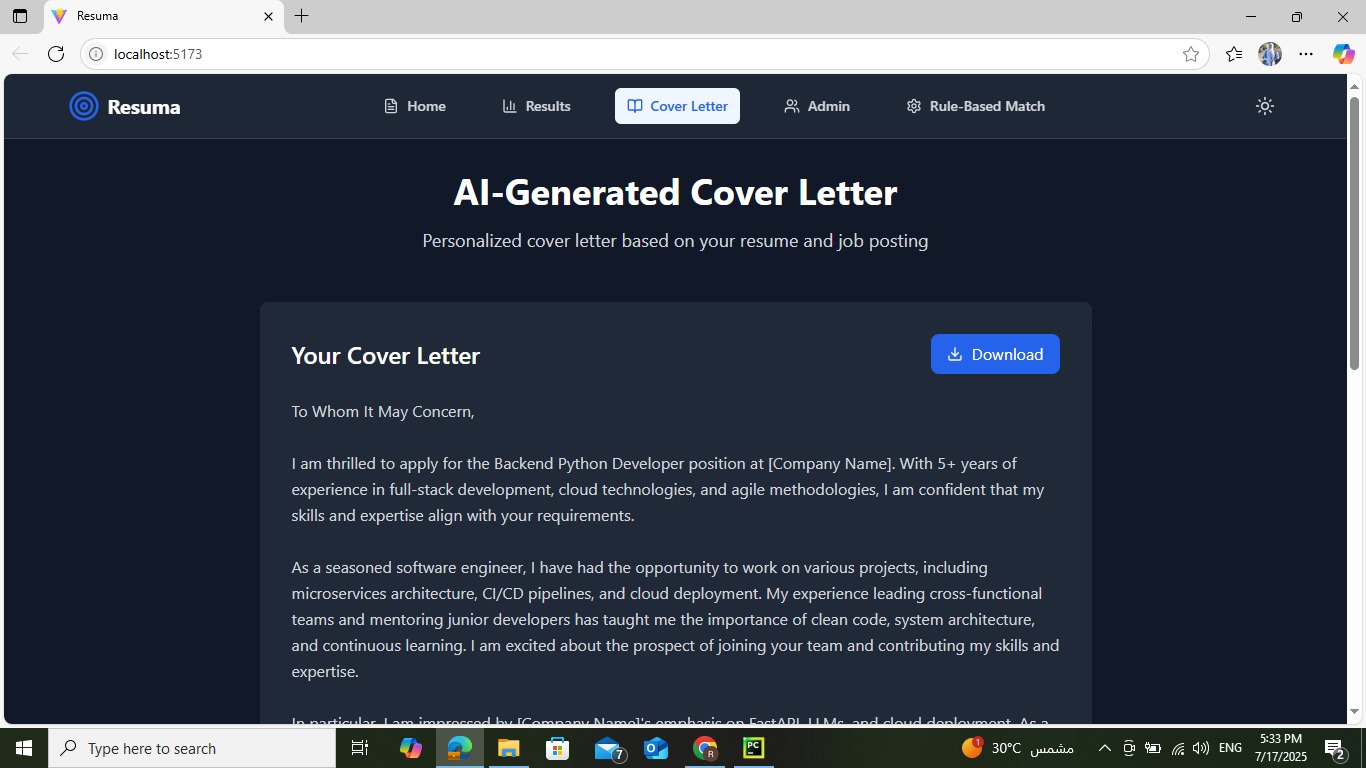


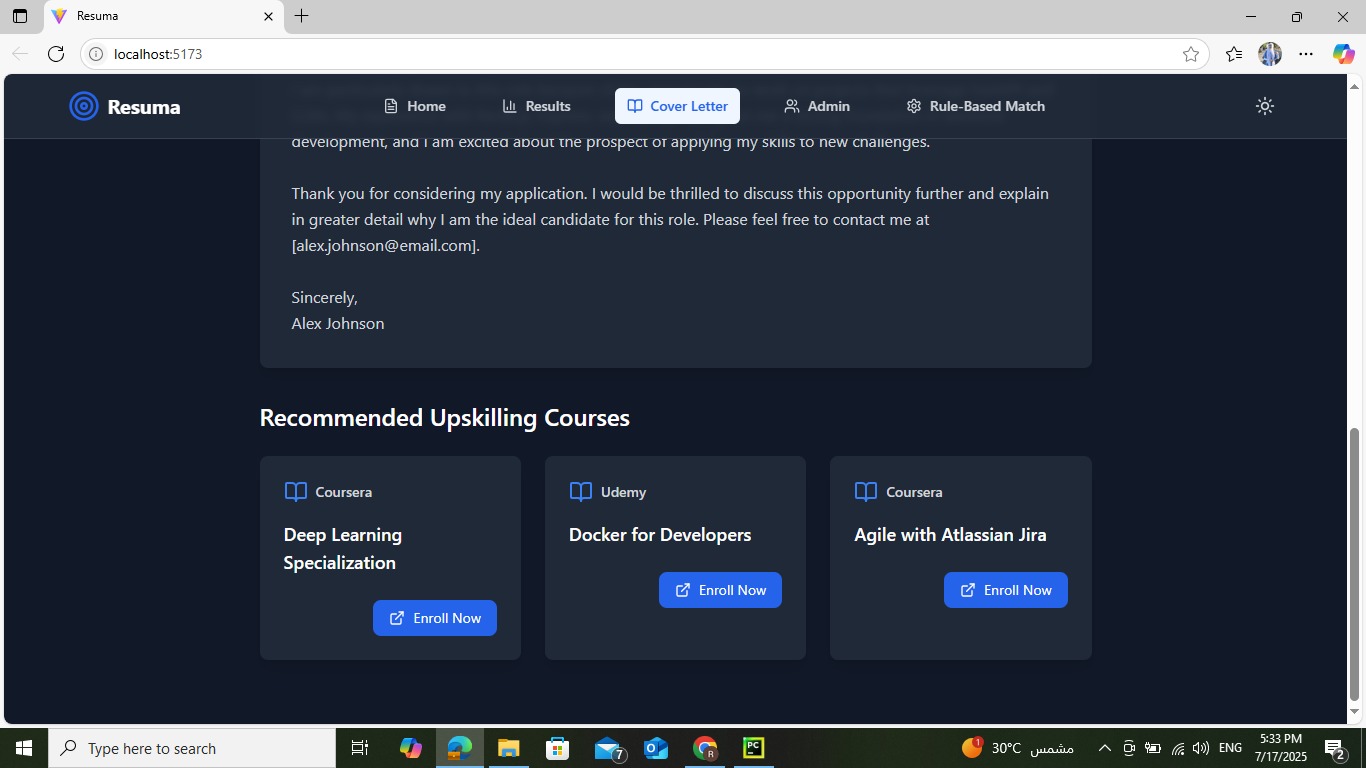
1. Frontend Interface

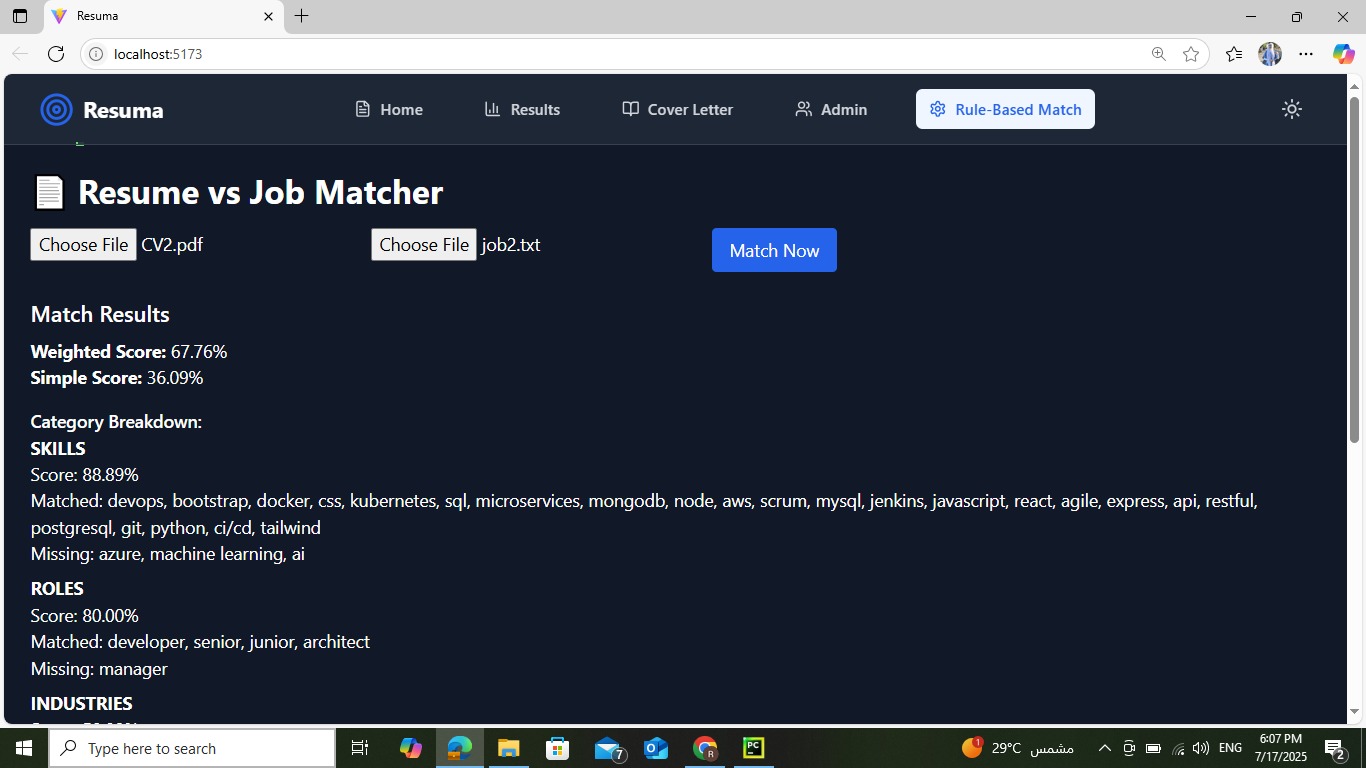
A clean and intuitive React-based web interface allows users to upload resumes, input job descriptions, view parsed data, download the final resume, and review evaluation metrics. The frontend enhances usability and acts as the bridge between the user and the backend AI agents.











Conclusion & Future Work

This project introduced an AI-powered resume optimization system that integrates natural language processing and generative models to assist users in tailoring their resumes and cover letters to specific job descriptions. By implementing a dual-agent architecture—one for intelligent resume parsing and evaluation, and another for content generation—the system ensures both semantic relevance and high-quality output. Through a modern web interface and full backend automation, users can easily upload resumes, view evaluations, and download optimized documents with minimal friction.

Future improvements could focus on expanding support for multi-page resumes, multilingual parsing and generation, and deeper integration with job platforms like LinkedIn or Indeed. Additionally, incorporating feedback loops from recruiters or HR tools could further refine the optimization pipeline. Expanding the training dataset and fine-tuning models on real-world hiring data would also enhance accuracy and personalization.