LTIMindtree

2/07/2024

INTERNSHIP PROJECT REPORT

RESUME PARSING USING GEN AI

PREPARED BY :
Aishik Ghosh

Abstract:

This project focuses on developing a web application that helps employee recruiters and HRs with the process of parsing resumes, extracting key information, and displaying it in a structured format. Utilizing Python Flask for the backend, HTML and CSS for the frontend, SQLAlchemy for database management, and Gemini GenAI for enhanced data extraction, the simple web app offers a user-friendly interface for uploading resumes and viewing parsed data. Key functionalities include the extraction of name, contact, email, skills and experience of individuals from various resume formats. The project addresses challenges such as format variability and data inconsistency, providing solutions and suggesting future enhancements like user authentication, support for multiple file formats, advanced UI/UX improvements, and performance optimization. This system demonstrates the effective integration of GenAI in resume processing, offering a scalable solution for managing resume data efficiently.

Introduction:

In today's fast-paced and competitive job market, efficient management of resumes is crucial for both job seekers and employers. This project addresses the need for an dedicated system to parse and display resume information, streamlining the hiring process. The main goal of this project was to create an application that utilizes Gemini Gen AI to extract critical information from resumes. The tool is intended to help recruiters save time and ensure no potential candidate is missed due to manual errors.

Companies and recruiters typically rely on the Named Entity Recognition (NER) functionality of the Spacy library for this task. However, this approach encounters challenges in accuracy and handling various resume formats. To address these issues, this project implements a solution using the Gemini GenAI API, which offers higher accuracy in extracting critical information, better adaptability to different resume structures, and reduced need for manual maintenance.

By leveraging the capabilities of Python Flask for backend development, HTML and CSS for frontend design, SQLAlchemy for database management, and Gemini GenAI for intelligent data extraction, the application aims to provide a seamless solution for handling resume data. Users can easily upload resumes, which are then processed to extract key details such as contact information, education, experience, and skills. The extracted information is stored in a database and presented in a clear, structured format, making it accessible and easy to review. This project not only enhances efficiency but also demonstrates the integration of advanced AI technologies in solving real-world problems.

Methodology:

The methodology for developing this resume parsing and display application focuses on backend development, frontend design, database management, and GenAI integration. The following sections outline the detailed approach taken to achieve the project's objectives.

Backend Development (Python Flask):

The backend of the application is built using Python Flask, handling server-side logic and routes. Key routes include:

1. /upload: Accepts resume uploads, processes the files, and extracts information.
2. /result: Displays a list of parsed resumes with options to add or clear entries.
3. /view-data: Shows all stored data from the database.
4. /details/<id>: Displays detailed information for a specific resume.

The application is configured with an SQLite database for storing parsed resume data. The backend leverages various libraries for processing different file formats and extracting text content:

1. pdfminer for PDFs
2. docx2txt for DOCX files
3. Python's built in .read() for TXT files

P.T.O.

The backend also has the integration of the Gemini GenAI API for the parsing procedure which is discussed later in this section.

Frontend Design (HTML, CSS)

The frontend of the application is designed with basic HTML and CSS to provide a user-friendly interface. Key pages include:

- Upload Page (upload.html): A form for users to upload resumes.
- Results Page (result.html): Displays a table of parsed resumes with options to view details, add more resumes, or clear entries.
- View Data Page (view.html): Shows a list of all resumes stored in the database.
- Details Page (details.html): Shows detailed information for a specific resume.

CSS is used to style the pages and ensure a clean, organized layout.

Database Management (SQLAlchemy)

SQLAlchemy is used to manage the application's database operations. An SQLite database is used to store the extracted resume information. The 'ResumeData' model defines the structure of the stored data, which includes fields such as name, email, phone, IT skills, programming, front-end, back-end, database skills, AI/ML skills, other skills, and experience.

AI Integration (Gemini Gen AI)

Gemini Gen AI is integrated in the backend to handle the parsing and extraction of information from resumes. The AI model is configured and called via an API, processing uploaded resumes and returning key information in JSON format. This approach offers several advantages over traditional methods, including higher accuracy, better adaptability to different resume formats, and reduced need for manual maintenance.
The following steps outline the extraction process:

1. First the text content from the uploaded resume is extracted using appropriate libraries.
2. Next the extracted text is passed to the Gemini Gen AI API with a properly engineered prompt to ensure the result is in JSON format.
3. The JSON text is received and processed to extract relevant resume details.
4. The extracted data is stored in the SQLite database.

Error Handling and Validation

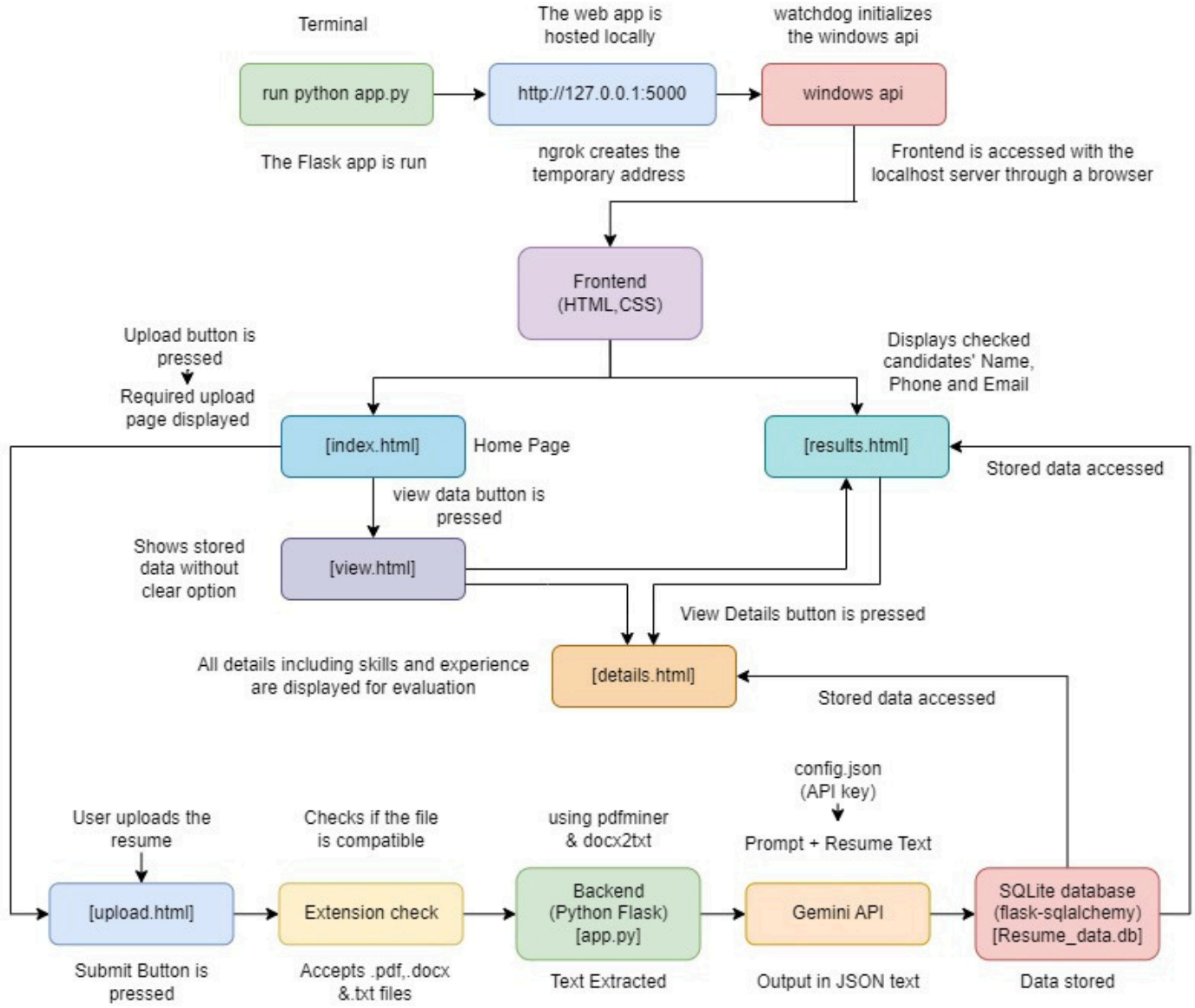
Robust error handling and validation mechanisms are implemented to manage various potential issues, such as file format inconsistencies and API response errors. This ensures that the application can handle a wide range of resume formats and data representations effectively.

Problems Faced

1. Resume Format Variability: Resumes come in various formats and layouts, making it challenging to accurately parse and extract information.
2. Data Inconsistency: Inconsistent data representation within resumes (e.g., different formats for dates, job titles) led to difficulties in standardizing the extracted information.
3. Prompt Engineering: Crafting effective prompts for the Gemini Gen AI API posed a significant challenge. Designing prompts that consistently yield accurate and comprehensive results from diverse resume formats required iterative refinement and testing. Adjusting prompts to accommodate variations in language, structure, and content within resumes was essential to enhance the AI model's performance and ensure reliable data extraction.
4. API Limitations: Integrating with the Gemini Gen AI API had limitations in terms of request rate and response time, affecting the overall performance.
5. Error Handling: Implementing robust error handling for file uploads, API responses, and database operations required significant effort.

P.T.O.

Working of the application:



P.T.O.

Future Enhancements

This project is just a prototype and can be improved further in future to enhance the efficiency and security.

1. User Authentication and Authorization: User authentication can be implemented to secure access to resume data and provide personalized experiences.
2. Enhanced AI Model: The AI model can be enhanced to allow for the extraction of additional information such as certifications, languages, and projects, and to handle more complex resume formats.
3. Advanced UI/UX Improvements: The user interface can be enhanced with modern frontend frameworks like React or Vue.js to provide a more interactive and dynamic user experience.
4. Data Visualization: Data visualization tools can be integrated to provide insights and analytics on the parsed resume data.
5. Performance Optimization: The backend can be optimized for better performance, including the implementation of caching mechanisms and asynchronous processing of resume parsing.
6. Multilingual Support: The parsing capabilities can be extended to handle resumes in multiple languages, thereby broadening the application's usability.

Conclusion

This project exemplifies the synergy between Python Flask, HTML, CSS, SQLAlchemy, and Gemini Gen AI to create a robust web application. Python Flask facilitates the backend logic, handling file uploads and interfacing with SQLAlchemy for efficient database management. HTML and CSS contribute to a sleek and intuitive user interface, enhancing user experience and accessibility.

Gemini Gen AI elevates the application by seamlessly parsing resume data, offering superior accuracy and adaptability to diverse resume formats. The integration of AI streamlines data extraction, reducing manual effort and ensuring consistent results. Overall, this application underscores the transformative impact of AI in optimizing data-driven workflows, making complex tasks like resume parsing more efficient and effective.