**SMART RECRUIT**

**This code is a Streamlit web application named "SmartRecruit" designed to assist with resume review using Google's Generative AI model (GEMINI).**

This project is a web application called "SmartRecruit" that helps analyze resumes. Imagine you're hiring for a job, and you receive many resumes. It can be time-consuming to go through each one. This app makes it easier by using a special AI (artificial intelligence) to analyze the resumes.

**Here's how it works:**

You provide a job description, like what skills you're looking for.

You upload the resumes you want to analyze (in PDF format).

The app uses Google's AI technology to read the resumes and understand how well they match the job description.

It then gives you insights, like strengths and weaknesses of each candidate, skills they could improve, missing keywords in their resumes, and how well they match the job.

Overall, it's like having a smart assistant to help you quickly review and understand resumes, so you can make better hiring decisions.

**1. \*\*Imports and Setup\*\*:**

- Various Python libraries are imported including Streamlit, PIL (Python Imaging Library), io, pdf2image, base64, and google.generativeai (presumably a custom module for accessing Google's Generative AI).

- The environment variables are loaded using `dotenv`.

- The Google API key is retrieved from the environment variables and used to configure the GEMINI AI model.

**2. \*\*Streamlit App Configuration\*\*:**

- The Streamlit page configuration is set with a custom page title.

- Custom CSS styles are defined to style the appearance of the web application. It includes setting a background image and customizing the font.

**3. \*\*Functions\*\*:**

- `get\_gemini\_response(input, pdf\_content, prompt)`: This function takes input text, PDF content, and a prompt as arguments. It utilizes the GEMINI AI model to generate a response based on the provided input, PDF content, and prompt.

- `input\_pdf\_setup(uploaded\_file)`: This function processes the uploaded PDF file. It converts the PDF to images, takes the first page, converts it to bytes, and encodes it to base64 format.

**4. \*\*Streamlit App\*\*:**

- The main Streamlit application starts with a header and subheader introducing the purpose of the application.

- It provides a text input for the job description and a file uploader for uploading resumes in PDF format.

- Buttons are provided for different actions such as analyzing the resume, suggesting skills improvement, identifying missing keywords, and calculating the percentage match.

- Each button click triggers a specific action:

- It checks if a PDF file is uploaded and then calls `input\_pdf\_setup` to process the PDF content.

- It then uses the `get\_gemini\_response` function to generate a response based on the selected action (prompt) and the provided job description.

- The response is displayed in a subheader.

**5. \*\*Conclusion\*\*:**

- A horizontal line and a caption are added at the end of the application.

Overall, this code represents a basic Streamlit web application that utilizes Google's Generative AI (GEMINI) model to analyze resumes based on a provided job description. It allows users to upload resumes, input a job description, and perform various analyses on the resumes using the GEMINI model, presenting the results within the web interface.

**code step by step explanation :**

**1. \*\*Imports\*\*:**

- The code begins by importing necessary modules:

- `dotenv` for loading environment variables.

- `streamlit` for creating the web application.

- `os` for interacting with the operating system.

- `PIL.Image` from the Python Imaging Library (PIL) for image processing.

- `io` for handling input/output operations.

- `pdf2image` for converting PDF files to images.

- `base64` for encoding and decoding base64 data.

- `google.generativeai` for accessing Google's Generative AI (GEMINI).

**2. \*\*Environment Variables\*\*:**

- `load\_dotenv()` is used to load environment variables from a `.env` file into the script's environment.

- The Google API key is retrieved from the environment variables using `os.getenv("GOOGLE\_API\_KEY")`.

**3. \*\*GEMINI AI Configuration\*\*:**

- The GEMINI AI model is configured with the obtained Google API key using `genai.configure()`.

**4. \*\*Streamlit App Configuration\*\*:**

- `st.set\_page\_config()` is used to configure the Streamlit web application, setting the page title to "SmartRecruit".

**5. \*\*Custom CSS Styles\*\*:**

- Custom CSS styles are defined to style the appearance of the web application.

- This includes setting a background image and customizing the font.

**6. \*\*Functions\*\*:**

- Two main functions are defined:

- `get\_gemini\_response(input, pdf\_content, prompt)`: This function utilizes the GEMINI AI model to generate a response based on input text, PDF content, and a prompt.

- `input\_pdf\_setup(uploaded\_file)`: This function processes the uploaded PDF file, converting it to images and encoding the first page as base64 data.

**7. \*\*Streamlit App\*\*:**

- The main Streamlit application starts with a header and subheader introducing the purpose of the application.

- It provides a text input for the job description and a file uploader for uploading resumes in PDF format.

- Buttons are created for different actions such as analyzing the resume, suggesting skills improvement, identifying missing keywords, and calculating the percentage match.

- Each button click triggers a specific action:

- It checks if a PDF file is uploaded.

- It then calls the `input\_pdf\_setup` function to process the PDF content.

- Finally, it uses the `get\_gemini\_response` function to generate a response based on the selected action (prompt) and the provided job description, displaying the response in a subheader.

**8. \*\*Conclusion\*\*:**

- A horizontal line and a caption are added at the end of the application.

This code essentially creates a Streamlit web application called "SmartRecruit" that allows users to upload resumes, input a job description, and analyze the resumes using Google's Generative AI (GEMINI) model. The application provides various features for resume analysis and presents the results within the web interface.