Medical AI Assistant App

Detailed Description of "app.py":

A Flask-based Medical AI Web App

"app.py" defines a Flask web application that utilizes Google's Gemini API to provide various medical AI functionalities. Here's a breakdown of the code, along with instructions on how to run it:

I. Functionalities:

The app offers five main features:

1. **Diagnosis:** Users can input their symptoms or a specific disease name to receive insights about potential medical conditions, possible causes, suggested diagnostic tests, common medications, workout regimens, dietary advice, and precautions.
2. **Medicine Information:** Users can search for information about specific medicines, either by generic or brand name. The app provides details about indications, pharmacology, dosage, interactions, side effects, pregnancy & lactation considerations, precautions, overdose effects, storage, and alternate names.
3. **Image Analysis:** Users can upload medical images (like X-rays or skin images) to receive AI-powered analysis. The app offers various analysis types, including X-ray description, skin cancer detection, tumor detection, pregnancy detection, and custom prompts for specialized analysis.
4. **Chatbot:** Users can interact with a medical chatbot powered by Gemini. The chatbot allows specifying a medical topic and inputting multiple molecules with their properties, making it suitable for more focused medical inquiries.
5. **History:** This section displays logs of past user interactions with the diagnosis, medicine information, image analysis, and chatbot features. Users can also download individual interaction logs for offline reference.

II. Running the Application:

1. Prerequisites:

- **Python 3.7 or later:** Make sure you have a compatible Python version installed on your system.
- **pip:** The package installer for Python should be installed with your Python installation.

2. Installing Dependencies:

- Open terminal and install below packages using “sudo”:
- sudo apt update
- sudo apt install python3.11
- sudo apt install python3-pip
- sudo apt install python3-flask

- Navigate to the directory where "app.py" is located in your terminal.
- Execute the following command to install necessary libraries:

  ```
  pip install -r requirements.txt
  ```

- **requirements.txt:** This file likely contains the following packages:

  ```
  Flask
  google-generativeai
  python-dotenv
  Pillow
  Werkzeug
  Flask-WTF
  ```

3. Setting up the Environment:

- **API Key:**
  - Obtain a Google API key for accessing the Gemini API.
  - Create a file named ".env" in the same directory as "app.py."
  - In the ".env" file, add the following line, replacing "YOUR_API_KEY" with your actual API key:

    ```
    GOOGLE_API_KEY="YOUR_API_KEY"
    ```

4. Running the App:

- From your terminal, execute the following command:

  ```
  flask run
  ```

- The app will start a local development server, typically accessible at

  http://127.0.0.1:5000/

  in your web browser.

III. Code Structure & Key Components:

- **Imports:** Imports necessary modules for web framework (Flask), AI API (Google Gemini), environment variables loading (dotenv), image processing (PIL), security (Werkzeug), and database interaction (sqlite3).
- **API Setup:** Configures the Gemini API using the provided API key and specifies the Gemini model to use.
- **Directories:** Creates directories for storing logs of user interactions with different app functionalities.
- **User Database (SQLite):** Sets up a simple SQLite database to store user credentials (username and hashed passwords) for basic authentication.
- **Utility Functions:**

    - get_gemini_response(): Sends prompts to the Gemini API and retrieves the generated text response.
    - save_interaction(): Saves user input and the corresponding Gemini response to a text file in the appropriate directory.
    - read_interactions_from_file(): Reads interaction log files from a given directory and returns the content for display on the history page.
- **Function-Specific Logic:** Contains functions for:
    - Obtaining user input from web forms.
    - Constructing specific prompts for diagnosis and medicine information requests.
    - Processing uploaded images and generating prompts based on the chosen analysis type.
    - Handling user interactions with the chatbot.
- **Routes:** Defines the different URL endpoints of the application and their associated functions.
    - /: The homepage.
    - /signup: User registration page.
    - /login: User login page.
    - /logout: Logs the user out.
    - /diagnosis: Handles user interactions with the diagnosis functionality.
    - /medicine: Handles user interactions with the medicine information retrieval.
    - /image: Handles image uploads and analysis.
    - /chat: Manages the chatbot interaction.
    - /history: Displays the history of user interactions.
    - /history/<interaction_type>/<filename>: Allows downloading specific interaction logs.

IV. Additional Notes:

- **Error Handling:** Implement robust error handling to gracefully manage situations like invalid API keys, incorrect user input, or failed image processing.
- **Security:** The current authentication is very basic. For a production environment, consider using more secure authentication methods and protecting sensitive information like API keys more effectively.
- **Data Privacy:** Ensure compliance with data privacy regulations, especially if handling sensitive medical information. Consider anonymizing or encrypting stored user data.
- **Model Updates:** Keep track of updates and improvements to the Gemini API and update the

model used in your application accordingly.