**Gemini-Powered Chatbot with RAG**

**Overview**

This project implements a mental health support chatbot powered by the Google Gemini API, enhanced with Retrieval-Augmented Generation (RAG) for improved contextual accuracy. It features a user-friendly Streamlit interface and utilizes a Chroma vector database for efficient information retrieval.

**Features**

* **Gemini Integration:** Uses the gemini-2.0-flash model for generating intelligent and context-aware responses.
* **Retrieval-Augmented Generation (RAG):** Employs Google's models/embedding-001 to generate embeddings and retrieve relevant information.
* **Interactive UI:** Built with Streamlit to provide a clean and accessible chat experience.
* **Chroma Vector Database:** Stores and retrieves text embeddings efficiently to support RAG functionality.

**Project Structure**

* appUI.py — Streamlit-based UI for user interaction.
* chatbot.py — Core chatbot logic and Gemini API integration.
* embeddings.py — Handles embedding creation and Chroma DB operations.
* requirements.txt — Lists all necessary dependencies.
* chroma\_gemini\_db/ — Directory for storing the vector database.

**Dependencies**

Dependencies are listed in requirements.txt. Example entries include:

langchain\_community

streamlit

google-generativeai

chromadb

Install them with:

pip install -r requirements.txt

**Setup Instructions**

1. **Clone the Repository:**
2. git clone <your\_repository\_url>
3. cd <your\_repository\_directory>
4. **Install Dependencies:**
5. pip install -r requirements.txt
6. **Configure the API Key:**

You’ll need a valid Google Cloud API key for the Gemini API.

* + Set the environment variable:
  + export GOOGLE\_API\_KEY="YOUR\_API\_KEY"
  + Alternatively, in Python:
  + import os
  + os.environ["GOOGLE\_API\_KEY"] = "YOUR\_API\_KEY"

1. **Ensure the Vector Database Exists:**

The chroma\_gemini\_db folder should be present. If not, generate it by running the relevant functions from embeddings.py.

1. **Launch the Chatbot:**
2. streamlit run appUI.py

**Usage**

* Open your browser to the local Streamlit server (usually at <http://localhost:8501>).
* Enter queries in the chat interface.
* The chatbot will respond based on Gemini-generated results and relevant information retrieved from the vector database.

**How It Works**

**appUI.py**

* Builds the Streamlit UI.
* Handles user inputs and displays chatbot responses.
* Communicates with chatbot.py.

**chatbot.py**

* Implements chatbot logic.
* Initializes the gemini-2.0-flash model.
* Uses embeddings.py to retrieve relevant context via RAG.
* Generates and returns responses.

**embeddings.py**

* Creates text embeddings using models/embedding-001.
* Stores and retrieves data from the Chroma vector store.
* Supports RAG by providing context to the chatbot.

**Data Flow**

1. User submits a query via the Streamlit UI.
2. The query is passed to chatbot.py.
3. chatbot.py calls functions in embeddings.py to:
   * Generate an embedding for the query.
   * Retrieve similar embeddings from chroma\_gemini\_db.
4. A response is generated using gemini-2.0-flash, incorporating retrieved information.
5. The response is displayed back to the user.

**Notes**

* Ensure a valid Google API key is available and correctly set as an environment variable.
* The vector database should be initialized for RAG to function properly.

Let me know if you'd like to add screenshots, examples of interactions, or customize the tone for a portfolio, academic report, or README!