Project Information

Name: Soumy Jain
Email ID: jainsoumya7378@gmail.com
Deployed Backend: https://square-sherie-sucker3699-4d0288b8.koyeb.app/
Deployed Frontend: https://stan-bot-frontend.vercel.app/
GitHub Repository: https://github.com/Frontkick/stan-bot

Miko Chatbot API

Miko is a human-like, emotionally intelligent, context-aware chatbot API built with Flask and Google Gemini, capable of "remembering" users, adapting tone, and supporting rich back-and-forth conversations with personalized memory.

Features

- **Human-like empathy & tone adaptation** (casual, friendly, formal, playful…)
- **Personalized user memory**: Remembers your name, interests, and previous chats with long/short-term memory
- **Powered by Google Gemini (`google-generativeai`)**
- **SQLite for light, persistent backend memory (per user)**
- **Modular 5-file architecture—easy to extend or deploy**
- **REST API: ready for integration with web/mobile apps**
- **Example usage and test method included**

Setup & Installation

1. **Clone the repo or get these files in a directory:**
    - `app.py`
    - `db.py`
    - `utils.py`
    - `gemini_client.py`
    - `requirements.txt`
2. **Install dependencies:**

```
pip install -r requirements.txt
# Or individually:
# pip install flask google-generativeai
```

3. **Get your Google Gemini API key.**
    - Go to Google AI Studio

- Create and copy an API key

4. **Set your API key in .env file**

```
GOOGLE_API_KEY = your-gemini-api-key-here
```

Run the Server

```
python3 app.py
```

API Usage

POST `/chat`

Send a conversation turn to the bot.

Request:

```
{
"user_id": "alex123",
"message": "Hi, my name is Alex. I like gaming and pizza."
}
```

Response:

```
{
"bot": "Miko",
"reply": "Hey Alex, I'm here for you. Want to chat about gaming or pizza to
lift your mood?",
"user_profile": {
"name": "Alex",
"likes": "gaming;pizza"
}
}
```

Project Workflow

How it Works

1. **API Receives Message:** Each user/message arrives at `/chat`.
2. **Profile & Memory Lookup:** Looks up user’s profile/interests and prior messages in SQLite.

3. **Context + Memory Prompt:** Constructs a tailored "system prompt" with:
    - User's known facts (e.g. "My name is X", "I like...") and chat history summaries
    - Conversation tone detection (empathetic, cheerful, etc.)
4. **Chatbot Response:** Sends prompt to Google Gemini for a reply.
5. **Memory Update:** New facts are extracted and remembered in future conversations.
6. **Reply Returned:** API provides a memory-aware, emotionally intelligent response.

File Structure

File	What it does
app.py	Flask app, HTTP routes, glue
db.py	All SQLite/user/memory database code
utils.py	Embedding, memory search, tone, prompt/facts
gemini_client.py	Handles Gemini LLM setup and calls
requirements.txt	All dependencies for quick installation

Testing & Example Scenarios

1. Long-Term Memory Recall

- POST: `{ "user_id": "bob87", "message": "My name is Bob. I like sci-fi." }`
- Later, POST: `{ "user_id": "bob87", "message": "What do you know about me?" }`
- Response will reference "Bob" and "sci-fi".

2. Context-Aware Tone

- POST: `"I'm feeling sad"` → Bot responds with empathy.
- POST: `"Let's roast someone!"` → Bot uses a playful tone.

3. Personalization Over Time

- Mention `"I live in Delhi"`; later messages reference location or prior interests.

4. Response Naturalness & Diversity

- POST: `"hi"`, `"hello"`, `"what's up"` — bot gives varied, human-like greetings.

5. Identity Consistency

- POST: `"Are you an AI?"`, `"What's your name?"`
- Miko never reveals itself as AI, stays "in character."

6. Hallucination Resistance

- POST: `"Did you see me yesterday?"`
- Bot offers a playful but grounded response without fabricating events.

Quick Curl Example

```
curl -X POST -H "Content-Type: application/json"
-d '{"user_id":"sammy","message":"My name is Sam. I feel great today!"}'
http://localhost:5000/chat
```