WhatsApp Memory Assistant

Application Scenario

Imagine you start your day by greeting your personal AI via WhatsApp. You send a photo of your new haircut, record a voice note about your to-dos, and send a text note listing grocery items. Days later, you ask “What did I plan for dinner?” or “Show me my hairstyle update from last week,” and the bot retrieves the relevant memories.

Overview

Build a WhatsApp chatbot using Twilio’s WhatsApp API and Mem0’s memory layer to ingest and recall images, audio, and text as memories. The bot should support natural language queries and a command to list all saved memories.
Additionally, design and implement your own database schema to persist interactions and derived memories, and use it to power listing, filtering, and simple analytics.

Objectives

- **Multimodal Ingestion:** Accept and process images, voice notes, and text messages.
- **Persistent Memory:** Store entries in Mem0 with metadata and embeddings for fast, semantic retrieval.
- **Database Persistence (your design):** Capture user interactions and memories in a database you choose. Demonstrate idempotent ingestion, media deduplication, and timezone-aware queries.
- **Interactive Chat:** Handle conversational queries using context awareness.
- **Memory Listing:** Implement a `/list` command to enumerate all user memories.
- **DB-backed Queries & Analytics:** Provide endpoints for recent interactions and a small analytics summary using your database.

- **Innovative Features:** (Creative freedom) e.g., mood detection, geo-tagging, scheduled reminders, or group memory sharing etc.

Tasks

1. **Twilio Integration**
    - Provision a Twilio sandbox number for WhatsApp.
    - Configure webhooks to receive inbound messages and fetch media URLs.
2. **Backend Service**
    - Set up a Node.js / Python / serverless service.
    - Install and configure the Mem0 SDK with API key.
    - **Add a database layer (SQLite, Postgres, or similar) with migrations—design the schema yourself** to store:
        - Users and their WhatsApp identity.
        - Inbound messages (text, image, audio), media metadata, and any transcripts.
        - The memories you create (and their linkage to source interactions and Mem0).
        - **Required behaviors (implementation details are up to you):**
            - **Idempotent ingestion:** processing the same Twilio message twice must not create duplicates (use a stable dedup key).
            - **Media deduplication:** identical media should be stored once and re-referenced (use a repeatable content fingerprint).
            - **Linkage:** each memory should be traceable to its originating interaction.
            - **Timezone-aware filtering:** support queries like “last week” in the user’s timezone.
    - Create endpoints:
        - `POST /webhook` to handle incoming messages.

- `POST /memories` to add multimodal memories.
- `GET /memories?query=<text>` to search memories (via Mem0) and enrich responses with your DB.
- `GET /memories/list` to return all memories (read from your DB, newest first).
- **NEW:** `GET /interactions/recent?limit=<n>` to return recent interactions from your DB.
- **NEW:** `GET /analytics/summary` to return simple DB-derived stats (e.g., totals by type, top tags/labels if you use them, last ingest time).

3. **Multimodal Processing**
   - Images: Use Mem0 to create memories from images; persist media metadata and memory linkage in your DB.
   - Audio: Download, transcribe with Whisper, persist the transcript in your DB, and create a memory via Mem0.
   - Text: Directly ingest to Mem0; also persist the original interaction in your DB.
4. **Business Logic & Chat Flows**
   - On message receipt, detect type, persist interaction, and add a memory.
   - On query, search Mem0 and format responses; include DB details (when/where it came from, any labels/tags you store).
   - On `/list`, return memories **from your DB**.
   - Ensure idempotency and media dedup in ingestion.
   - Handle basic errors and edge cases gracefully.
5. **Demo**
   - Record a 3–5 min demo: ingesting media, querying by intent, listing memories.
   - **Show DB usage live:** call `/interactions/recent` and `/analytics/summary`.
6. **Documentation**
   - README: Setup steps for Twilio, environment variables, deployment.

- API reference: Endpoint definitions and sample payloads.
- **Schema overview (your design):** brief ERD or diagram, key constraints, and why you chose them.
- **Notes on idempotency/dedup/timezone handling:** explain your approach.

Deliverables

- Public GitHub repo with server and any client code.
- Demo video link (YouTube, Vimeo, etc.).
- Comprehensive README with setup, usage, and architecture.
- **Database migrations/DDL** and a small seed script (your schema).

Evaluation Criteria

- **End-to-End Functionality:** Successful memory ingestion and retrieval; natural WhatsApp UX.
- **Code Quality:** Readable, maintainable, with proper error handling.
- **API & Data Design:** Clear payloads and a coherent schema of your own design; idempotent ingestion and media dedup working.
- **DB Queries & Listing:** `/memories/list` reads from your DB; `/interactions/recent` and `/analytics/summary` are correct and responsive.
- **UI/UX via WhatsApp:** Clear, helpful responses and prompts.
- **Innovation:** Creative features beyond the basics.
- **Documentation & Demo:** Clarity, completeness, and professionalism.

Resources

- Twilio API for WhatsApp: https://www.twilio.com/docs/whatsapp/api
- Mem0 GitHub: https://github.com/mem0ai/mem0
- Mem0 Docs: https://docs.mem0.ai
- OpenAI CLIP: https://github.com/openai/CLIP

- OpenAI Whisper: https://github.com/openai/whisper