**AI Financial Support – Task 4**

Small CLI Python app that talks to Google GenAI (Gemini) to answer finance-related questions and extract user details.

* Uses two Gemini API keys: one for general chat/generation and another for dedicated extraction tasks.
* Keeps a running chat history in-memory and attempts to extract personal details from user inputs (name, account number, goal, contact, etc.).

**Contents**

* app.py — main CLI program and business logic
* requirements.txt — Python dependencies
* .env — (not committed) recommended place for API keys in development
* app2.ipynb — an alternative notebook (if present)
* Dockerfile
* setup.sh

**Archietecture**

* **User Interface:** CLI input/output loop.
* **Conversation Engine:** Uses the new **google-genai** Python client with gemini-2.5-flash for chat generation.
* **Key Difference from Old API:**
  + No REST calls or requests; everything is done via the official Python client.
  + Cleaner client creation with genai.Client(api\_key=...).
  + Supports chats.create() and models.generate\_content() natively.
* **Extractor Engine:** Secondary Gemini client for structured data extraction (returns JSON).
* **Memory Layer:** chat\_history and user\_db to store ongoing conversation and extracted info.
* **Environment Management:** .env file + dotenv to securely load API keys.
* **Lifecycle Management:** atexit and signal handlers ensure clean shutdown.

**Detailed Flow**

A diagram of a chat bot

AI-generated content may be incorrect.

1. **User Input:** User enters a query or message.
2. **Message Handling:** give\_prompt() sends query to Gemini model.
3. **AI Response:** Gemini returns an answer displayed to user.
4. **Detail Extraction:** extract\_details() asks second Gemini model to parse personal info in JSON.
5. **Database Update:** update\_db() updates in-memory record if new info appears.
6. **Exit Handling:** On exit or Ctrl +C, connections are closed gracefully.

**Containerization with Docker**

* Docker still ensures isolated, reproducible deployment.
* .env provides API keys for both chat and extraction Gemini clients.

**Key Features**

* Highlight modern Python client instead of REST-based legacy API.
* Mention built-in chat and content generation methods.
* Emphasize JSON extraction is handled directly via model configuration (response\_mime\_type="application/json").
* Stress that session management is simplified (no manual token handling).