

# Multimodal Chat AI Service (Text + Images)

YOU ARE OPEN TO USE ANY AI TO COMPLETE IT. BUT WILL BE HIRED BASED ON THE LEVEL OF UNDERSTANDING ON THE TECHNOLOGY STACK USED HERE

## Project Overview

Build a backend AI service that allows **conversational understanding of images integrated into text chat**. The system combines:

- A chat-based LLM (candidate choice: LLama 8B/13B/70B)
- A Vision-Language Model (VLM) for understanding images
- Multimodal reasoning: the LLM receives embeddings or processed info from the VLM to incorporate images into the conversation

The system exposes APIs for:

- Text chat with optional image input
- Image-based conversation where the LLM “knows” the image content via the VLM

All interactions (text, images, reasoning) should be logged for reproducibility and analysis.

## 2. Requirements

### 2.1 Functional Requirements

#### 2.1.1 User Management

- Users can register/login
- Users can view their conversation history (text + images)

**Users Table Fields:**

`id, email, password_hash, created_at, updated_at`

#### 2.1.2 Multimodal Chat Endpoint

- `/multimodal-chat` endpoint
- Accepts:
  - Text query (optional field)
  - Image upload(s) (optional field)
- Returns a model response that reasons about the text **in context of the image**
- Stores: user query, uploaded image(s), LLM response, model used, and timestamp

#### ConversationLogs Table Fields:

`id, user_id, text_query, image_url(s), response_text, llm_model_name, vlm_model_name, timestamp`

#### Key Requirement:

- The LLM should not just caption the image.
- The VLM processes the image into embeddings or structured info, which is passed to the LLM as context for reasoning.
- The response can reference image content, answer questions about it, or combine it with the textual conversation.

## 2.2 Core Design & Algorithm Requirements

- **OOP:** Classes: `MultimodalManager`, `ChatManager`, `VLMManager`
- **Data Structure:** Indexed relational table for logs + optional vector storage for embeddings
- **Algorithm:**
  1. Image uploaded → processed by VLM → embeddings or structured info
  2. Text query + VLM output → LLM → response
  3. Store full conversation with image context
- **Design Pattern:** Strategy pattern for model selection (LLama 8B/13B/70B + BLIP-2 / LLaVA / GPT-4V / Hugging Face VLM)

- **Performance:** Async endpoints for I/O-heavy tasks (embedding computation, image upload, LLM inference)

### 3. Non-Functional Requirements

- FastAPI backend with REST endpoints
- Validation for text, image, and combined queries
- Logging, error handling, secure API key storage
- Modular and scalable design to support multiple VLMs/LLMs

### 4. Deliverables

#### 4.1 Documentation

- System architecture diagram (Text + Image → VLM → LLM → Response)
- ERD for `Users` and `ConversationLogs`
- API documentation (Swagger/Postman)
- Sequence diagram: text + image → VLM → LLM → response

#### 4.2 Code Deliverables

- FastAPI backend module for multimodal chat
- Seed data for users and sample images

#### 4.3 AI Integrations

- LLama model integration
- Vision-Language Model integration (BLIP-2, LLaVA, GPT-4V, or Hugging Face)
- Pipeline that passes VLM output to LLM

#### 4.4 Testing

- Unit tests for `MultimodalManager`, `ChatManager`, and `VLMManager`

- API tests for text + image chat
- End-to-end tests: text + image → VLM → LLM → response → log storage

#### **4.5 Deployment**

- Docker configuration for backend + DB
- Instructions to run locally and test endpoints