Project Name: Multimodal Hospitality Concept Generator

---

1. Introduction

The Multimodal Hospitality Concept Generator is an AI-based application that generates a hospitality business concept in both textual and visual form based on user input.

The system uses:

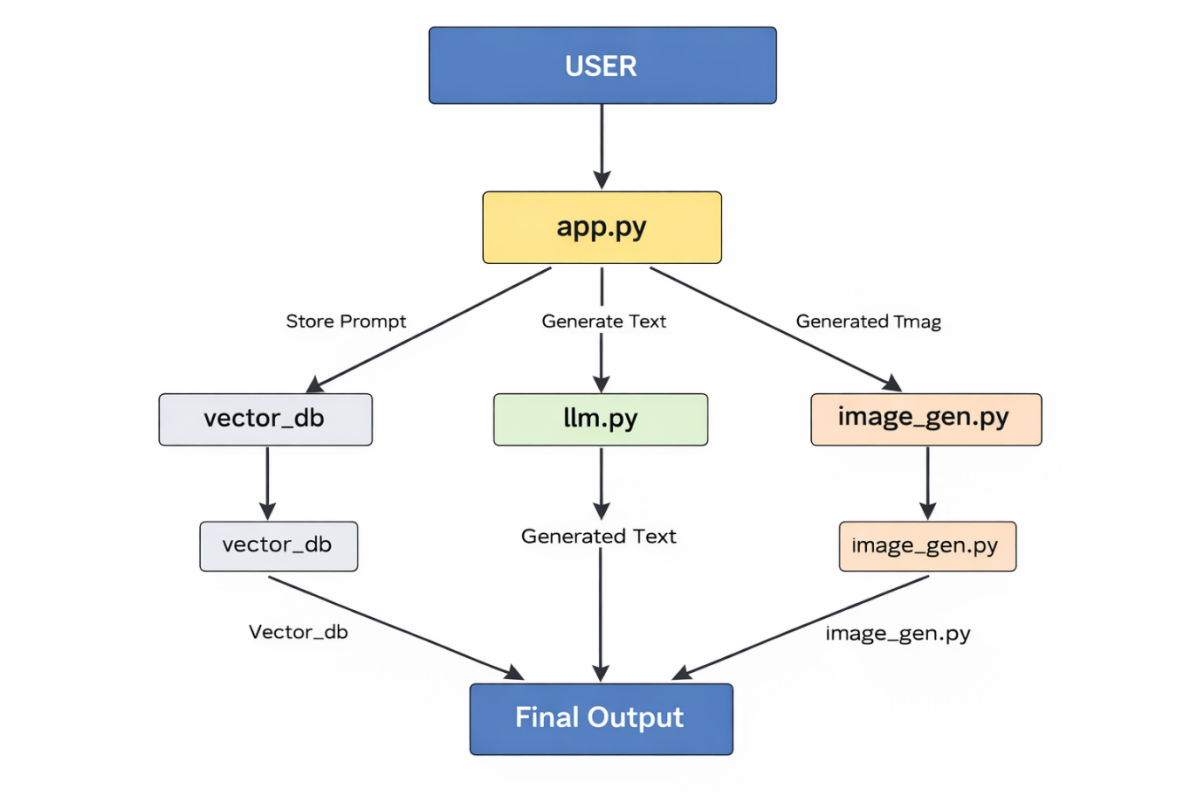
- A Language Model (for text generation)
- An Image Generation Model (for visual output)
- A Vector Database (for storing user prompts)

The purpose of this system is to automate the concept creation process for hotels, cafes, and resorts.

---

2. System Overview

The system follows a modular architecture where each module has a specific responsibility.

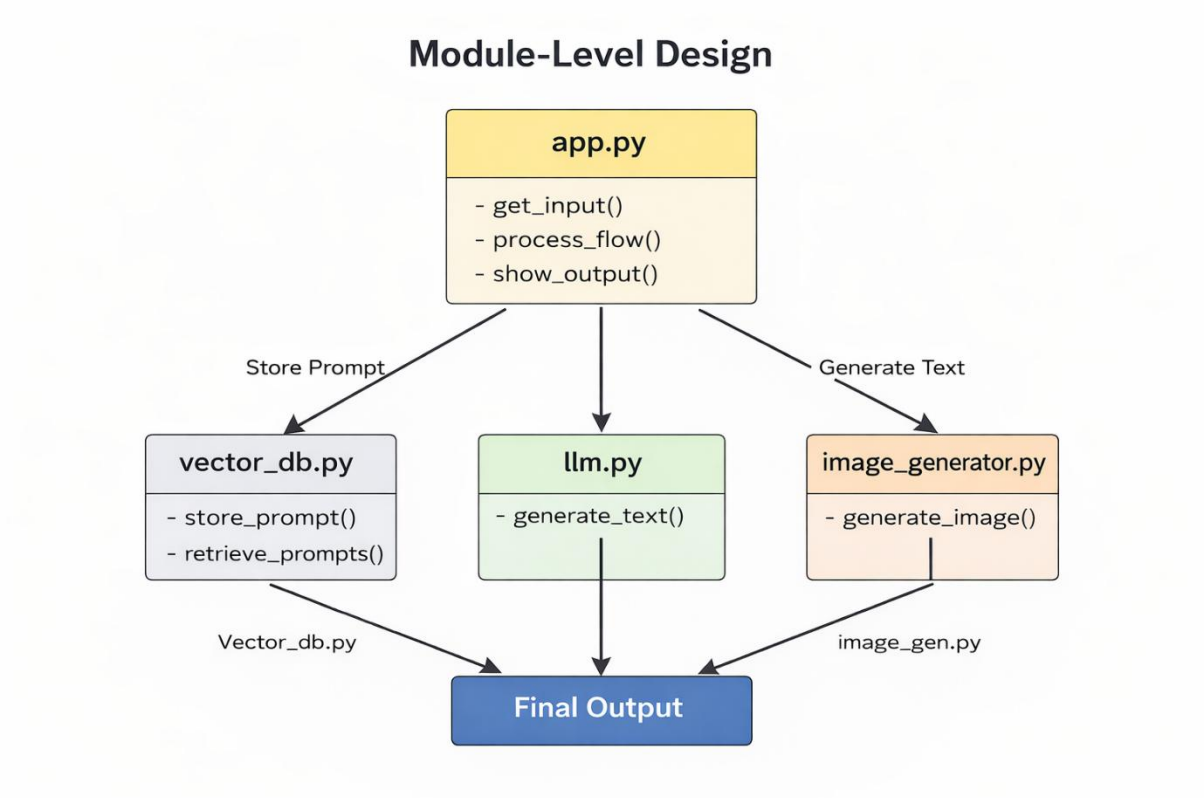


System Components:

1. Main Controller (app.py)
2. Text Generation Module (llm.py)
3. Image Generation Module (image_generator.py)

4. Vector Database Module (vector_db.py)

3. Module Design



3.1 Main Controller Module (app.py)

Responsibility:

- Accept user input
- Coordinate between different modules
- Display output

Functions:

- Take user prompt
- Call store_prompt()
- Call generate_narrative()
- Call generate_image()
- Print results

Internal Flow:

1. User enters hospitality concept
2. Prompt stored in database

3. Text generated
4. Image generated
5. Results displayed

3.2 Text Generation Module (llm.py)

Responsibility:

Generate detailed hospitality description using an AI language model.

Function:

generate_narrative(prompt)

Working:

1. Receive prompt
2. Send prompt to language model API
3. Receive generated text
4. Return text to main controller

Input:

String (User Prompt)

Output:

String (Generated Narrative)

3.3 Image Generation Module (image_generator.py)

Responsibility:

Generate visual representation of hospitality concept.

Function:

generate_image(prompt)

Working:

1. Receive prompt
2. Send prompt to image generation API
3. Receive image data
4. Save image as PNG file
5. Return image path

Input:

String (User Prompt)

Output:

Image File Path (generated_image.png)

3.4 Vector Database Module (vector_db.py)

Responsibility:

Store and retrieve user prompts for memory and similarity search.

Functions:

store_prompt(prompt)
retrieve_prompts(query)

Working:

1. Convert prompt into embeddings
2. Store embeddings in ChromaDB
3. Assign unique ID to each prompt
4. Allow retrieval of similar prompts

Input:

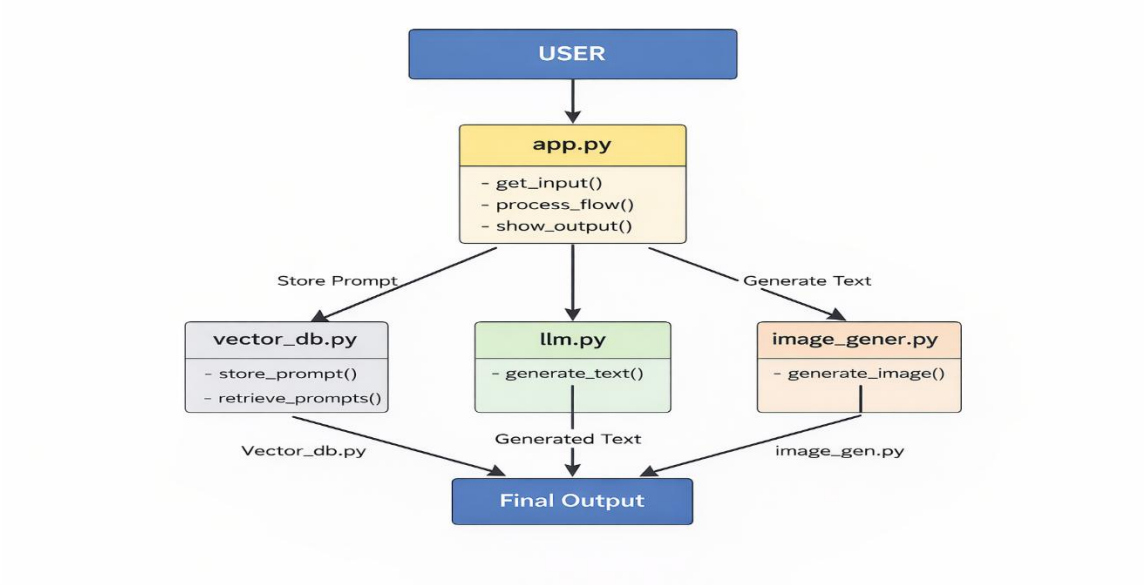
String (User Prompt)

Output:

Stored document reference / Retrieved results

---

4. Data Flow Design

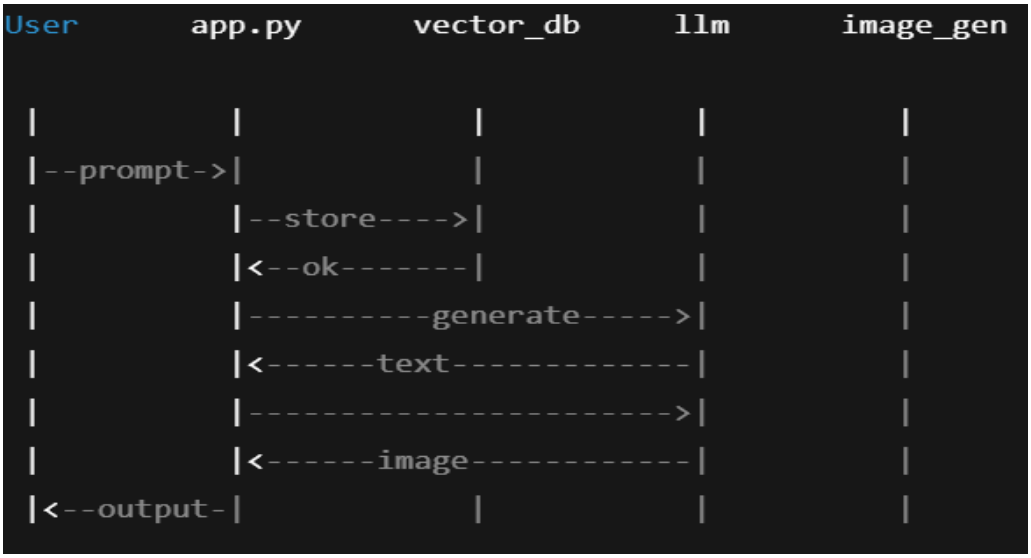


Step-by-step Data Flow:

1. User enters hospitality concept.
2. app.py receives input.
3. Prompt stored in vector database.
4. Prompt sent to text generation module.
5. Prompt sent to image generation module.
6. Generated narrative and image returned.
7. Results displayed to user.

---

5. Sequence Flow (Runtime Execution)



1. Application starts.
2. User provides input.
3. System stores prompt.
4. System generates text.
5. System generates image.
6. Output displayed.
7. Application ends.

---

6. Data Storage Design

The system stores user prompts in a vector database (ChromaDB).

Each stored prompt includes:

- Unique ID
- Original text
- Vector embedding

Purpose:

- Memory storage
- Similarity search
- Future RAG implementation

---

7. Error Handling Design

Possible Errors:

- API failure
- Network issues
- Invalid API key
- Image generation failure

Handling Strategy:

- Display error message
- Prevent corrupted image saving
- Retry mechanism (future improvement)

---

8. Future Enhancements

- Add Graphical User Interface (GUI)
- Add multiple image generation
- Add prompt history view
- Deploy as web application
- Add authentication system

---

9. Conclusion

The system follows a modular low-level design where each module performs a single responsibility.
The controller coordinates between modules while maintaining separation of concerns.
The design allows easy scalability and future improvements.

Current system supports text and image generation and can be extended for recommendation systems in future.