Project - High Level Design

Multimodal Hospitality Creator – AI Powered Hospitality Concept Visualization System

Course Name: Generative AI

Institution Name: Medicaps University – Datagami Skill Based Course

Sr no	Student Name	Enrolment Number
1	HARSH RAI	EN22CS301393
2	HEMANT DHAKAD	EN22CS301421
3	GAURAV DWIVEDI	EN23CS3L1008
4	HIMANI JAISWAL	EN22CS30123
5	HARSHWARDHAN YADAV	EN22CS301416
6	GOUTAM VERMA	EN22CS301375

Group Name: 12D4

Project Number: 48

Industry Mentor Name:

University Mentor Name: Divya Kumawat

Table of Contents

1. Introduction

The Multimodal Hospitality Creator is a Generative AI-based system developed to visualize hospitality concepts using advanced AI models. The system integrates a Large Language Model (LLM) with Image Generation models through APIs to transform textual prompts into both detailed narrative descriptions and high-quality visual outputs.

The objective of this project is to demonstrate the integration of text-to-text and text-to-image AI systems to generate immersive hospitality designs such as luxury hotels, resorts, themed restaurants, and premium interiors.

This system enables users to rapidly prototype hospitality ideas for academic, conceptual, and professional presentation purposes.

1.1 Scope of the Document

This document provides the High-Level Design (HLD) of the Multimodal Hospitality Creator system. It includes:

- Overall architecture
- Application design
- Data design and flow
- API integrations
- Security and performance considerations
- Non-functional requirements

This document serves as a technical reference for implementation and evaluation.

1.2 Intended Audience

The intended audience for this document includes:

- Developers and Software Engineers
- AI Engineers
- Academic Evaluators
- Industry Mentors
- System Architects

1.3 System Overview

The system allows users to enter a hospitality-related textual prompt such as:

- "Luxury beach resort with infinity pool"
- "Dark themed premium hotel lobby with golden lighting"

The system workflow includes:

1. Accepting user input.
2. Sending the input to an LLM API for prompt enhancement.
3. Generating structured descriptive content.
4. Creating embeddings and storing them in a Vector Database.
5. Sending structured prompts to an Image Generation API.
6. Displaying generated images and descriptions in a web interface.

The system is modular and API-driven.

2. System Design

The system follows a layered modular architecture.

Layers include:

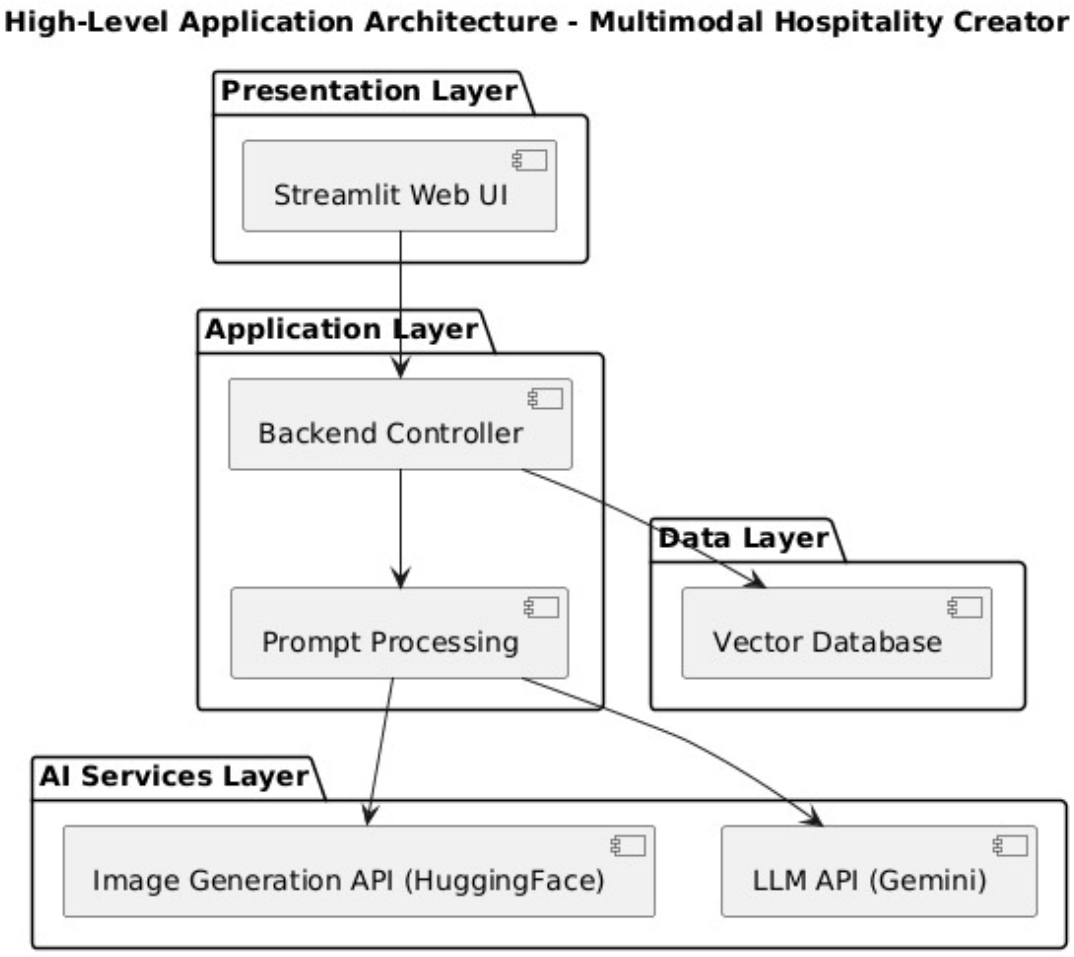
- Presentation Layer (Frontend UI)
- Application Layer (Business Logic)
- AI Services Layer (LLM + Image Generation APIs)
- Data Layer (Vector Database)

The design ensures scalability, modularity, and ease of integration.

2.1 Application Design

Frontend:

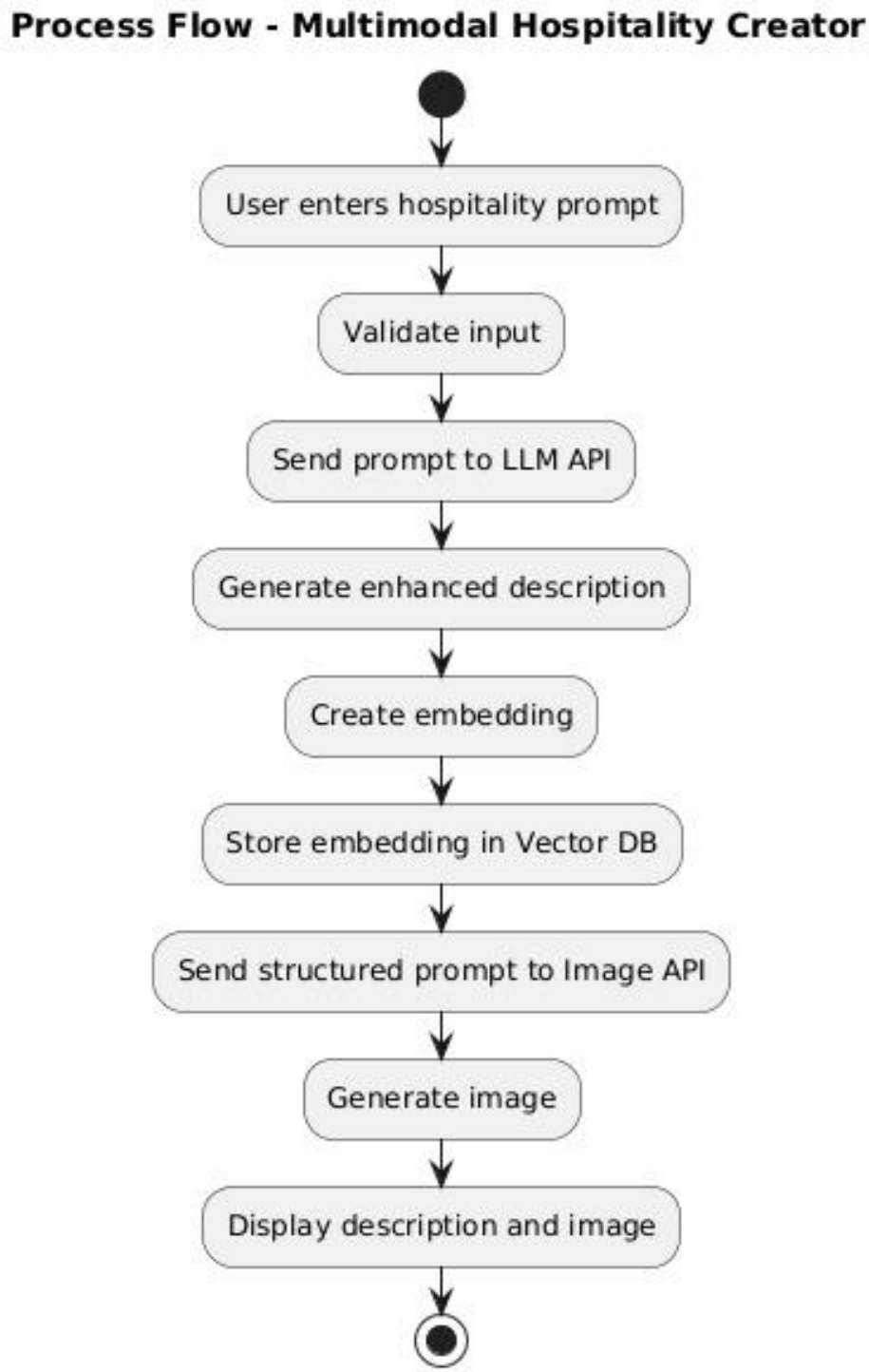
- Built using Streamlit
- Luxury dark theme
- User prompt input interface
- Display area for text and image outputs



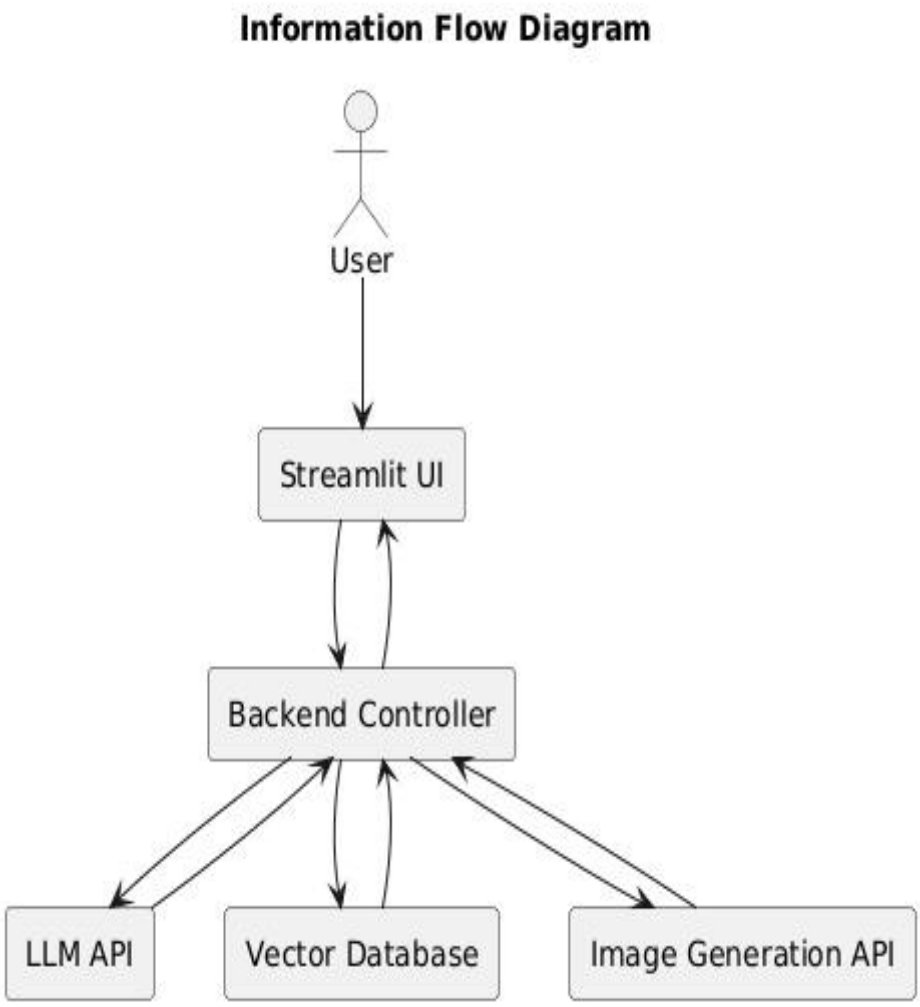
Backend:

- Python-based processing
- Handles API communication
- Performs prompt engineering
- Interacts with Vector Database

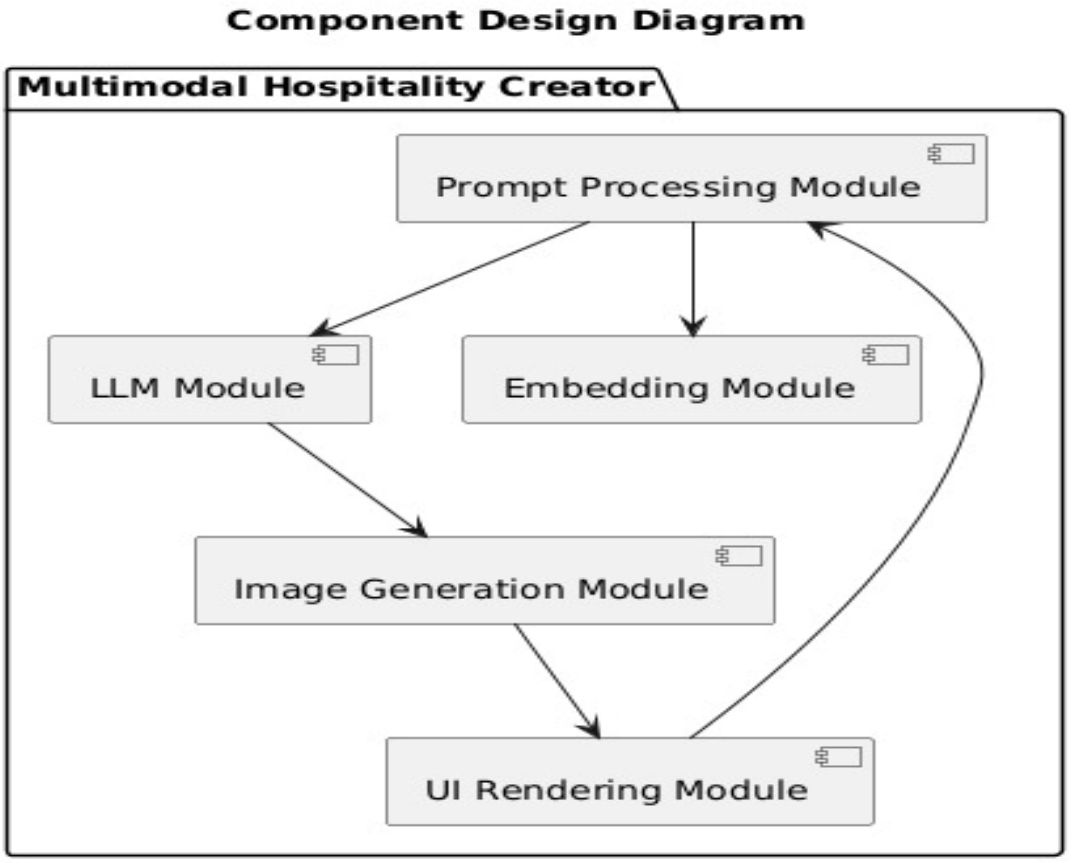
2.2 Process Flow



2.3 Information Flow



2.4 Components Design

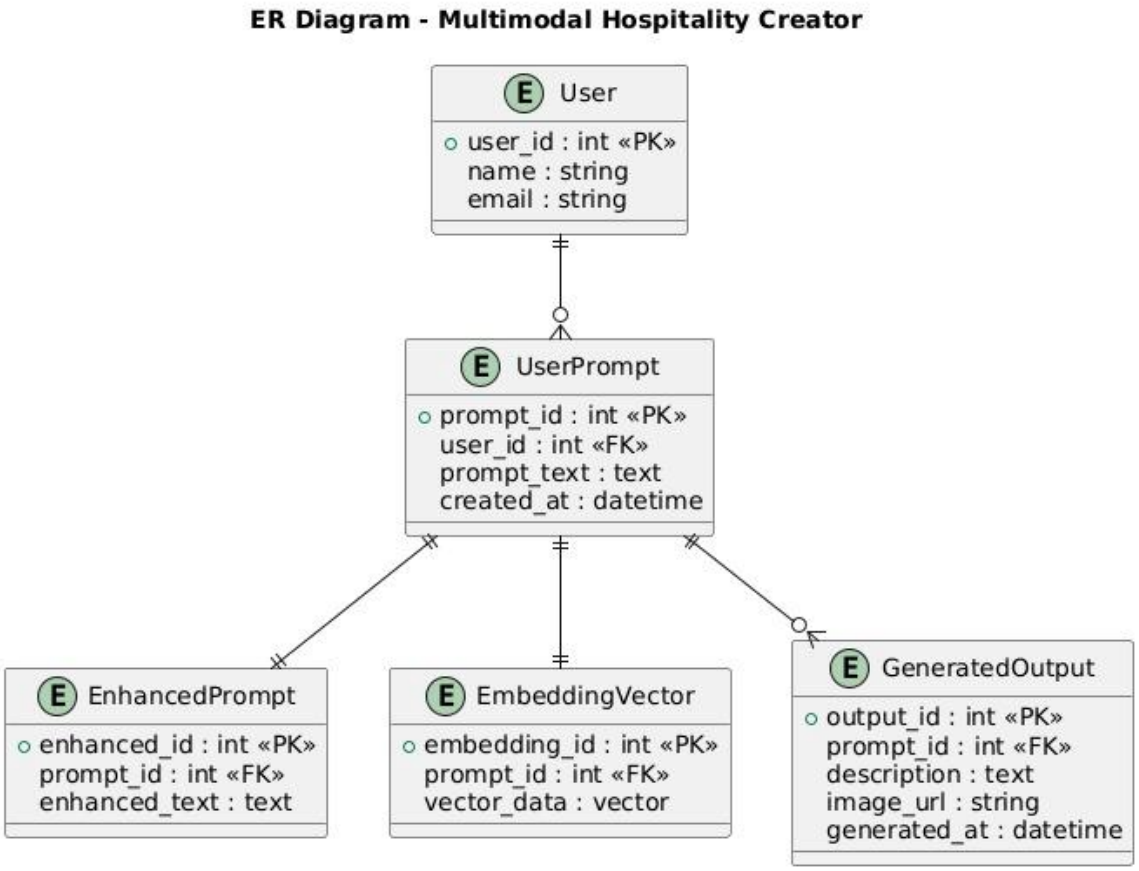


1. Prompt Processing Module
    - Cleans and enhances user input
    - Adds stylistic and structural details
2. LLM Module
    - Generates descriptive narratives
    - Structures design output
3. Vector Database Module
    - Stores embeddings
    - Enables semantic similarity

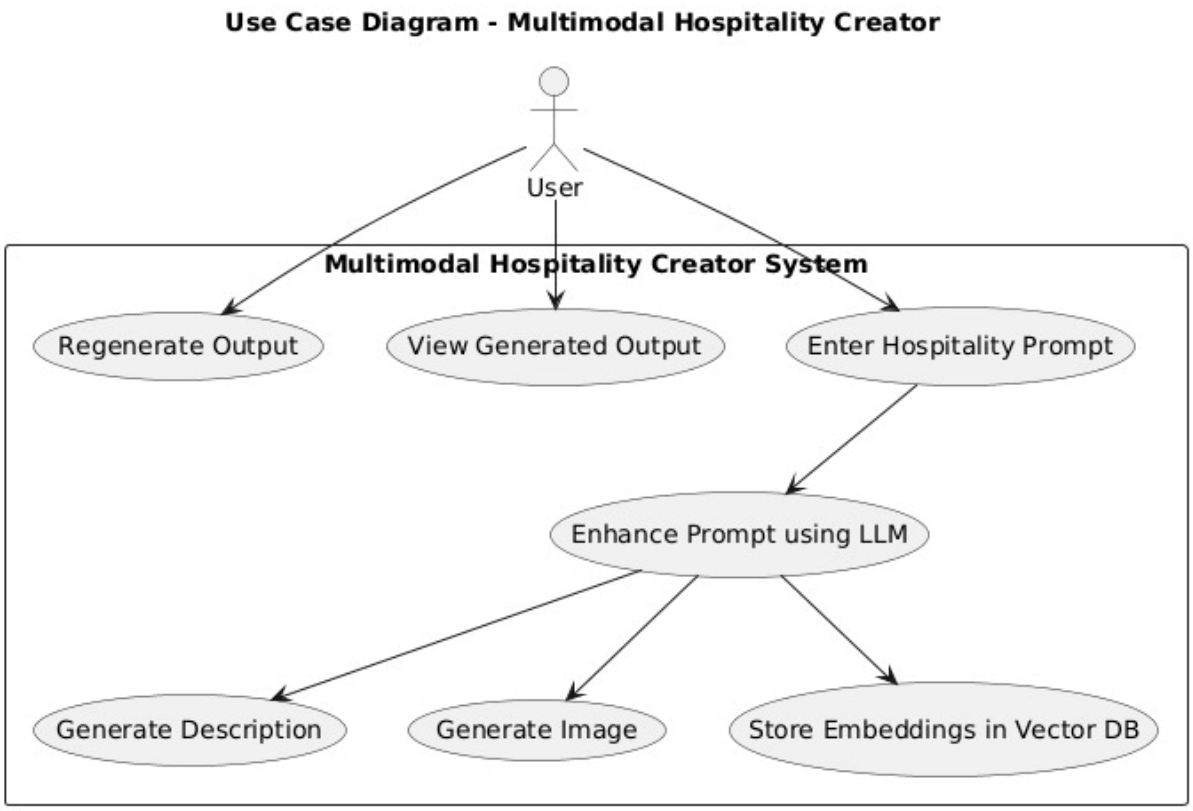
- Maintains historical prompt data

4. Image Generation Module
    - Converts text prompts into high-resolution images
    - Uses diffusion-based models
5. UI Rendering Module
    - Displays outputs
    - Allows regeneration

2.5 ER Design:



2.6 Use Case Diagram



2.7 Key Design Considerations

- Modular architecture for scalability
- Secure API key management
- Efficient embedding storage
- Low latency API handling
- Future extensibility for additional AI models

2.8 API Catalogue

LLM API (Gemini or equivalent)

- Prompt enhancement
- Narrative generation

Image Generation API (HuggingFace diffusion models)

- Text-to-image conversion

Vector Database API

- Embedding storage
- Similarity search

3. Data Design

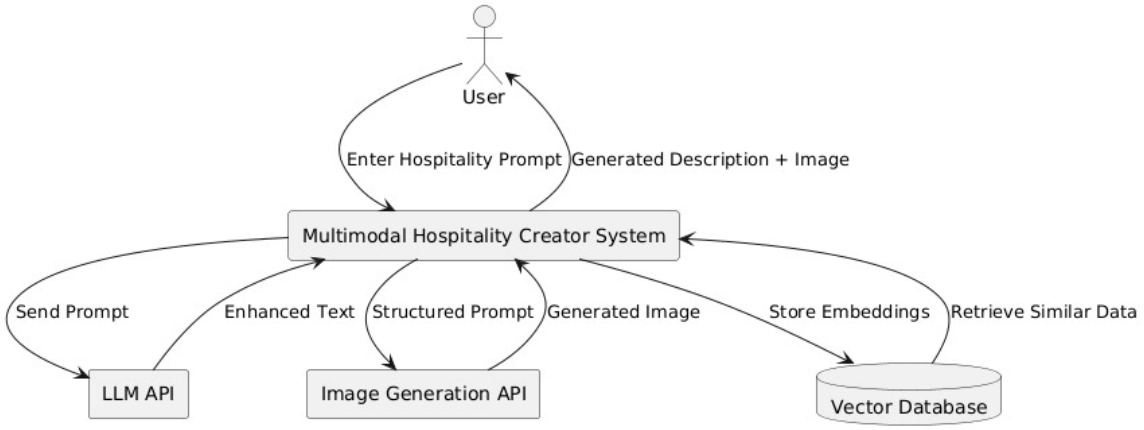
The system manages structured and unstructured data.

Data types include:

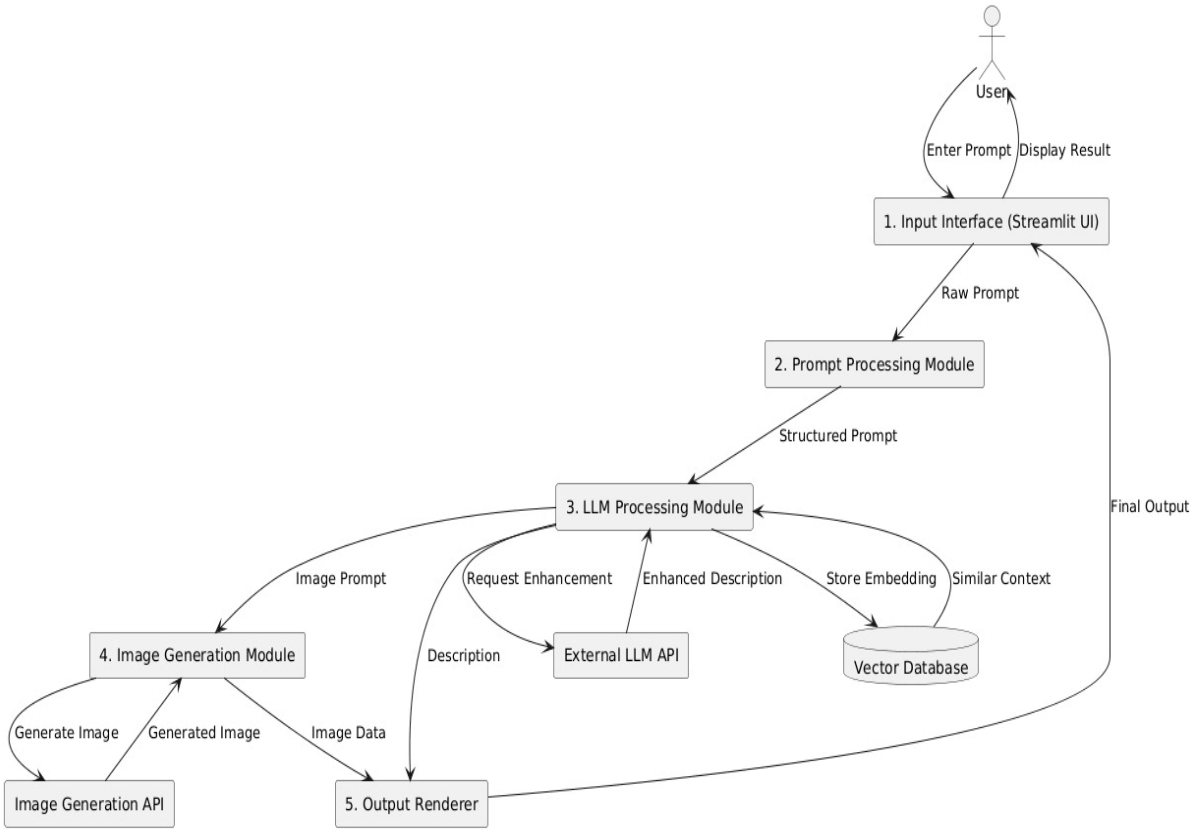
- User prompts
- Enhanced prompts
- Embeddings
- Generated descriptions
- Generated image metadata

3.1 Data Model

DFD Level 0 - Context Diagram (Multimodal Hospitality Creator)



DFD Level 1 - Detailed Data Flow



3.2 Data Access Mechanism

- REST API Calls: The system communicates with external AI services (LLM API and Image Generation API) using REST APIs to send prompts and receive responses.
- JSON-Based Communication: Data exchange between frontend, backend, and APIs uses JSON format for structured and efficient request–response handling.
- Token-Based Authentication: Secure API access is maintained using authentication tokens stored as environment variables.
- HTTPS Communication: All data transmission occurs over HTTPS to ensure encrypted and secure communication

3.3 Data Retention Policies

- Embedding Storage: Prompt embeddings are stored in the vector database for semantic search and context reuse.
- Session Data: Temporary user session data is stored in application state and cleared after session completion.
- Logging: System logs are maintained for monitoring and debugging purposes.
- Data Privacy**:** No sensitive user information is permanently stored

3.4 Data Migration

Future scalability options include:

- **Cloud Migration:** Supports migration to cloud-based vector databases for scalability.
- **Model Upgrades:** Architecture allows easy replacement or upgrade of AI models.
- **Schema Evolution:** Data structure can be extended without affecting existing functionality.

4. Interfaces

4.1 User Interface

- Web-based interface developed using Streamlit.
- Allows users to enter hospitality prompts and view generated descriptions and images.

4.2 External AI Interfaces

- LLM API: Used for prompt enhancement and narrative generation.
- Image Generation API: Used for converting structured prompts into visual outputs.

4.3 Data Interface

- Vector Database Connection: Stores and retrieves prompt embeddings for semantic similarity and context management.

5. State and Session Management

- Managed using Streamlit session state.
- Temporarily stores user prompts and generated results during active sessions.
- Session resets upon application refresh.
- Optional persistent storage can be implemented for maintaining prompt history

6. Caching

- Frequently requested prompts are cached to reduce repeated API calls.
- Embeddings are reused for semantically similar queries.
- API responses may be cached to minimize latency.
- Improves overall system performance and efficiency

7. Non-Functional Requirements

The system must satisfy the following non-functional requirements:

- **Scalability:** Ability to handle increasing users, prompts, and AI requests efficiently.
- **Reliability:** Stable and consistent operation with proper error handling for API failures.
- **Maintainability:** Modular architecture enabling easy updates and model replacements.
- **Usability:** Simple and intuitive user interface for smooth interaction.
- **Security:** Secure handling of data, APIs, and system access.
- **Performance:** Fast response times with optimized processing and minimal latency

7.1 Security Aspects

- API keys stored securely using environment variables.
- All API communication secured via HTTPS protocol.
- Input validation and sanitization to prevent malicious input.
- Protection against injection and unauthorized requests.
- Access control mechanisms for backend services.

7.2 Performance Aspects

- Optimized prompt structure to reduce processing overhead.
- Asynchronous API handling to improve responsiveness.
- Efficient vector database search for embeddings.
- Reduced response latency through caching and structured processing.
- Minimal server-side resource consumption.

8. References

[1] Google AI, *Gemini API Documentation*. Available: https://ai.google.dev
Accessed: January 2026.

[2] Hugging Face, *Diffusion Models Documentation*. Available:
https://huggingface.co/docs/diffusers
Accessed: January 2026.

[3] Pinecone Systems Inc., *Vector Database Documentation*. Available:
https://docs.pinecone.io
Accessed: January 2026.

(If you are using another vector DB like FAISS or Chroma, replace accordingly.)

[4] Streamlit Inc., *Streamlit Official Documentation*. Available: https://docs.streamlit.io
Accessed: January 2026.

[5] Python Software Foundation, *Python 3 Official Documentation*. Available:
https://docs.python.org/3/
Accessed: January 2026.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*. MIT Press, 2016.