Multimodal Hospitality Creator

Course Name: Generative AI

Institution Name: Medicaps University – Datagami Skill Based Course

Student Name(s) & Enrolment Number(s):

Sr no	Student Name	Enrolment Number
01	Irifa Khaishagi	EN22CS301435
02	Akash Ghosh	EN22CS301082
03	Adarsh Dibey	EN22CS301037
04	Aman Gour	EN22CS301104
05	Aastha Gupta	EN22CS301019
06	Aeshna Preyasi	EN22CS301075

Group Name: Group 08D1

Project Number: GAI-08

Industry Mentor Name: Aashruti Shah

University Mentor Name: Prof. Ajaj khan

Academic Year: 2025-2026

1. Introduction

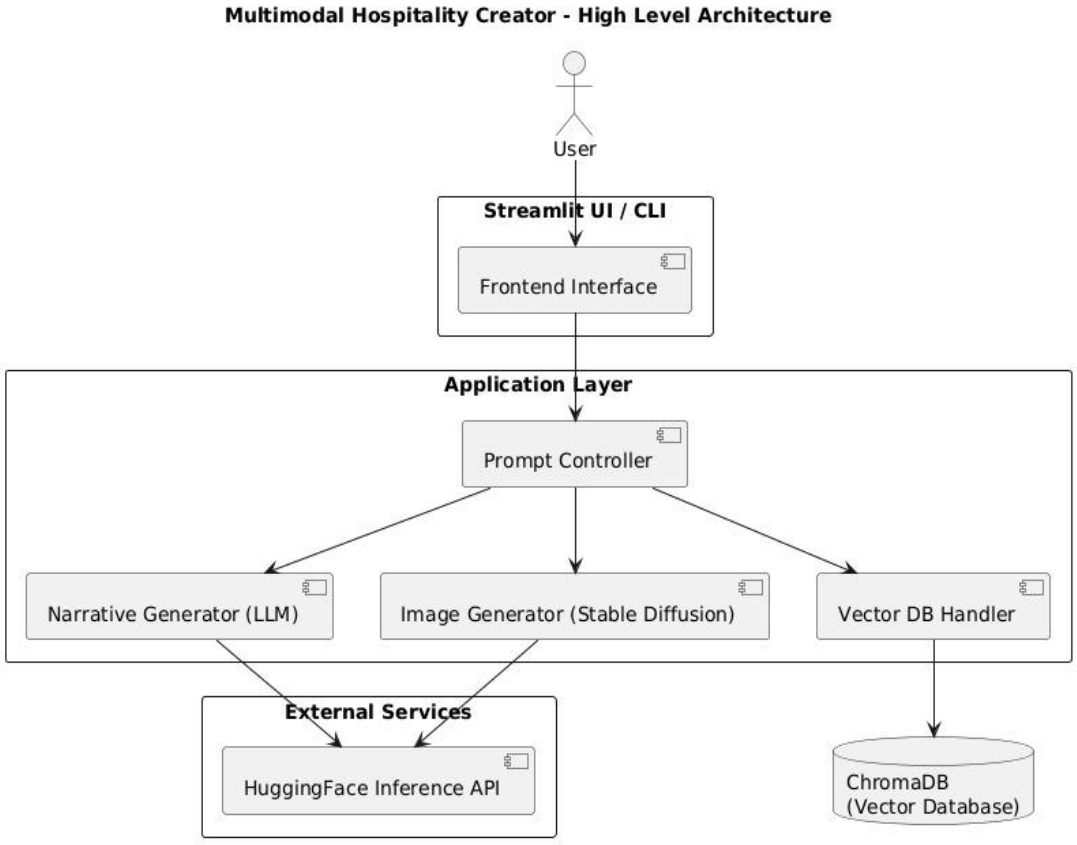
1.1. Scope of the document - This document provides the High-Level Design (HLD) for the Multimodal Hospitality Creator system. It covers Application Architecture, Multimodal AI integration, HuggingFace Inference API usage, System components and interactions, Vector database design and Prompt processing workflow

1.2. Intended Audience – AI/ML Engineers – Software developers – Academic Evaluators - Technical Review Panel - System Architects.

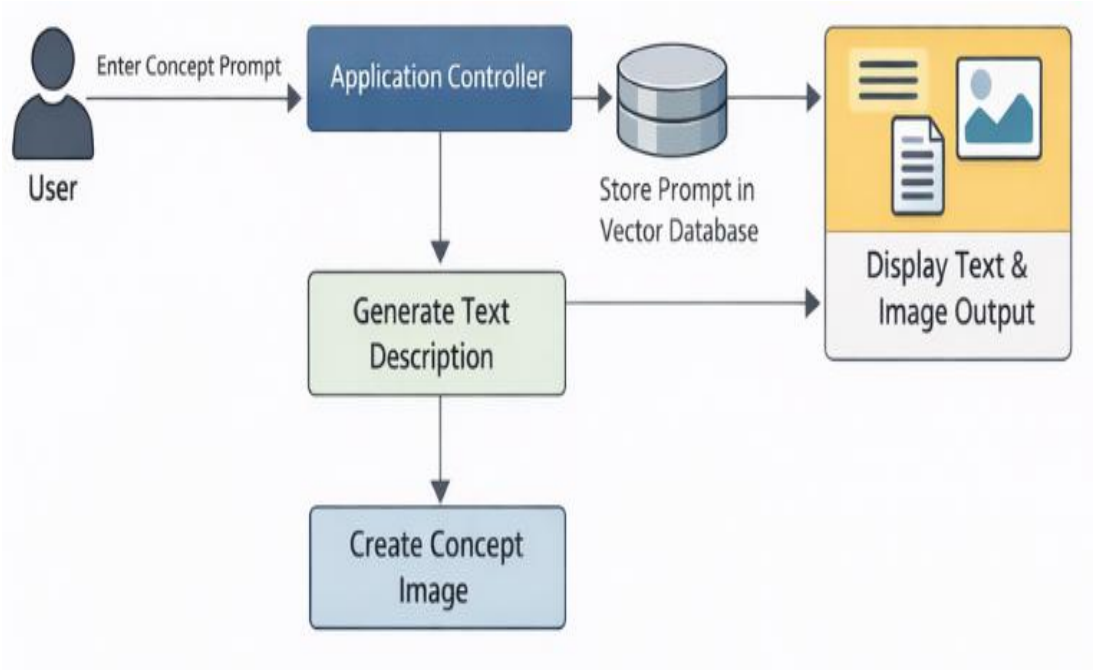
1.3. System overview - The Multimodal Hospitality Creator is a Python-based AI application with the system provides Streamlit Web UI, CLI Interface, HuggingFace hosted LLMs for narrative generation, Stable diffusion models for image generation, and ChromaDB vector database for semantic memory

2. System Design

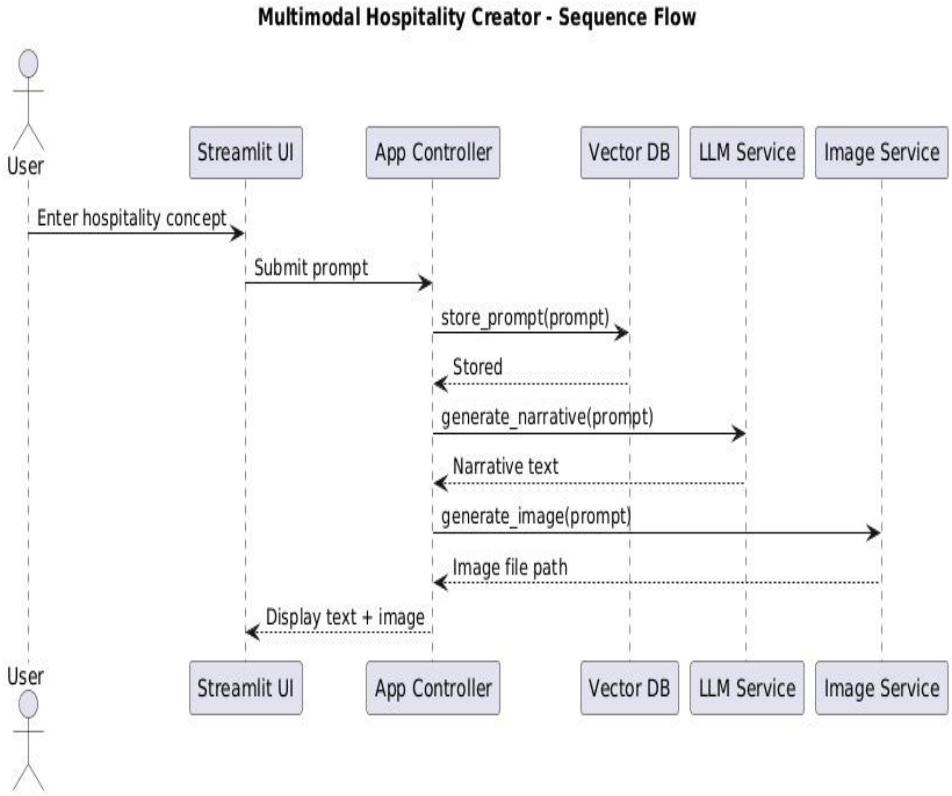
2.1. Application Design –



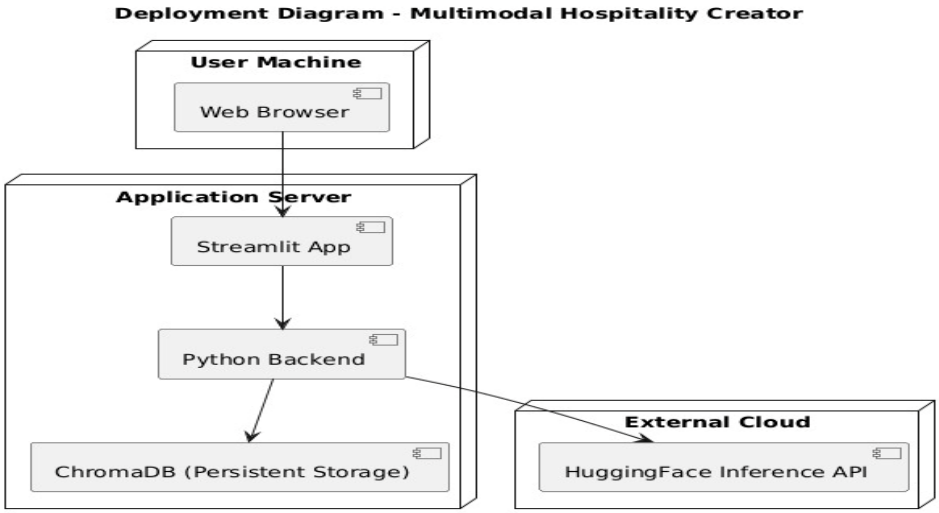
2.2. Process Flow –



2.3. Information Flow –



2.4. Components Design –



The system consists of the following major components, each designed with a clearly defined responsibility:

- **Application Controller:** Manages execution order, coordinates component interactions, and ensures consistent data usage.
- **Text Generation Component:** Interfaces with an external language model to generate a detailed textual description of the hospitality concept.
- **Image Generation Component:** Interacts with an image generation service to produce a visual representation aligned with the user prompt.
- **Vector Database Component:** Stores user prompts as vector embeddings to support similarity-based retrieval and future enhancements. The components are loosely coupled and interact only through the controller, improving system robustness.

2.5. Key Design Considerations – The design of the Multimodal Hospitality Concept Generator is driven by architectural and operational considerations that ensure the system remains robust, extensible, and suitable for future growth. These considerations influence how components are structured, how responsibilities are assigned, and how external dependencies are managed.

A central objective of the design is to maintain a clear separation of concerns,

ensuring that each component focuses on a single responsibility. This minimizes interdependencies and simplifies maintenance. The system is also designed to be extensible, allowing new features or interfaces to be introduced without requiring major architectural changes.

Another important consideration is the system's reliance on external AI services, which introduces cost and latency factors. The architecture therefore emphasizes controlled API usage and centralized orchestration to avoid redundant or unnecessary service calls

- **Modularity:** Each major function is isolated within its own component
- **Extensibility:** Support for future features such as web interfaces or recommendations
- **Maintainability:** Simple interactions and centralized coordination
- **Scalability:** Ability to evolve without redesigning core architecture
- **Cost Awareness:** Efficient use of external AI services

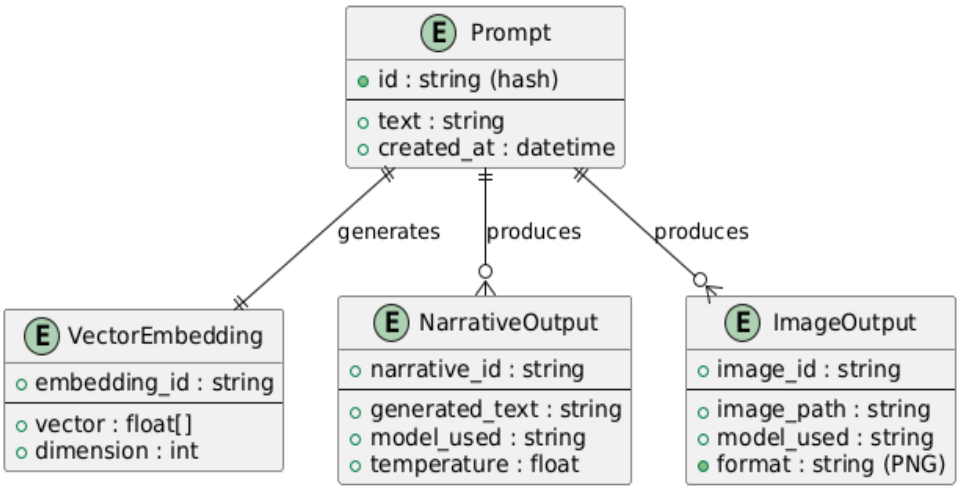
2.6. API Catalogue –

API CATEGORY	DESCRIPTION
Text Generation API	Generates detailed narrative descriptions for hospitality concepts
Image Generation API	Produces visual representations aligned with user prompts
Embedding API	Converts user prompts into vector embeddings for storage and retrieval

3. Data Design

3.1. Data Model – The system follows a **simple conceptual data model** that reflects its core functionality: generating and storing hospitality concepts based on user input. The **user prompt** is treated as the primary data entity, with all generated outputs linked to it.

Multimodal Hospitality Creator - Data Model Diagram



3.2. Data Access Mechanism – Data access is **centrally controlled** by the application controller

- All read/write operations go through the controller
- No direct access to the database by individual components
- Ensures consistency and data integrity

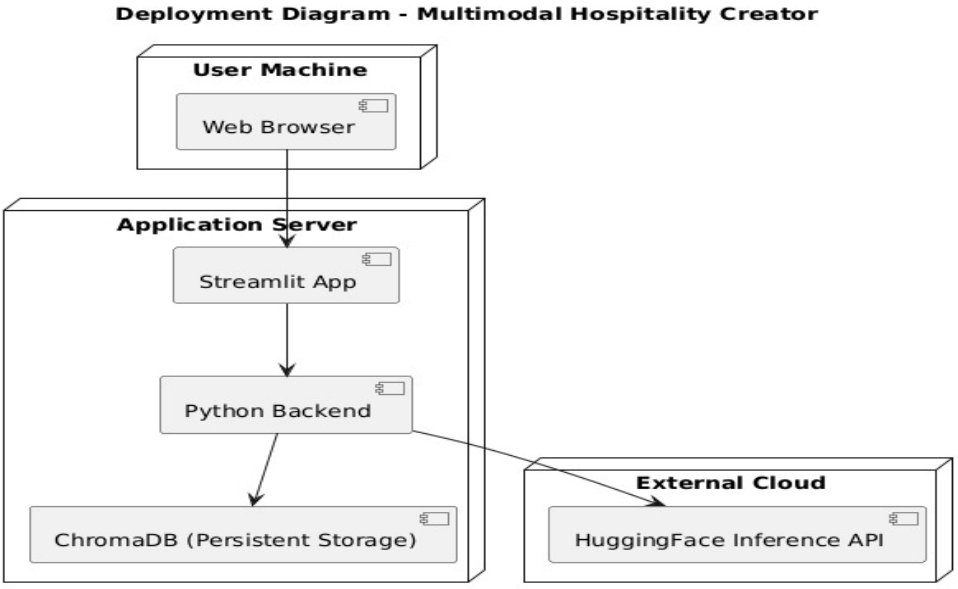
3.3. Data Retention Policies – The system retains data only when it adds functional value.

- User prompts
- Generated text outputs
- Image metadata
- Vector embeddings

3.4. Data Migration - The current system does not require active data migration, as it operates within a single environment and uses a single vector database for storing user prompts and embeddings.

- **Migration may be required in cases such as**:
- Transition from local to cloud-based vector storage
- Replacement or upgrade of the vector database backend
- Deployment across multiple environments

4. Interfaces



External Interfaces –

- **Text Generation Interface:** Used to send user prompts to a language model service and receive generated narratives.
- **Image Generation Interface**: Used to generate a visual representation of the hospitality concept.
- **Vector Database Interface**: Used to store and retrieve vector embeddings of user prompts via ChromaDB.

4.1. User Interfaces –

- The system currently operates through a Command-Line Interface (CLI).
- Users provide a hospitality concept prompt as input.
- Generated text and image output paths are displayed through the CLI.

5. State and Session Management

- No user sessions or login context
- Each run processes one prompt independently
- No shared runtime state between executions
- Prompt history is stored only at the data layer (vector database)

6. Caching

Caching is not implemented in the current version of the system. Each user request results in fresh processing and generation of outputs.

Current behaviour:

- Prompts are always processed anew.
- Text and image generation APIs are called for every execution.

Design consideration (from LLD & repo structure):

- Vector storage already enables **prompt reuse and similarity search**.
- Future caching may leverage stored prompts and embeddings to:
    - Reduce repeated API calls
    - Improve response time for similar inputs

Caching is intentionally deferred to keep the system simple and transparent at this stage.

7. Non-Functional Requirements

Non-functional requirements define the quality attributes of the system that influence how it operates rather than what functionality it provides. For the Multimodal Hospitality Concept Generator, these requirements focus primarily on security and performance, based on its current command-line execution model and reliance on external AI services.

7.1 Security Aspects

- The system follows a basic but appropriate security model aligned with its prototype-level and academic scope. Security considerations mainly address safe handling of external service access and protection of system configuration.

Security measures reflected in the current design:

- API keys for text and image generation services are managed internally and are not exposed to end users.
- The system does not collect or store sensitive personal user data.
- External AI services are accessed only through controlled application logic.
- No direct external access is provided to the vector database.

Design implications:

- Reduced risk of credential leakage.
- Limited attack surface due to absence of authentication or session handling.
- Suitable for local and academic execution environments.
- Future enhancements such as authentication or role-based access control may be introduced if the system is deployed as a multi-user application.

7.2 Performance Aspects

- System performance is primarily influenced by the response times of the external AI services used for text and image generation. The current design prioritizes clarity and correctness over optimization.

Current performance characteristics:

- Sequential execution of text generation followed by image generation.
- One prompt processed per application run.
- No parallel processing or background execution.
- Performance considerations based on design:
- Acceptable latency for prototype and academic usage.
- Predictable execution flow due to centralized orchestration.
- Modular structure allows future optimizations such as parallel calls or batching without architectural changes.
- Overall, the system provides sufficient performance for its intended scope while remaining flexible for future improvements.

---

8. References

1. ChromaDB Documentation
   Official documentation for ChromaDB, used as the vector database for storing and retrieving vector embeddings of user prompts. https://docs.trychroma.com

2. OpenAI / Large Language Model Documentation
   Reference documentation for large language models used for text generation in the system.https://platform.openai.com/docs (or equivalent LLM provider documentation used during implementation)

3. Image Generation Model Documentation
   Documentation for AI-based image generation models used to generate visual representations of hospitality concepts.
   https://platform.openai.com/docs/guides/images
   (or equivalent image generation service documentation)

4. Python Programming Language Documentation
Official Python documentation, as the system is implemented using Python.
https://docs.python.org/3/

5. Vector Embeddings and Similarity Search Concepts
Relevant technical resources explaining vector embeddings and similarity-based retrieval used in the system design. https://www.pinecone.io/learn/vector-embeddings/
(conceptual reference)

6. GitHub Repository – Multimodal Hospitality Concept Generator
Source code repository for the project implementation.
https://github.com/AkashGhosh3/Multimodal-hospitality-Creator