Project - High Level Design

on

Multimodal Education Creator

Course Name: GEN AI (Datagami Skill Based Course)

Institution Name: Medicaps University

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

Sr no	Student Name	Enrolment Number
1.	GAURAV BARVE	EN22CS301361
2.	HARSHVADAN TIWARI	EN22CS301412
3.	HARSHWARDHAN RAJPUT	EN22CS301415
4.	HASHIM SAIFY	EN22CS301417
5.	ISHA JAMINDAR	EN22CS301437
6.	TANISHQ VIJWAL	EN22CS3011022

Group Name: Group 08D4

Project Number: GAI-44

Industry Mentor Name:

University Mentor Name: Prof. Divya Kumawat

Academic Year: 2026

Table of Contents

1. Introduction

The **Multimodal Education Creator** is a Generative AI–powered application designed to transform textual educational prompts into rich learning materials. The system integrates Large Language Models (LLMs) with image generation models to produce both descriptive narratives and high-quality visual outputs such as flashcards.

This platform enhances concept visualization, improves learner engagement, and supports educators in creating professional educational content efficiently.

1.1 Scope of the Document

This document describes:

- System architecture and design
- Data flow and processing pipeline
- API integrations
- Non-functional requirements
- Security and performance considerations

It serves as the technical blueprint for development, deployment, and maintenance of the Multimodal Education Creator.

1.2 Intended Audience

This document is intended for:

- Software developers
- AI/ML engineers
- System architects
- Project reviewers
- Academic evaluators
- DevOps engineers

1.3 System Overview

The system accepts a **user text prompt** describing an educational concept and generates:

- AI-generated explanation (via LLM)
- Concept visualization image
- Flashcards for learning
- Structured educational content

Core Technologies

- Python backend
- LLM API (Gemini/OpenAI/etc.)
- Image Generation API
- Vector Database (FAISS/Pinecone/Chroma)
- Frontend (Streamlit/React)

2. System Design

2.1 Application Design

The application follows a **modular microservice-inspired architecture**.

Layers

1. **Presentation Layer**
    - User interface
    - Input form
    - Results display
2. **Application Layer**
    - Prompt processing
    - Workflow orchestration
    - Flashcard generator

3. **AI Services Layer**
    - LLM service
    - Image generation service
    - Embedding service
4. **Data Layer**
    - Vector database
    - Metadata storage
    - Cache

2.2 Process Flow

Step-by-Step Workflow

1. User enters educational prompt
2. System validates input
3. Prompt sent to LLM API
4. LLM generates explanation
5. Image prompt derived
6. Image generation API called
7. Flashcards generated
8. Embeddings stored in vector DB
9. Results displayed to user

2.3 Information Flow

User → Frontend → Backend API

→ LLM Service → Text Output

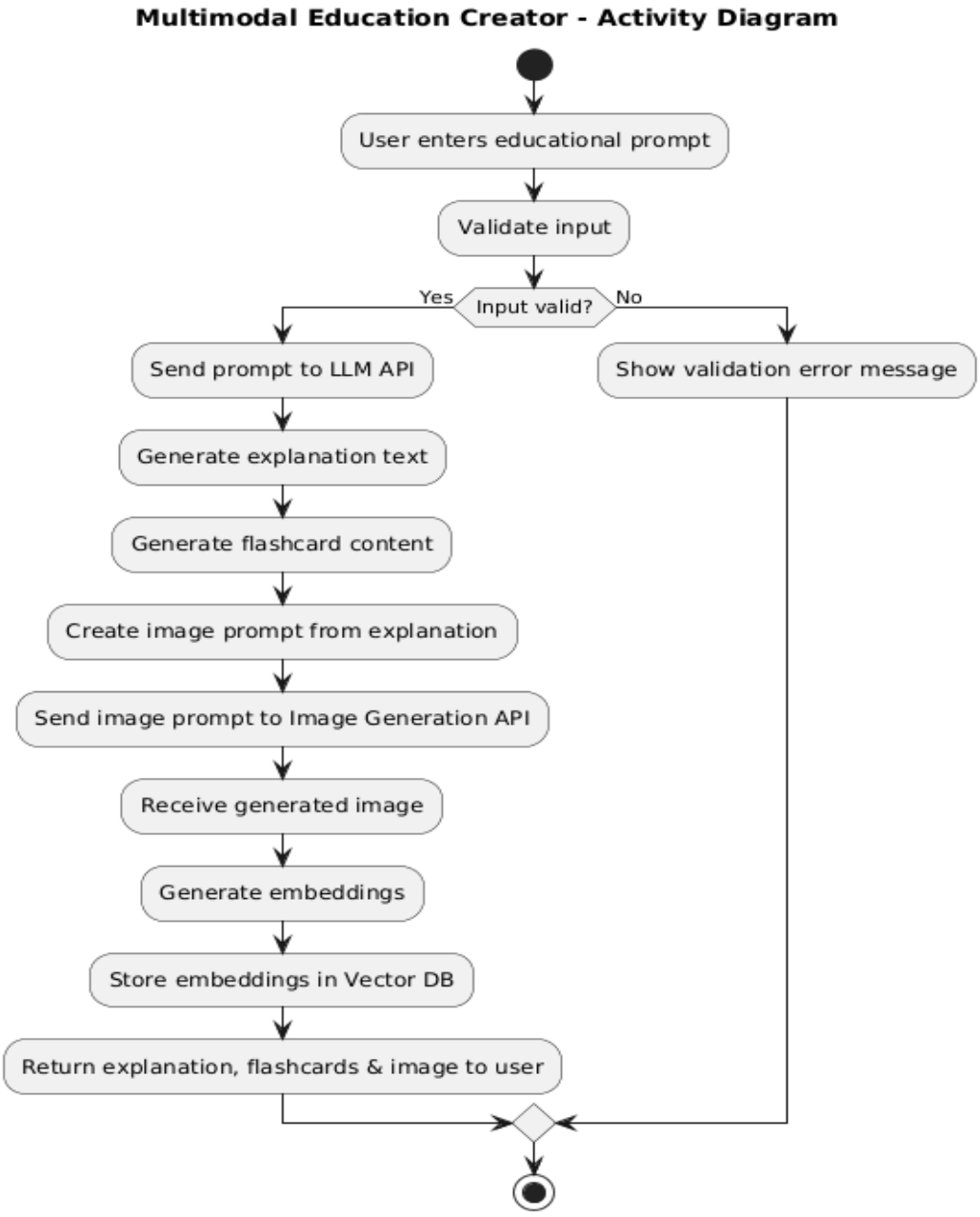
→ Image Service → Visual Output

→ Vector DB → Storage/Retrieval

→ Frontend → User Display

Data Types

- Input text
- Generated narrative
- Generated images
- Embeddings
- Metadata



2.4 Components Design

Major Components

1. User Interface

- Prompt input
- Results viewer
- Flashcard display

2. Prompt Processor

- Cleans input
- Enhances prompt
- Routes requests

3. LLM Module

- Concept explanation
- Flashcard text generation
- Summary generation

4. Image Generation Module

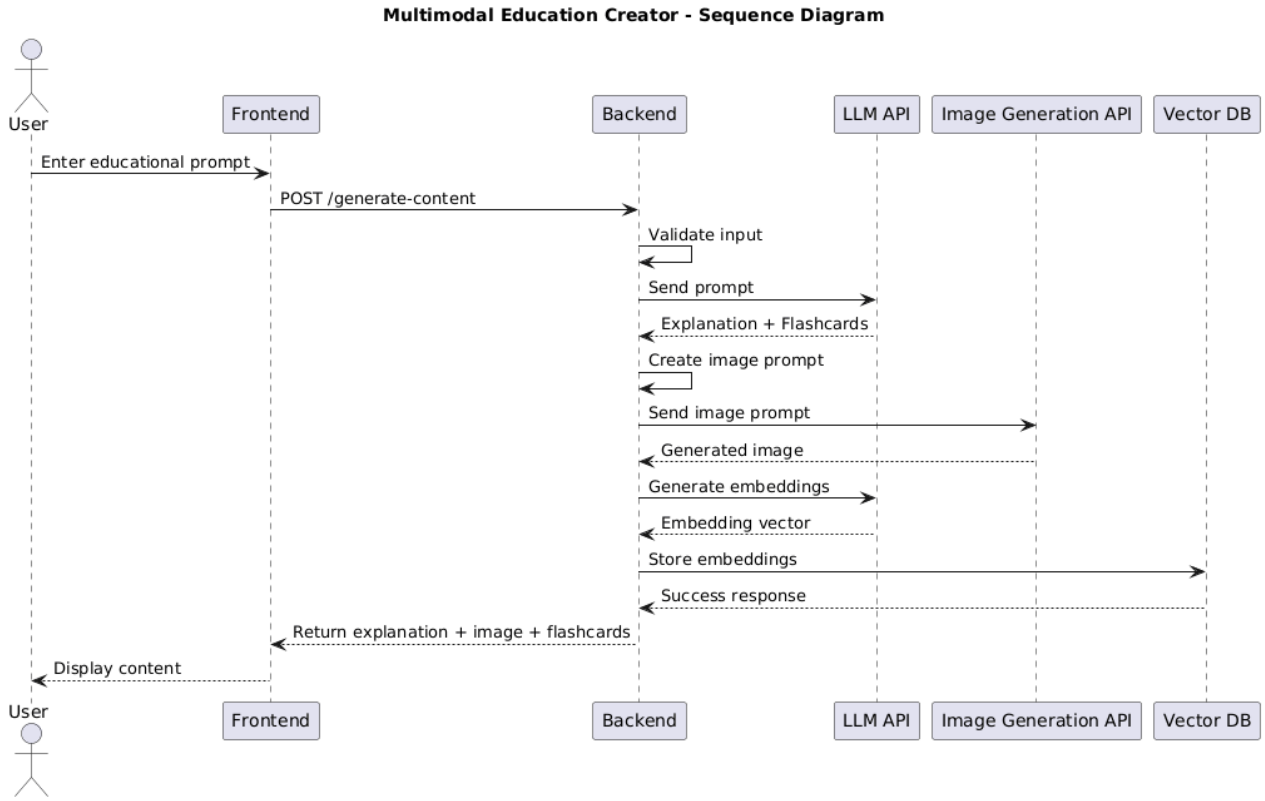
- Converts concept → visual prompt
- Calls diffusion/image API
- Returns high-quality image

5. Vector Database

- Stores embeddings
- Enables semantic search
- Supports retrieval-augmented generation (RAG)

6. Orchestrator

- Controls workflow
- Handles retries
- Manages API calls



2.5 Key Design Considerations

- Scalability for multiple users
- Modular architecture
- Low latency generation
- Cost optimization for API calls
- Prompt quality control
- Fault tolerance
- Extensibility for new models

2.6 API Catalogue

LLM API

Field	Description
Endpoint	/generate-text

Method	POST
Input	User prompt
Output	Explanation text

Image Generation API

Field	**Description**
Endpoint	/generate-image
Method	POST
Input	Image prompt
Output	Generated image

Embedding API

Field	**Description**
Endpoint	/embed
Method	POST
Input	Text
Output	Vector embedding

Flashcard API

Field	**Description**
Endpoint	/generate-flashcards
Method	POST
Output	Q&A flashcards

3. Data Design

3.1 Data Model

Entities

UserRequest

- request_id
- prompt
- timestamp

GeneratedContent

- content_id
- explanation_text
- image_url
- flashcards

Embeddings

- vector_id
- embedding_vector
- metadata

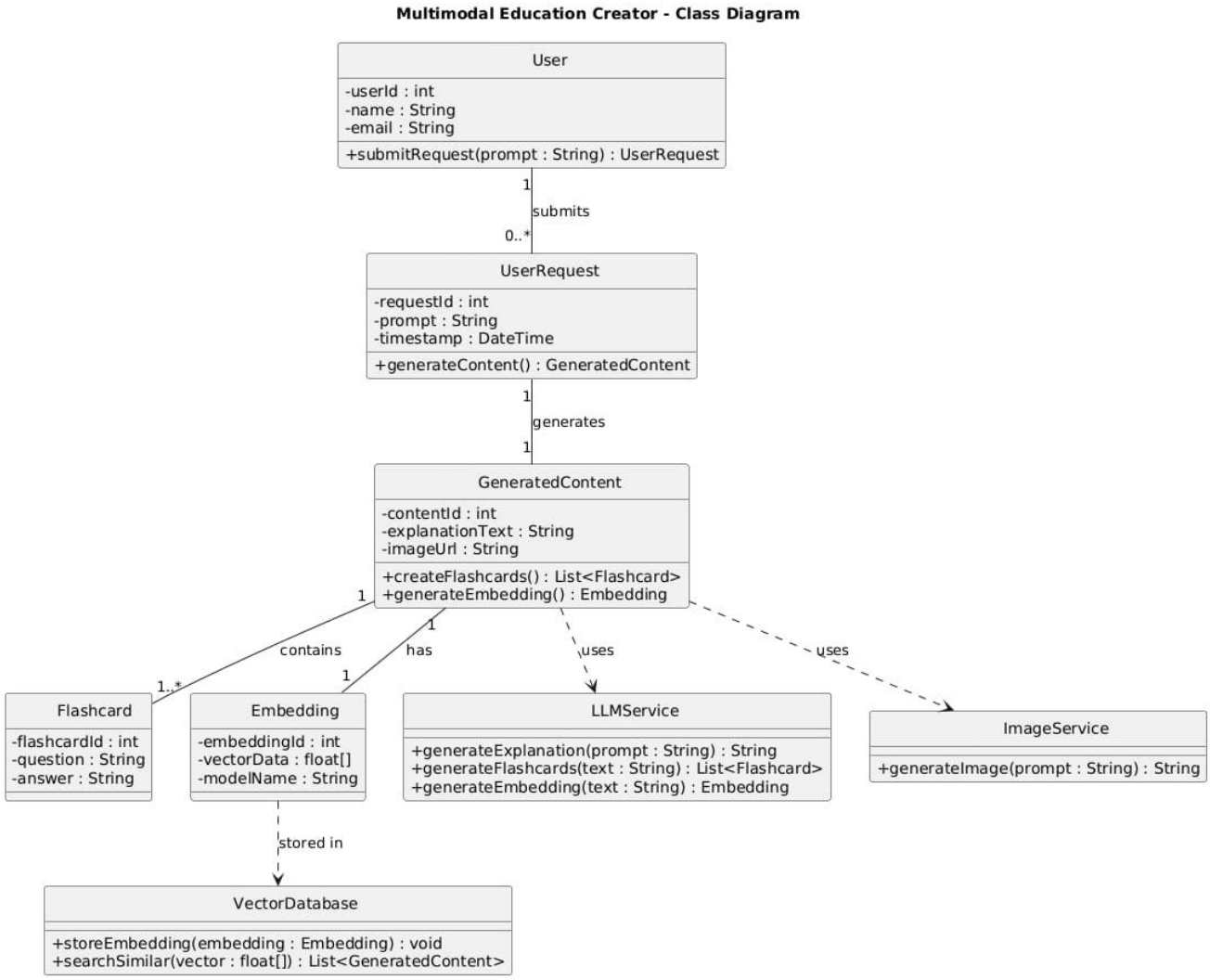
3.2 Data Access Mechanism

- REST APIs for CRUD operations
- Vector similarity search
- Indexed metadata queries
- Batch embedding storage

Technologies

- FAISS / ChromaDB
- SQLite / MongoDB

- Python ORM



3.3 Data Retention Policies

- User prompts stored for analytics
- Generated images stored temporarily
- Embeddings stored long-term
- Logs retained for debugging

Retention Rules

- Images: 30 days

- Logs: 90 days
- Embeddings: persistent

3.4 Data Migration

Migration strategies include:

- Schema versioning
- Backup before migration
- Rolling updates
- Vector DB re-indexing

4. Interfaces

External Interfaces

- LLM API
- Image Generation API
- Vector DB service

Internal Interfaces

- Frontend ↔ Backend REST APIs
- Backend ↔ AI services
- Backend ↔ Cache

5. State and Session Management

The system uses **stateless backend design** with optional session tracking.

Session Strategy

- Session ID per user
- Stored in Redis/local memory
- Maintains conversation context

- Supports multi-turn queries

6. Caching

Caching improves performance and reduces API cost.

Cache Layers

- Prompt response cache
- Image cache
- Embedding cache

Tools

- Redis
- In-memory cache
- CDN (optional)

7. Non-Functional Requirements

7.1 Security Aspects

- API key protection
- HTTPS communication
- Input validation
- Rate limiting
- Secure environment variables
- Role-based access (optional)

Risks Mitigated

- Prompt injection
- API abuse

- Data leakage
- Unauthorized access

7.2 Performance Aspects

Targets

- Response time < 5 seconds
- Concurrent users supported
- Scalable horizontally

Optimization Techniques

- Async API calls
- Response caching
- Batch embeddings
- Lazy image loading
- Model selection based on cost/speed

8. References

[1] Google Cloud, *Generative AI Architecture Framework*. Available: https://cloud.google.com/architecture/generative-ai

[2] Pinecone Systems, *Vector Database Documentation*. Available: https://docs.pinecone.io

[3] Chroma, *Chroma Documentation*. Available: https://docs.trychroma.com

[4] OpenAI, *OpenAI API Documentation*. Available: https://platform.openai.com/docs

[5] Google AI, *Gemini API Documentation*. Available: https://ai.google.dev

[6] Jonathan Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[7] FastAPI, *FastAPI Documentation*. Available: https://fastapi.tiangolo.com

[8] Facebook AI Research, *FAISS Documentation*. Available: https://faiss.ai