Multimodal Education Creator

Course Name: GEN AI (Datagami Skill Based Course)

Institution Name: Medicaps University – Datagami Skill Based Course

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 08 D4

Project Number: GAI-44

Industry Mentor Name:

University Mentor Name: Prof. Divya Kumawat

Academic Year: 2026

TABLE OF CONTENT

1. Problem Statement & Objectives

1.1 Problem Statement

In the modern digital learning ecosystem, students and educators increasingly rely on online platforms for academic content. However, most existing platforms provide static, pre-written material that does not dynamically adapt to individual learning needs, grade levels, or specific conceptual queries. Learners often struggle to instantly generate structured study material, simplified explanations, revision flashcards, and visual representations of complex concepts from a single prompt.

Traditional content creation methods are time-consuming and require manual effort from educators to prepare notes, summaries, and visual aids. Moreover, available AI tools typically generate only textual outputs without integrating multimodal learning elem ents such as concept diagrams or flashcards in a structured academic format.

There is a clear need for an AI-powered educational assistant capable of:

- Generating structured academic content (overview, key points, examples, summaries)
- Creating explanations tailored to different learning levels (School, College, Professional)
- Producing relevant AI-generated visual diagrams to enhance concept visualization
- Automatically generating revision-ready flashcards
- Providing an interactive, web-based user interface for seamless learning

Additionally, deploying AI-driven applications introduces technical and operational challenges. Large Language Models (LLMs) and image generation models require API integrations, efficient orchestration, caching mechanisms, and vector storage for embeddings. When deployed on cloud platforms—especially under free-tier constraints—performance limitations such as cold-start latency, API response delays, and resource restrictions must be managed effectively.

Therefore, the core problem addressed by this project is the development of a scalable, multimodal, AI-powered educational content generation system that delivers structured textual explanations, visual outputs, and interactive learning resources efficiently within real-world cloud deployment constraints.

1.2 Project Objectives

The primary objective of the Multimodal Education Creator project is to design and develop a scalable, AI-powered educational platform capable of generating structured, multimodal learning content dynamically from user prompts. The system aims to combine text generation, visual content creation, and semantic storage into a unified and interactive web-based solution.

The detailed objectives of the project are as follows:

a. To Build an AI-Powered Educational Content Generation Platform

Develop a web-based application that accepts academic topics as input and automatically generates structured educational content including overviews, key points, real-world examples, summaries, and flashcards. The platform should support multiple learning levels (School, College, Professional) to adapt content depth and vocabulary accordingly.

b. To Integrate Large Language Models (LLMs) for Dynamic Text Generation

Incorporate external LLM APIs to generate accurate, structured, and context-aware educational explanations. The system must:

- Validate whether the input topic is academic in nature
- Format responses in structured JSON
- Ensure clarity, correctness, and level-based customization
- Generate flashcards automatically from the explanation

c. To Integrate Image Generation Models for Visual Learning

Implement integration with an image generation API to create high-quality educational diagrams related to the input concept. The generated images should:

- Visually represent core concepts
- Avoid embedded text for clarity
- Enhance conceptual understanding
- Support multimodal learning

d. To Deploy Backend Services on Cloud Infrastructure (Render)

Host the backend API using a cloud platform (Render) to enable real-time access over the internet. The deployment should:

- Expose REST endpoints for content and image generation
- Handle CORS for frontend-backend communication
- Ensure stability under limited free-tier resources

e. To Create an Interactive and Responsive Frontend Using Streamlit

Design a user-friendly and visually appealing frontend using Streamlit that:

- Allows users to enter prompts easily
- Displays structured explanations in tabs
- Shows generated diagrams
- Presents flashcards interactively
- Maintains session-based chat history

f. To Implement a Scalable and Modular Architecture

Design the system using a modular, microservice-inspired architecture consisting of:

- Presentation Layer (Frontend UI)
- Backend API Layer (FastAPI)
- AI Services Layer (LLM + Image API)
- Vector Database Layer (Embeddings Storage)

This modular structure ensures easy future integration of:

- Additional AI models
- Authentication systems
- Analytics dashboards
- Retrieval-Augmented Generation (RAG) pipelines

g. To Implement Vector Embeddings and Semantic Search

Integrate a lightweight vector database using SentenceTransformers to:

- Generate embeddings of explanations
- Store vectors persistently
- Enable semantic similarity search
- Support future RAG-based improvements

h. To Optimize Performance Under Free-Tier Hosting Constraints

Address challenges such as:

- Cold start latency
- API response delays
- Limited compute and memory
- Rate limits on external AI APIs

Implement caching mechanisms (LRU caching), optimized API calls, and lightweight vector storage to ensure acceptable performance within cost constraints.

1.3 Scope of the Project

The scope of the *Multimodal Education Creator* project defines the functional capabilities and technical boundaries of the system. It focuses on building a multimodal AI-powered platform that generates structured textual and visual educational content through a web-based interface. The current scope emphasizes modular design, cloud deployment, AI integration, and performance optimization within free-tier hosting constraints, while excluding advanced enterprise-level features.

- **AI-Based Text Generation** – The system integrates Large Language Model APIs to generate structured academic content such as overview, key points, real-world examples, flashcards, and summaries. The content is dynamically formatted in JSON and customized according to selected grade levels to ensure appropriate depth and clarity.
- **AI-Based Image Generation** – The application connects with an external image generation API to create concept-based educational diagrams. These diagrams enhance visual understanding and are rendered dynamically within the frontend.
- **Cloud-Based Backend Deployment** – The FastAPI backend is deployed on Render, enabling real-time internet accessibility. REST endpoints are exposed for content and image generation, with environment-based configuration for secure API key management.
- **Interactive Web-Based Frontend** – A Streamlit-based user interface provides a responsive and user-friendly environment where users can enter prompts, select learning levels, and view generated explanations, flashcards, and diagrams in a structured layout.
- **API-Based Communication –** The frontend and backend communicate through RESTful APIs, ensuring separation of concerns and maintainability of system components.
- **Modular & Scalable Architecture** – The system is structured into distinct layers including Presentation, Backend API, AI Services, and Vector Storage layers, allowing easy extension and integration of additional features in the future.
- **Vector Embeddings & Semantic Search** – A lightweight vector database using SentenceTransformers and NumPy is implemented to store embeddings of generated explanations, enabling semantic similarity search and supporting future RAG enhancements.
- **Caching & Performance Optimization** – LRU caching mechanisms are implemented to reduce redundant API calls and manage latency issues associated with free-tier cloud hosting and external AI APIs.
- **Session-Based Interaction Handling** – Temporary in-memory session tracking is implemented to maintain conversation context during a user's interaction with the system.

2. Proposed Solution

The proposed solution is **EduGen AI**, an AI-driven multimodal educational assistant designed to automatically generate structured academic content and visual learning materials from a single user prompt. The system is built to bridge the gap between traditional static learning resources and dynamic AI-powered content generation by combining advanced language models, image generation models, and a responsive web-based interface.

EduGen AI integrates a Large Language Model (*gpt-oss-20b*) for generating structured textual explanations and an image generation model (*imagen-4*) for producing high-quality educational diagrams. These AI services work together to provide both descriptive and visual outputs, supporting multimodal learning and improving conceptual understanding.

The platform operates on a client-server architecture. The Streamlit-based frontend acts as the client interface where users enter academic topics and select grade levels. This request is sent to the backend API hosted on Render, which acts as the server responsible for processing and orchestrating AI service calls.

Within the backend, the request is routed to two primary services. The LLM Service generates structured educational content including overview, key points, examples, flashcards, and summary. Simultaneously, the Image Service generates a concept-based visual diagram related to the same topic. Once both services complete processing, the structured response is returned to the frontend and displayed in an organized, interactive layout.

Since the backend is deployed on Render under a free-tier plan, the first request after a period of inactivity may experience a delay due to cold start behavior, where the server instance spins up before processing requests. Despite this initial latency, subsequent requests are processed more efficiently due to active server state and implemented caching mechanisms.

Overall, EduGen AI provides a scalable, modular, and cloud-deployed solution that integrates text intelligence, visual generation, and interactive web technologies into a unified educational platform.

2.1 Key Features

- AI-based structured text content generation
- AI-generated educational diagram creation
- Grade-level adaptive explanations
- Prompt validation for academic relevance
- Cloud deployment on Render platform
- Streamlit-based interactive frontend interface
- LRU caching for performance optimization
- Lightweight vector embedding storage

- Free-tier cloud hosting optimization
- Scalable and extensible system design

2.2 Overall Architecture / Workflow

Architecture Explanation

1. Frontend Layer (Streamlit)
    - Accepts user input
    - Sends API requests to backend
    - Displays generated text and images
2. Backend Layer (Render Cloud Deployment)
    - Handles API requests
    - Routes requests to appropriate services
    - Returns generated responses
3. Service Layer
    - LLM Service → Calls gpt-oss-20b
    - Image Service → Calls imagen-4
    - Vector DB → Handles embeddings (if used)
    - Prompts Module → Stores structured prompts
4. Model Layer
    - Text Model: gpt-oss-20b
    - Image Model: imagen-4

Workflow Steps

- The user submits an academic prompt through the Streamlit-based frontend interface, optionally selecting a grade level.
- The frontend sends an HTTP POST request to the backend API hosted on Render.
- The backend receives the request and validates it using predefined schemas to ensure proper structure and required fields.
- Once validated, the request is routed internally to two service modules: llm_service.py for structured text generation and image_service.py for educational image generation.
- The LLM model (gpt-oss-20b) processes the prompt and generates structured educational content in JSON format.
- The image generation model (imagen-4) processes a derived visual prompt and produces a high-quality educational diagram.
- The backend aggregates the generated textual and visual outputs into a structured response.
- The response is sent back to the Streamlit frontend.
- The frontend renders the explanation, flashcards, summary, and generated image in an organized and interactive layout for the user.

This workflow ensures modular processing, clean API communication, and seamless multimodal content delivery to the end user.

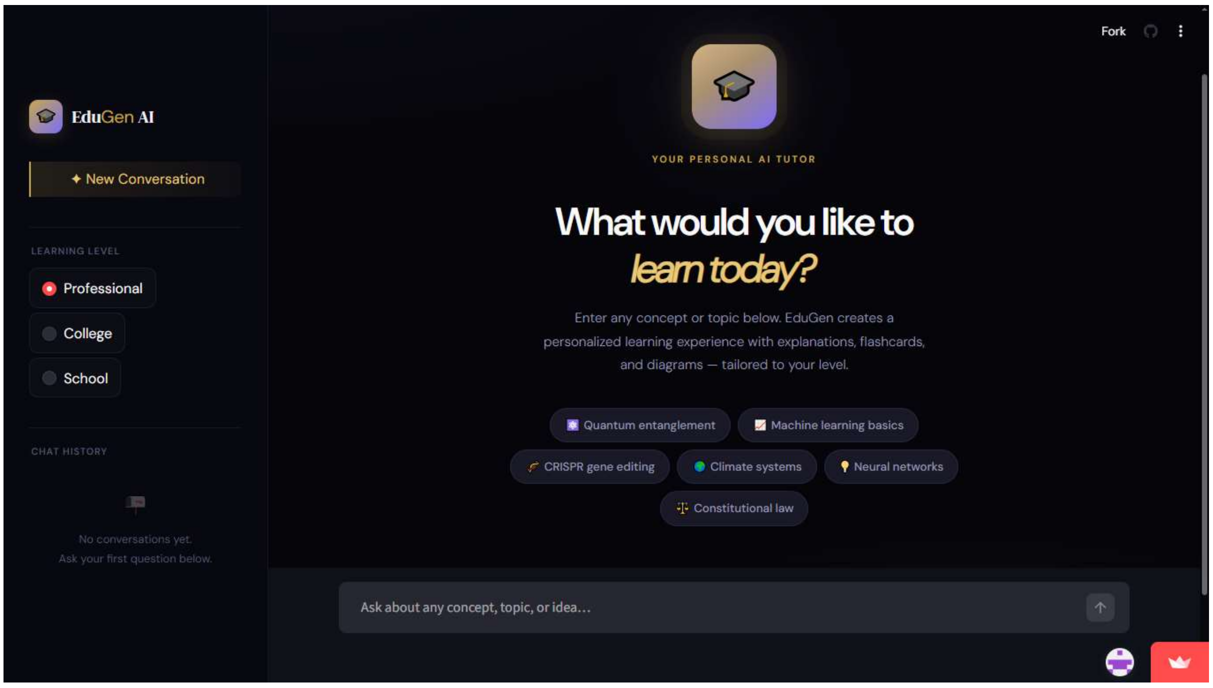
2.3 Tools & Technologies Used

Category	Technology	Description
Programming Language	**Python**	Core language used for frontend and backend development, AI integration, and API handling.
Frontend Framework	**Streamlit**	Used to build the interactive web interface for user input and display of generated text and images.
Backend Framework	**FastAPI**	Handles REST API requests, input validation, and routing to AI services.
Cloud Platform	**Render (Free Tier)**	Hosts backend services. Initial response delay occurs due to cold start in free-tier hosting.
Text Generation Model	**gpt-oss-20b**	Large Language Model used for generating structured educational content.
Image Generation Model	**imagen-4**	AI model used to generate images from textual prompts.
API Communication	**REST APIs (HTTP)**	Enables communication between Streamlit frontend and Render backend.
Dependency Management	**requirements.txt**	Manages project libraries and dependencies.

3. Results & Output

3.1 Screenshots / Outputs

Home page

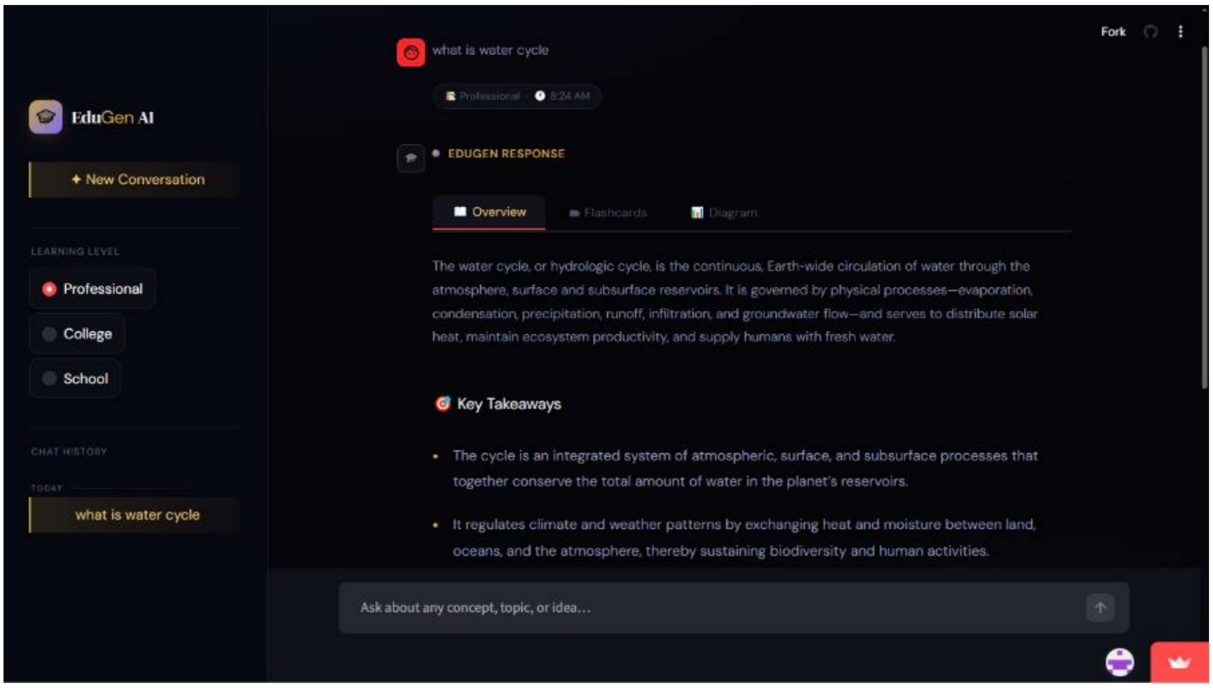


No conversations yet.
Ask your first question below.

What would you like to *learn today?*

Enter any concept or topic below. EduGen creates a personalized learning experience with explanations, flashcards, and diagrams — tailored to your level.

What is water Cycle?



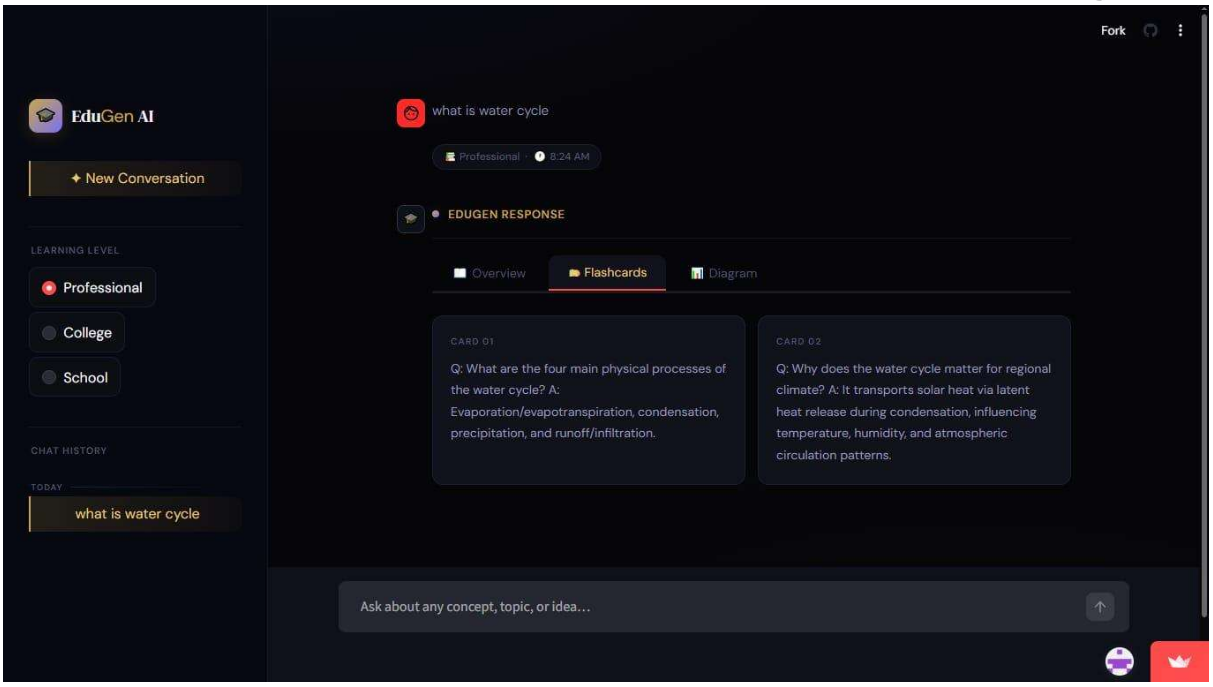
what is water cycle

The water cycle, or hydrologic cycle, is the continuous, Earth-wide circulation of water through the atmosphere, surface and subsurface reservoirs. It is governed by physical processes—evaporation, condensation, precipitation, runoff, infiltration, and groundwater flow—and serves to distribute solar heat, maintain ecosystem productivity, and supply humans with fresh water.

Key Takeaways

- The cycle is an integrated system of atmospheric, surface, and subsurface processes that together conserve the total amount of water in the planet's reservoirs.
- It regulates climate and weather patterns by exchanging heat and moisture between land, oceans, and the atmosphere, thereby sustaining biodiversity and human activities.

Generated Flashcard


Fork

EduGen AI

✦ New Conversation

LEARNING LEVEL

Professional

College

School

CHAT HISTORY

TODAY

what is water cycle

what is water cycle

Professional · 8:24 AM

EDUGEN RESPONSE

Overview | Flashcards | Diagram

CARD 01

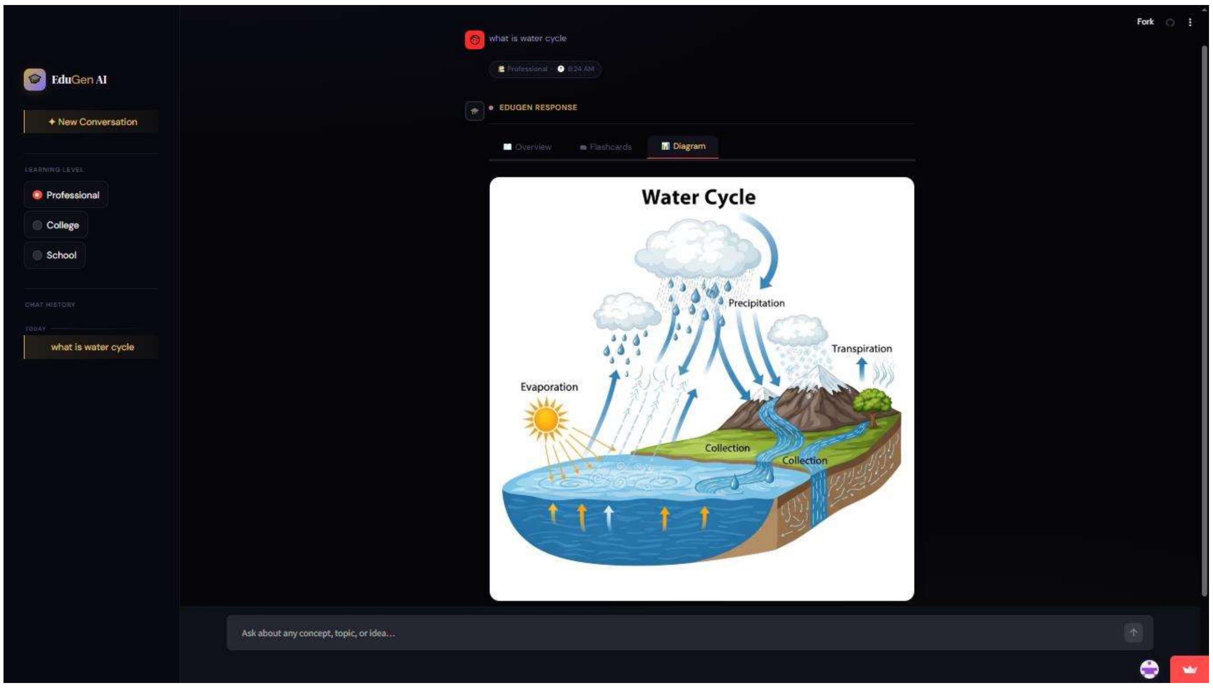
Q: What are the four main physical processes of the water cycle? A: Evaporation/evapotranspiration, condensation, precipitation, and runoff/infiltration.

CARD 02

Q: Why does the water cycle matter for regional climate? A: It transports solar heat via latent heat release during condensation, influencing temperature, humidity, and atmospheric circulation patterns.

Ask about any concept, topic, or idea...

Generated Image


Fork

EduGen AI

✦ New Conversation

LEARNING LEVEL

Professional

College

School

CHAT HISTORY

TODAY

what is water cycle

what is water cycle

Professional · 8:24 AM

EDUGEN RESPONSE

Overview | Flashcards | Diagram

Ask about any concept, topic, or idea...

3.2 Reports / Dashboards / Models

- AI-Generated Text Responses – Structured educational content generated by the Large Language Model, including overview, key points, examples, flashcards, and summary in JSON format.
- AI-Generated Educational Images – Concept-based visual diagrams generated through the integrated image generation model to enhance visual learning.
- Interactive Output Rendering – Streamlit-based dynamic display of explanations, flashcards, and images in an organized tabbed interface.

- Real-Time API Responses – Instant processing and response handling through FastAPI backend endpoints for content and image generation.

3.3 Key Outcomes

- Successfully integrated Large Language Model and Image Generation Model to produce multimodal educational content from a single prompt.
- Designed and implemented a modular backend architecture using FastAPI, ensuring clean separation of services and maintainability.
- Deployed the AI-powered backend on Render cloud platform with proper API exposure and environment configuration.
- Developed a fully functional and interactive Streamlit frontend for seamless user interaction and structured content display.
- Optimized performance using caching mechanisms and efficient API handling to manage free-tier cloud hosting limitations.

- Achieved a complete end-to-end AI-powered web application capable of generating structured text, flashcards, and visual diagrams in real time.

4. Conclusion

The *EduGen AI – Multimodal Education Creator* project successfully demonstrates the practical integration of Large Language Models and image generation models within a cloud-deployed web application. The system effectively combines AI-powered text generation and visual diagram creation to produce structured, multimodal educational content from a single user prompt. This integration highlights how generative AI can be leveraged to enhance digital learning experiences by delivering both conceptual explanations and visual understanding in real time.

By utilizing a Streamlit-based frontend and a FastAPI backend hosted on Render, the project implements a clean client-server architecture that enables seamless communication between user interface and AI services. The modular backend design ensures separation of concerns across prompt processing, LLM interaction, image generation, and embedding storage, making the system maintainable and extensible.

Although the application is deployed on Render's free-tier infrastructure, which introduces cold-start latency after periods of inactivity, the system performs efficiently once initialized. Performance optimization techniques such as caching and structured API handling help mitigate resource limitations and reduce redundant external API calls.

Through the development of this project, significant practical knowledge was gained in AI model integration, REST API architecture, cloud deployment strategies, modular software design principles, frontend-backend communication workflows, and handling real-world deployment constraints such as limited compute resources and API latency.

Overall, the project showcases a complete end-to-end implementation of an AI-powered educational platform and demonstrates how generative AI technologies can be applied effectively in real-world web-based learning systems.

5. Future Scope & Enhancements

- **User Authentication & Session Management** – Implement secure login, registration, and personalized user sessions to allow saved history and customized learning experiences.
- **Persistent Database Integration** – Integrate MongoDB or PostgreSQL for structured and long-term storage of user data, prompts, and generated content.
- **Performance Optimization** – Upgrade to a paid cloud hosting plan to eliminate cold start delays and improve response time and reliability.
- **Containerization with Docker** – Package the application using Docker to ensure consistent deployment environments and easier scalability.
- **Enhanced Security Measures** – Add API authentication, token-based access, and rate limiting to protect backend services from misuse.
- **Advanced Analytics Dashboard** – Develop a dashboard to track usage statistics, popular topics, and system performance metrics.
- **Multi-Model Support** – Allow users to select different LLMs or image models based on quality, speed, or cost preferences.
- **Improved Caching Mechanism** – Implement advanced caching strategies to reduce redundant API calls and enhance system efficiency.
- **Monitoring & Logging Tools** – Integrate monitoring systems to track API latency, error rates, and infrastructure health.
- **Mobile Optimization** – Improve UI responsiveness or develop a dedicated mobile-friendly version for better accessibility across devices.