Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	27 June 2025
Team ID	LTVIP2025TMID30136
Project Name	EduTutor AI: Personalized Learning with Generative AI and LMS Integration
Maximum Marks	4 Marks

Technical Architecture:

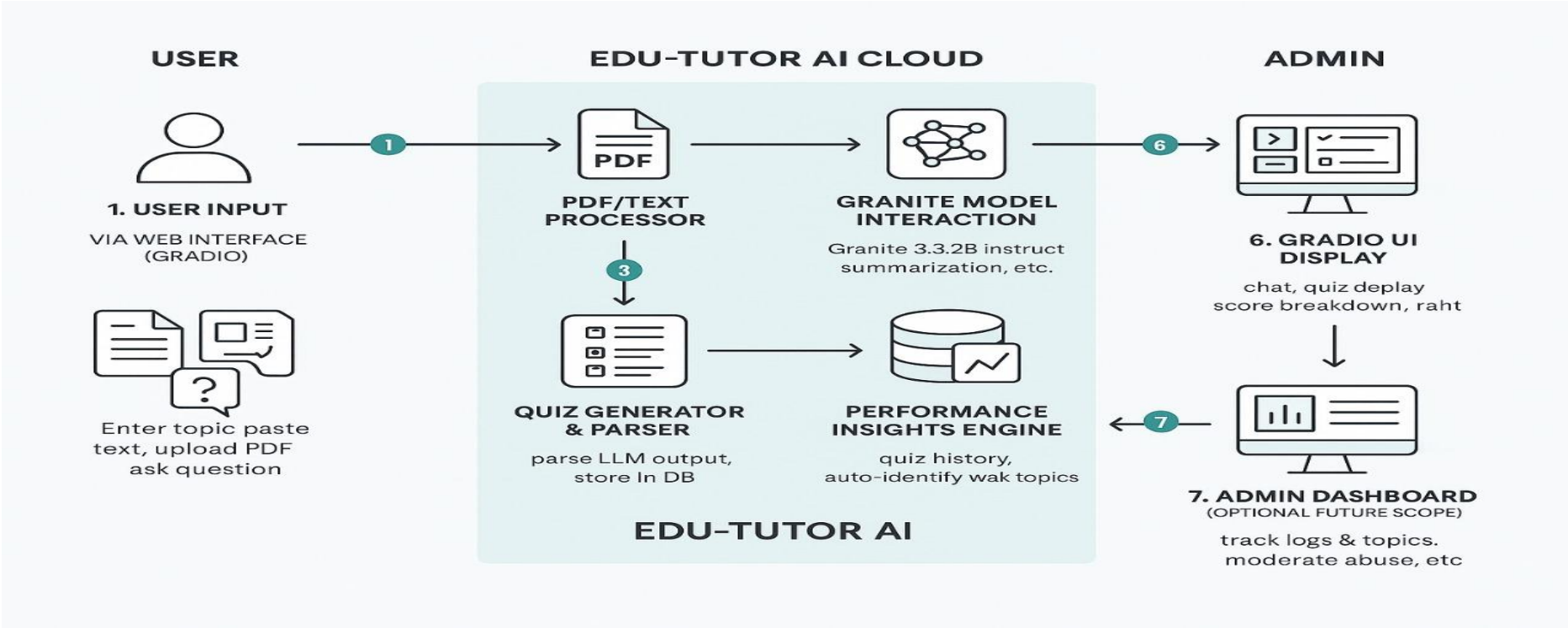

Table-1: Application Components:

S.No	Component	Description	Technology
1.	User Interface	How the user interacts with the application, including chat, various text processing forms, and interactive quiz displays.	Web UI, Gradio, HTML/CSS/JavaScript (underlying Gradio)
2.	Application Logic-1 (Core NLP Functions)	Logic for processing natural language requests such as chatbot responses, text summarization, text refinement, word meaning, sentence translation, and quiz question generation.	Python, Hugging Face Transformers, PyTorch, Accelerate
3.	Application Logic-2(PDF Processing)	Logic for extracting text content from uploaded PDF documents, primarily for generating quizzes based on document content.	Python, PyPDF2
4.	Application Logic-3 (Quiz Management)	Logic for handling quiz submission, scoring user answers, generating detailed feedback, and recording quiz performance results.	Python
5.	Database	Stores quiz results, including timestamp, topic, difficulty, score, and total number of questions, for performance tracking.	SQLite (file-based relational database)
6.	Cloud Database	Not Applicable	Not Applicable
7.	File Storage	Manages the storage of uploaded PDF files for quiz generation and the SQLite database file (edututor.db).	Local Filesystem
8.	External API-1	Not explicitly used for external services in the provided Python code.	Not Applicable
9.	External API-2	Not Applicable	Not Applicable
10.	Machine Learning Model	Performs various Natural Language Processing (NLP) tasks including text generation, summarization, translation, and question answering for quiz creation.	Causal Language Model (LLM) loaded via Hugging Face Transformers (AutoModelForCausalLM)
11.	Infrastructure (Server / Cloud)	The application is designed for local deployment and execution, providing a self-contained environment.	Local Server Configuration (Python runtime environment)

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List of key open-source libraries and frameworks utilized in the project.	Gradio, Hugging Face Transformers, PyTorch, Accelerate, PyPDF2, SQLite3
2.	Security Implementations	Standard Python and operating system-level security applies. No advanced security features (e.g., authentication, encryption for data at rest/in transit, explicit access controls) are implemented in the provided code.	Basic (OS-level security, Python's built-in security features)
3.	Scalable Architecture	The current architecture is a monolithic application running on a single instance. Scalability would involve vertical scaling (more resources to the server) or horizontal scaling by deploying multiple containerized instances behind a load balancer.	Monolithic application
4.	Availability	Availability is tied to the uptime of the single running application instance. High availability would require deploying redundant instances and using load balancing, which is not part of the current code.	Single Instance Deployment
5.	Performance	Performance for LLM inference is optimized through the use of accelerate and torch, allowing leverage of hardware accelerators (GPUs) if available. Overall performance depends on the chosen LLM's size and the host machine's specifications.	Accelerate, PyTorch (GPU acceleration)