**EduTutor AI: Personalized Learning with Generative AI and LMS Integration**

**Project Documentation**

**1. Introduction**

* **Project Title**: EduTutor AI: Personalized Learning with Generative AI and LMS Integration
* **Team member**: Ayeesha Parveen.K
* **Team member**: Balasaraswathi.B
* **Team member**: Bhavani.T
* **Team member**: Deepika.R

## 2. Project Overview

### **Purpose:**

The purpose of EduTutor AI is to provide students with personalized learning support using Generative AI, while integrating seamlessly with Learning Management Systems (LMS). The assistant explains concepts, generates quizzes, and helps teachers assess student progress.

* **Features:**

1. **Conversational Interface** – Interact with the AI tutor in natural language.
2. **Concept Explanation** – Get detailed, easy-to-understand explanations with examples.
3. **Quiz Generator** – Automatically create quizzes with MCQs, True/False, and short answers.
4. **Answer Section** – Each quiz includes a separate section with correct answers.
5. **Personalized Learning** – AI adapts explanations based on the student’s input style.

### **3. Architecture**

* **Frontend Interface** – Gradio-based web UI with tabs for concept explanation and quiz generation.
* **Backend Processing** – Python functions handle prompt creation and response generation.
* **Language Model** – Uses Hugging Face’s Granite 3.2-2B causal LLM for text generation.
* **Data Flow** – User input → prompt → model inference → output → displayed in UI.
* **Deployment & Hardware** – Supports GPU (float16) or CPU (float32) with optional Gradio public sharing.

**4**. **Setup Instructions**

* Install Python ≥ 3.8 and required libraries: torch, transformers, gradio.
* Use a CUDA-enabled GPU for faster inference or CPU as fallback.
* Hugging Face automatically downloads ibm-granite/granite-3.2-2b-instruct model and tokenizer.
* Set EOS token as pad token if tokenizer has no pad token for smooth generation.
* Run python app.py to launch the Gradio interface and optionally share publicly.

**5. Folder Structure**

* **app/** – Contains backend logic and API handling.
* **app/api/** – Stores modular API routes (chat, quiz, feedback).
* **ui/** – Holds frontend components for Gradio tabs and layouts.

**6. Running the Application**

* Ensure all project files are in the correct folder.
* Install dependencies using pip install -r requirements.txt.
* Run the main script with python edu\_tutor\_ai.py.
* The Gradio interface will open in your browser.
* Use share=True to generate a temporary public link for sharing.

**7. API Documentation**

* **/chat** – Handles user queries and generates AI responses.
* **/quiz** – Generates quizzes based on the provided topic.
* **/explain** – Returns detailed explanations for given concepts.
* **/feedback** – Accepts user feedback on AI responses or quizzes.
* **/status** – Returns the application’s current health and availability.

**8. Authentication**

The application uses a token-based authentication system to secure API endpoints. Each user must provide a valid API key with their requests to access features like chat, quiz generation, or concept explanations. Tokens are generated upon user registration and stored securely in the backend database. The server validates the token for every request, ensuring that only authorized users can interact with the AI services. Expired or invalid tokens are rejected with an error message, prompting users to re-authenticate. This mechanism helps protect the application from unauthorized access and ensures secure usage of all APIs.

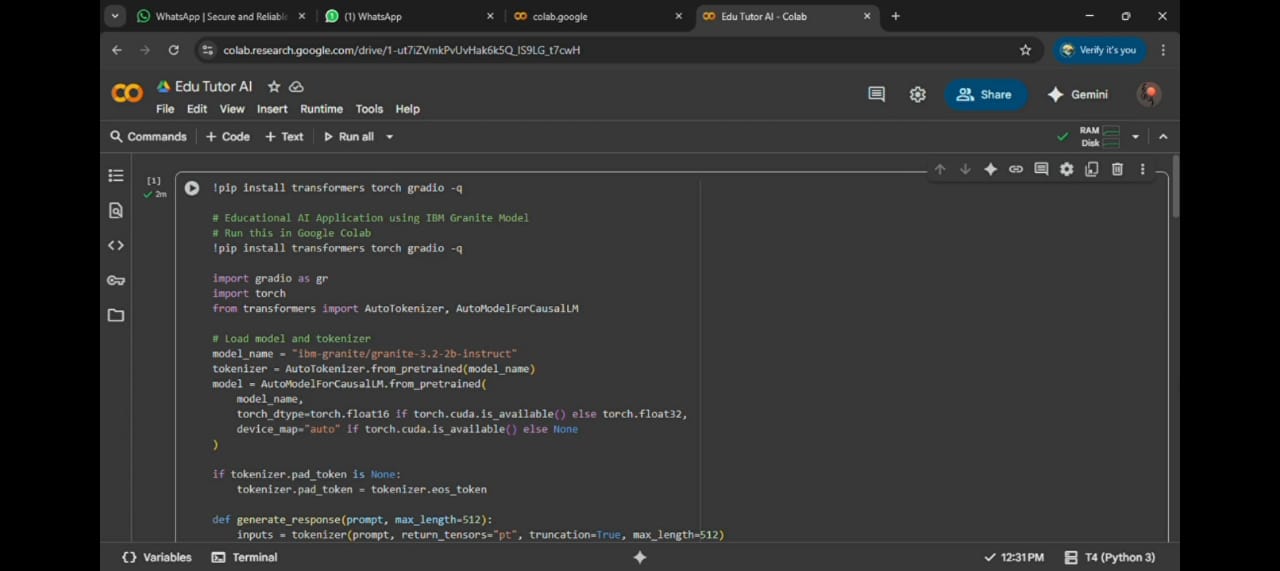
**9. User Interface**

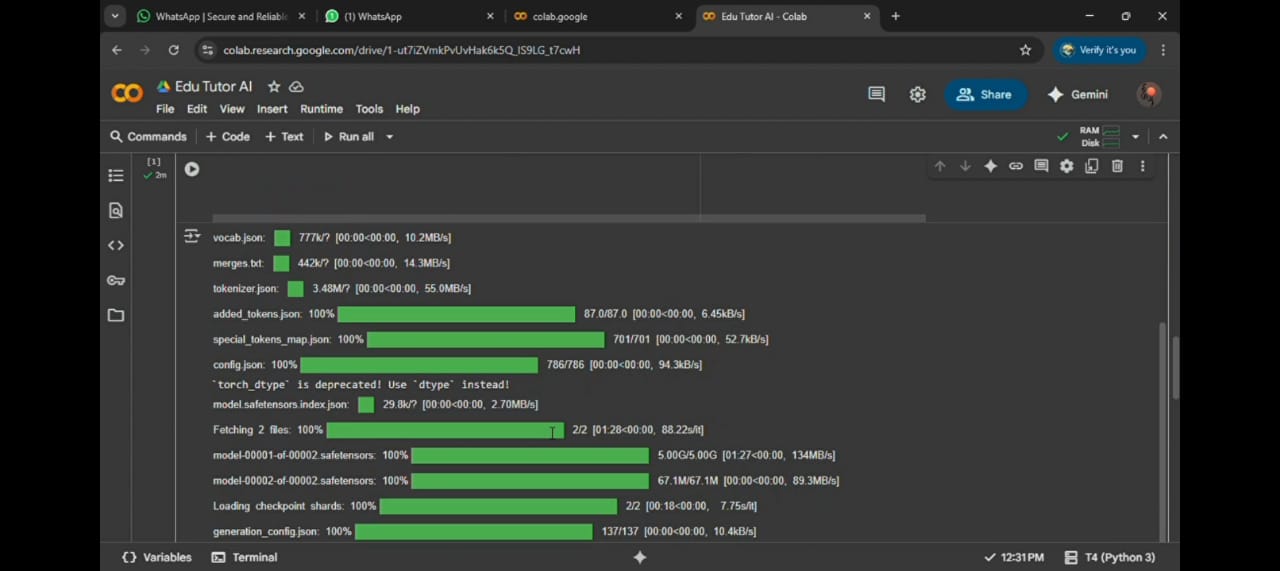
* **Concept Explanation Tab** – Users enter a concept to get detailed explanations with examples.
* **Quiz Generator Tab** – Users input a topic to generate quizzes with multiple question types and answers.
* **Interactive Design** – Gradio textboxes and buttons provide an intuitive, easy-to-use interface for all functionalities.

**10. Tesing**

* Test functions like generate\_response and quiz generation.
* Verify frontend and backend integration via Gradio.
* Collect user feedback for usability and accuracy.

**11. Screenshot**

****

****

**12. Known Issues**

* Large model may run slowly on CPU or low-memory GPUs.
* Long prompts can get truncated due to max token limits.
* Occasionally, generated text may repeat parts of the prompt.

**13. Future enhancement**

* Implement user authentication and role-based access control.
* Add support for multi-language concept explanations and quizzes.
* Integrate persistent storage for saving user progress and quiz history.
* Improve prompt handling to reduce repetition and handle longer inputs.
* Enhance the UI with more interactive elements like charts, images, and downloadable quizzes.