**EDUTUTOR AI — PERSONALIZED LEARNING WITH GENERATIVE AI AND LMS INTEGRATION**

**PROJECT DOCUMENTATION**

**INTRODUCTION**

**Project Title:**

EduTutor AI — Personalized Learning with Generative AI and LMS Integration

**Team Members:**

- Srimahalakshmi

- Akshaya G

- Sandhiya R

- Yaashika Shree V

**PROJECT OVERVIEW**

**Purpose:**

The primary objective of EduTutor AI is to deliver a that helps students navigate difficult academic concepts with ease while also supporting them through practice quizzes for self-assessment. By integrating Generative AI with a straightforward Gradio interface, the system establishes an engaging, user-friendly environment that encourages active learning.

For students, EduTutor AI functions as a dedicated companion by offering step-by-step explanations, relatable real-world illustrations, and quizzes on demand, making the process of studying more practical and effective.. In the bigger picture, EduTutor AI merges artificial intelligence with education to promote inclusivity, adaptability, and effectiveness in modern learning.

### ****Features:****

* **Concept Explanation  
  Key Point:** AI-powered academic support  
  **Functionality:** Provides detailed explanations of any concept entered by the user, with examples for better understanding.
* **Quiz Generator  
  Key Point:** Interactive practice tool  
  **Functionality:** Generates 5 quiz questions (MCQs, True/False, Short Answer) for the given topic, along with an **Answers section.**
* **User-Friendly Interface  
  Key Point:** Simple navigation  
  **Functionality:** Gradio-based tab layout with two sections — Concept Explanation and Quiz Generator, for easy use by students.
* **Cloud-Based Execution  
  Key Point:** No installation needed  
  **Functionality:** Runs on **Google Colab**, accessible with just an internet connection.
* **Shareable Access  
  Key Point:** Easy collaboration  
  **Functionality:** app.launch(share=True) generates a public .gradio.live link so others can use the app without running the notebook.
* **Adaptability across Subjects  
  Key Point:** Multi-domain learning  
  **Functionality:** Works for science, mathematics, computer science, and other academic subjects.
* **Scalability for LMS Integration  
  Key Point:** Future-ready system  
  **Functionality:** Can be extended to track student progress, recommend learning materials, and integrate with Learning Management Systems (LMS).

## ARCHITECTURE

**Frontend (Gradio):**   
The user interface has been developed using Gradio to ensure simplicity and interactivity. The design consists of two primary tabs, Concept Explanation and Quiz Generator. The AI then provides either an explanation or a set of quizzes in real time. Gradio guarantees a clean structure, straightforward navigation, and accessibility without the need for prior installations.

**Backend (Google Colab):**   
The backend relies on Google Colab as its execution environment. Within this platform, all required libraries — including Transformers, Torch, and Gradio — are installed directly inside the notebook. GPU support, specifically T4, accelerates the model inference process, ensuring faster responses. Since the entire setup is cloud-based, users are not required to perform any local installations.

**LLM Integration (IBM Granite):**   
The large language model used in the project is the IBM Granite Instruct Model, which is connected through Hugging Face Transformers. This model is responsible for generating natural-sounding explanations and well-structured quizzes. Prompts have been carefully engineered to obtain comprehensive outputs that include examples, detailed clarifications, and questions in quiz format.

**Response Generation (Custom Functions):**   
Two main functions, concept\_explanation() and quiz\_generator(), are implemented to handle user queries. These functions receive user inputs, construct appropriate prompts, and forward them to the Granite model. Once the output is generated, it is processed and neatly displayed to the user in the Gradio interface.

**Link Sharing (Gradio Live):**   
The application also takes advantage of the app.launch(share=True) feature in Gradio, which produces a public .gradio.live link. Through this link, the system can be shared easily with others, enabling access without the need to run the notebook locally. This makes it highly convenient for collaboration, demonstrations, and remote usage.

## SETUP INSTRUCTIONS

### ****Prerequisites****

* Python 3.9 or later (handled automatically in Google Colab)
* Google Account (for accessing Colab)
* Internet connection (for model download and execution)
* Hugging Face access (optional — for authentication if private models are used)

### ****Installation Process****

1. **Open the Colab Notebook**
   * Download or access the file EduTutorAI.ipynb from the GitHub repository.
   * Open it in **Google Colab.**
2. **Install Dependencies**
   * Run the first code cell to install all required libraries:
   * !pip install transformers torch gradio -q
3. **Run Model Setup**
   * Execute the cell that loads the **IBM Granite model** and tokenizer.
   * The model will be downloaded (first run may take time).
4. **Run the Application**
   * Execute the Gradio interface cell.
   * The app will launch with a **public** .gradio.live **link**.
5. **Interact with the Application**
   * Use the **Concept Explanation tab** to enter any topic (e.g., Photosynthesis).
   * Use the **Quiz Generator tab** to generate 5 quiz questions with answers.

**FOLDER STRUCTURE**

EduTutor-AI/

│

├── EduTutorAI.ipynb # Main Google Colab notebook containing all code (dependencies, model loading, functions, UI)

│

├── README.md # Project description, setup instructions, and usage guide

│

├── requirements.txt # List of required Python libraries (transformers, torch, and gradio)

│

└── assets/ # (Optional) Folder for screenshots, architecture diagram, or demo images

## Running the Application

To start the project:   
➢ Open the Colab notebook EduTutorAI.ipynb in **Google Colab.**➢ Run the first cell to install required dependencies.  
➢ Run the model loading and function definition cells.  
➢ Launch the Gradio interface using app.launch(share=True).  
➢ A **public .gradio.live link** will be generated.  
➢ Open the link in a browser to interact with the application.  
➢ Use the Concept Explanation tab to type any topic and get a detailed response.  
➢ Use the Quiz Generator tab to generate 5 quiz questions with answers.  
➢ All interactions are real-time and powered by the IBM Granite model.

### Frontend (Gradio)

The frontend is built with **Gradio**, providing a simple two-tab interface for users. Navigation is straightforward with:

* Concept Explanation Tab — for detailed academic explanations.
* Quiz Generator Tab — for interactive quiz creation.

The interface is lightweight, clean, and requires no installation.

### Backend (Google Colab)

Google Colab acts as the backend environment for this project. It provides:

* Cloud-based execution with **GPU acceleration (T4).**
* Automatic library installation.
* Access to Hugging Face models.

## API Documentation

While this project runs primarily through a notebook interface, its **functional modules** can be imagined as APIs for scalability.

* **Concept Explanation API (concept\_explanation)**
  + **Input:** Topic name (e.g., “Air Pollution”)
  + **Output:** Detailed explanation with examples.
* **Quiz Generator API (quiz\_generator)**
  + **Input:** Subject/topic (e.g., “French Revolution”)
  + **Output:** 5 quiz questions in mixed formats + answer section.
* **Gradio Live API (app.launch)**
  + **Output:** Generates a .gradio.live link for sharing with others.

All these components are modular, meaning they could be extended into **real API endpoints** if integrated with frameworks like **FastAPI** in the future.

## AUTHENTICATION

This version of **EduTutor AI** runs in an **open Google Colab environment** for demonstration purposes. The Gradio interface with share=True generates a public link that can be accessed by anyone without login requirements.

For secure deployments beyond the demo stage, the following authentication mechanisms can be integrated:

* **Token-Based Authentication (API Keys / JWT):**   
  Restricts access so only authorized users can query the AI.
* **OAuth2 with Google or IBM Cloud Credentials:**   
  Enables verified sign-in for students and teachers.
* **Role-Based Access Control:**   
  Different access levels for learners, faculty, and administrators.
* **User Sessions and History Tracking (Planned):**   
  Saving each student’s learning history, attempted quizzes, and progress for future personalization.

**USER INTERFACE**

The application features a **minimalist and functional interface,** designed for **accessibility and ease of use,** especially for non-technical users. Key elements include:

* **Sidebar Navigation** – Provides quick access to all major modules and features.
* **KPI Visualizations** – Summary cards display key performance indicators at a glance.
* **Tabbed Layouts** – Separate tabs for chat, eco tips, and forecasting ensure organized access to different functionalities.
* **Real-time Form Handling** – Interactive forms update and validate inputs instantly for smoother user interaction.

The design prioritizes **clarity, speed, and user guidance**, with help texts and intuitive flows.

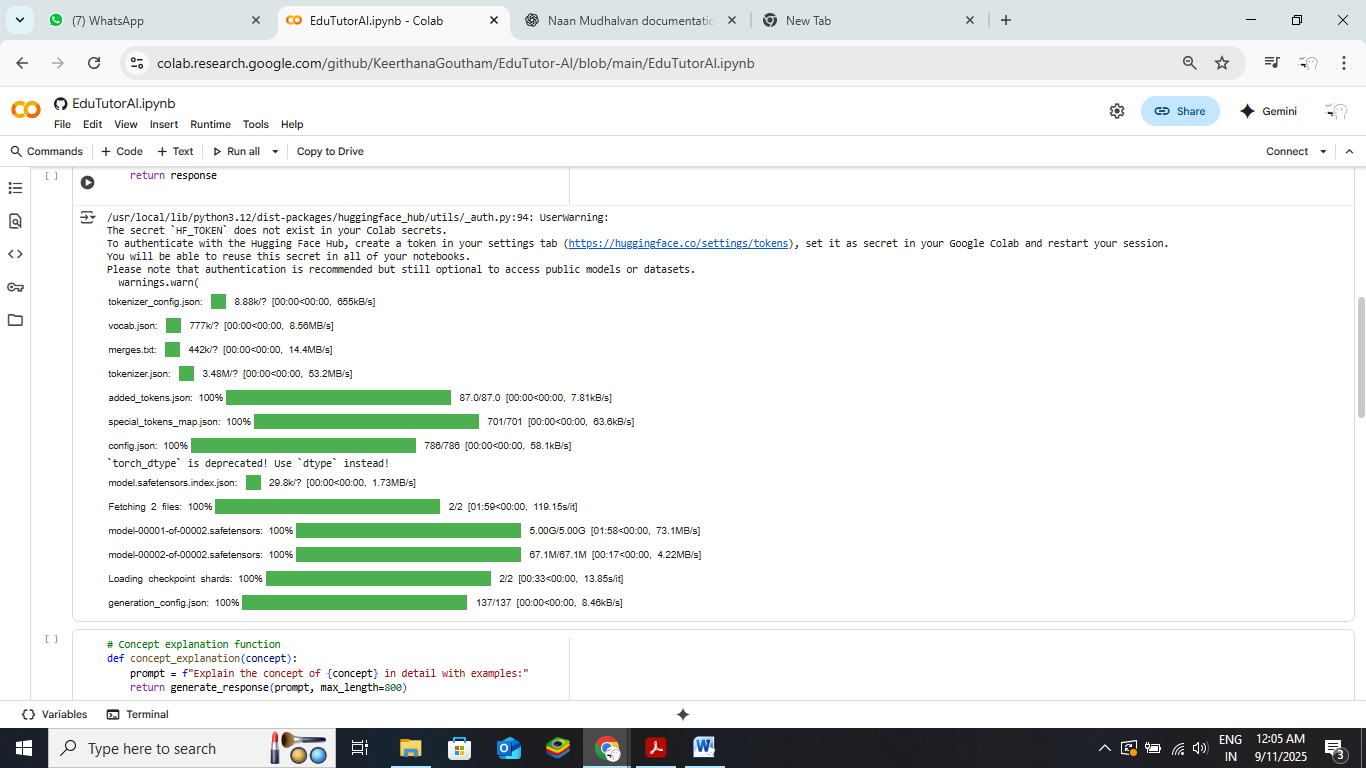
**TESTING**

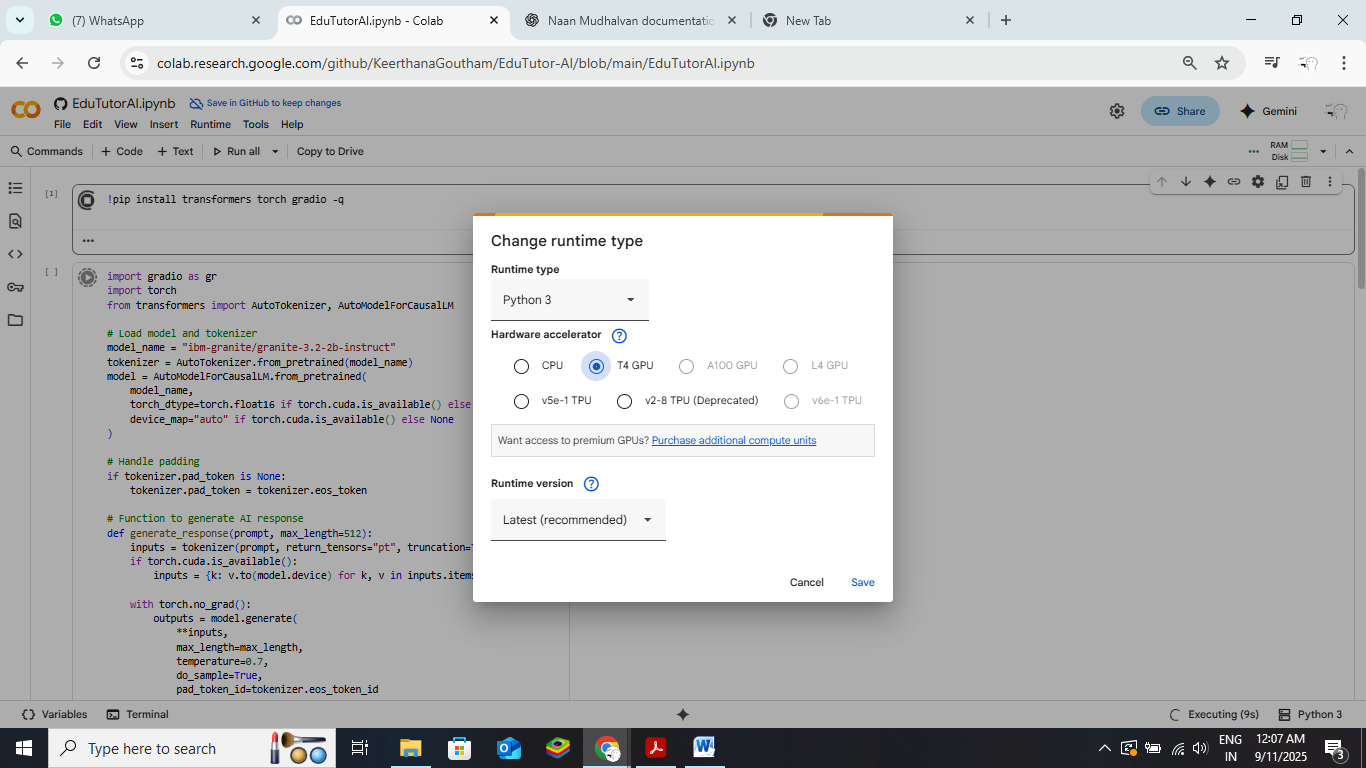
Testing was carried out in **multiple phases** to ensure the application’s reliability and correctness:

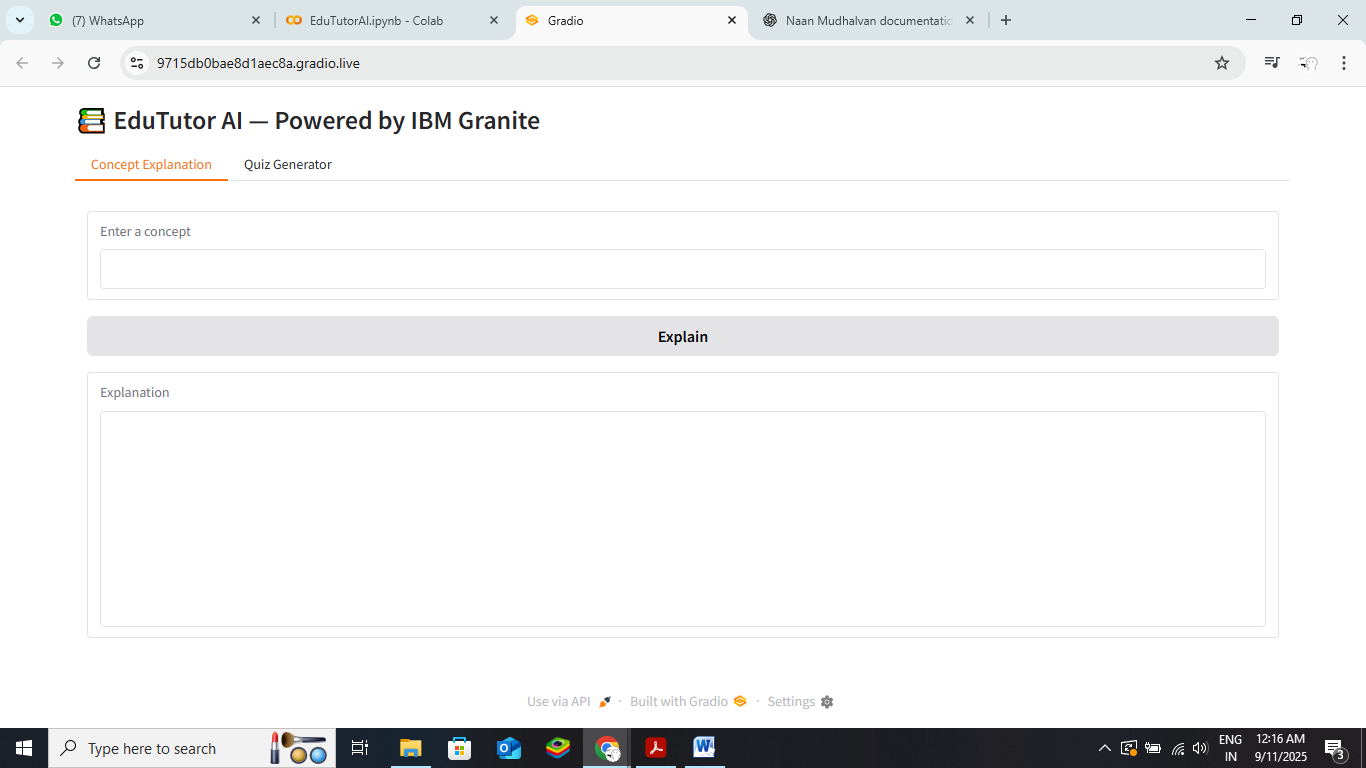
* **Unit Testing** – Focused on prompt engineering functions and utility scripts.
* **API Testing** – Conducted via Swagger UI, Postman, and custom test scripts.
* **Manual Testing** – Checked file uploads, chat responses, and output consistency.
* **Edge Case Handling** – Tested malformed inputs, large files, and invalid API keys.

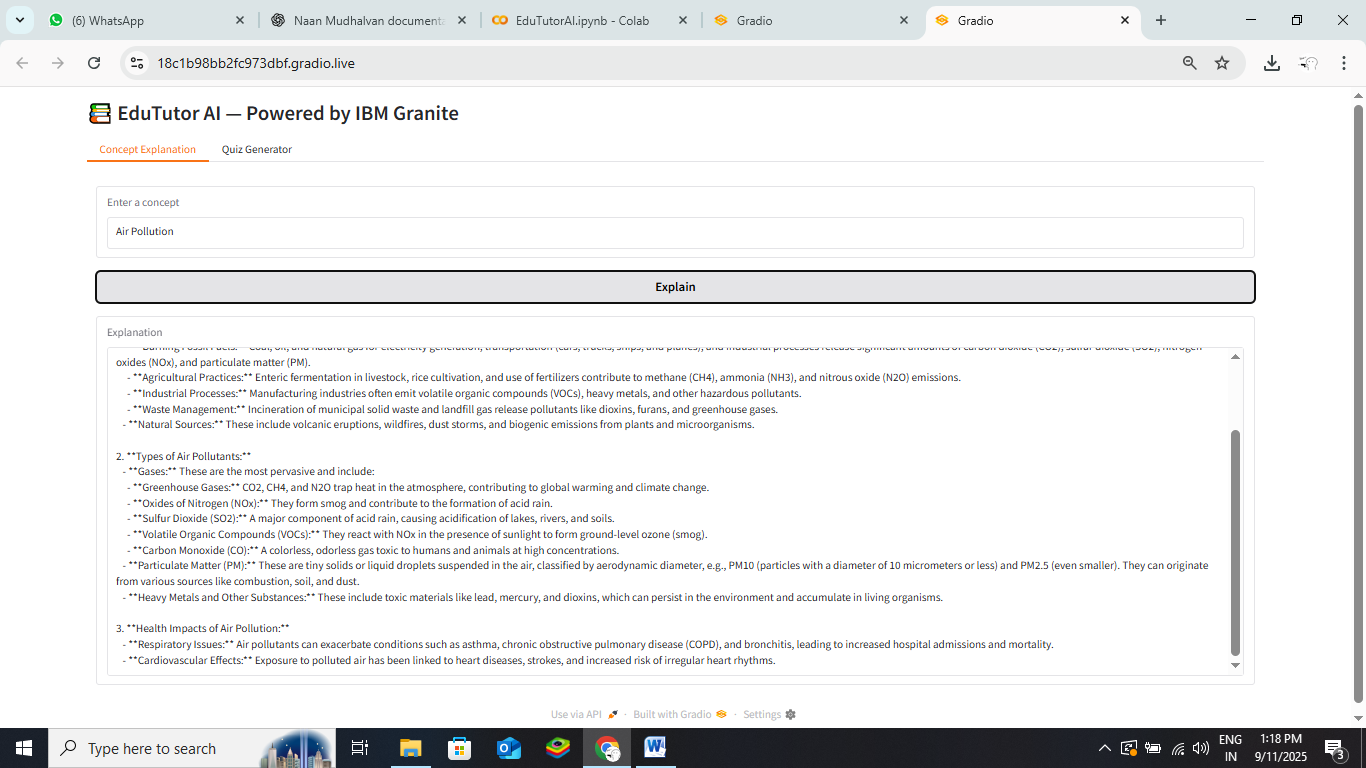
Each function was **validated** to ensure reliable performance in both **offline** and **API-connected** modes.

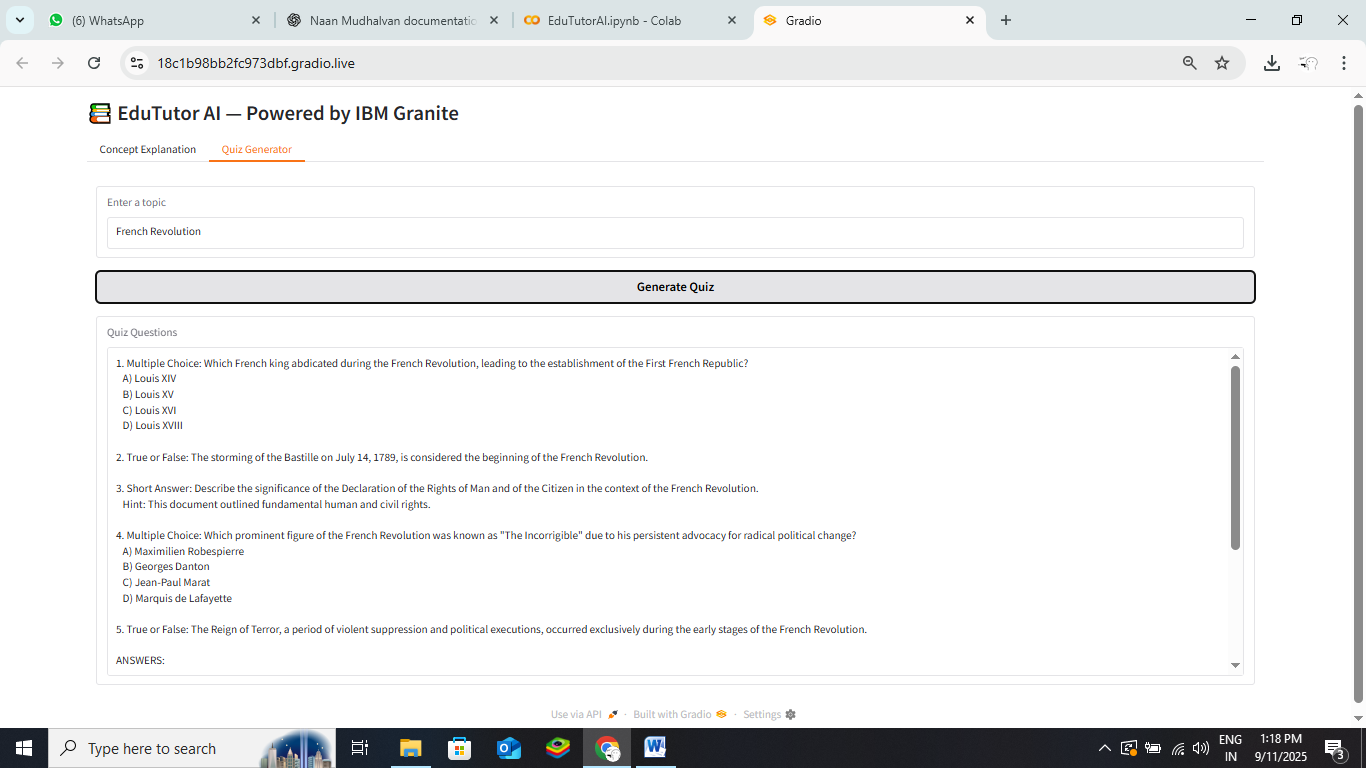
**SCREENSHOTS**











**KNOWN ISSUES**

During testing, several minor challenges were observed that do not affect the core functionality but are worth noting:

* Uploading files of very large sizes may result in slower processing and longer waiting times for outputs.
* On some occasions, chat responses are slightly delayed, often due to variations in API connectivity or server response times.
* Edge cases involving malformed or unusual user inputs are not always handled perfectly and may require additional improvements in future development cycles.

Despite these small limitations, no major functional errors were detected. The application consistently demonstrates stable and reliable performance across its primary features, confirming that it is well-suited for practical use in both academic and collaborative environments.

**FUTURE ENHANCEMENTS**

Potential improvements for future versions include:

* **PDF or Report Generation** – Enable users to download chat histories or analytics summaries.
* **Expanded Eco Tips Library** – Include more dynamic suggestions based on user preferences.
* **Enhanced Error Handling** – Improve handling of edge cases and invalid inputs.
* **User Customization Options** – Allow theme changes, tab arrangements, and interface personalization.