# EduTutor AI: Personalized Learning Assistant Using IBM Granite

## Project Documentation

### 1. Introduction

• **Project title** : Generative AI in Education – EduTutor AI

• **Team ID**: NM2025TMID09633

• **Team Leader** : Vijaysankar. R

• **Team member** :Vijayaragavendran

• **Team member** :WILSON JOYEL. C

• **Team member** :S. Sivasakthi

### 2. Project Overview

• **Purpose** :  
The purpose of this project is to develop an **Educational AI Assistant** using Generative AI that can provide students with clear explanations of academic concepts and generate quizzes for self-assessment. The system helps learners practice and understand subjects in an interactive way. It demonstrates the role of **generative AI in modern education** by offering personalized learning support.

• **Features** :

**Concept Explanation**

* *Key Point*: Topic-based learning
* *Functionality*: Users enter a concept (e.g., “Machine Learning”), and the assistant generates a detailed explanation with examples.

**Quiz Generator**

* *Key Point*: Practice and evaluation
* *Functionality*: Generates **5 quiz questions** in different formats (multiple choice, true/false, short answer) with answers at the end.

**Generative AI Model Integration**

* *Key Point*: Natural language understanding
* *Functionality*: Uses the **IBM Granite LLM** from Hugging Face to produce educational responses and explanations.

**User-friendly Interface**

* *Key Point*: Accessible to all learners
* *Functionality*: Built with **Gradio**, featuring tabbed layout for easy navigation between concept explanation and quiz generation.

**Deployment**

* *Key Point*: Simple and reliable setup
* *Functionality*: Deployed in **Google Colab**, ensuring quick access and low setup effort.

### 3. Architecture

**Frontend (Gradio):**  
The frontend is designed using **Gradio Blocks**, providing a browser-based user interface. It includes tabs for concept explanation and quiz generation, with clear input and output sections.

**Backend (Python with Hugging Face Transformers):**  
The backend manages model loading, tokenization, and response generation using the **IBM Granite instruction-tuned language model**.

**System Flow:**  
User input → Tokenizer → Granite Model generates response → Output displayed on UI

### 4. Setup Instructions

**Prerequisites:**

* Python 3.8 or above
* pip package manager
* Internet connection to download the model

**Installation Process:**

1. Clone the repository from GitHub
2. Install dependencies using requirements.txt
3. Run the application with app.py
4. Open the Gradio link in browser
5. Enter a topic or concept and receive explanation/quiz outputs

### 5. Folder Structure

* **app.py** – Main program integrating model and UI
* **requirements.txt** – Dependency file for Python packages
* **report.docx** – Project documentation
* **screenshots/** – Folder containing sample outputs and UI images
* **deployment\_link.txt** – File containing deployed application link

### 6. Running the Application

To start the application:

1. Run the app.py file in terminal
2. Wait for the model to load and the Gradio server to start
3. Open the local/share link displayed in terminal
4. Navigate between tabs to use “Concept Explanation” and “Quiz Generator”

### 7. API Documentation

The project is implemented with **Gradio** and does not require separate API endpoints. The core methods are:

* **Concept Explanation** – Generates explanations for entered topics
* **Quiz Generator** – Produces quizzes and answers for practice

### 8. Authentication

The current version does not include authentication.  
Future versions may include:

* User login for teachers and students
* Role-based access control
* Privacy protection for student data

### 9. User Interface

The interface is simple and user-friendly, including:

* Tabbed sections for different features
* Textbox inputs for topics/concepts
* Buttons to generate explanations or quizzes
* Output areas displaying results clearly

### 10. Testing

Testing was carried out in the following ways:

* **Unit Testing** – Functions for explanation and quiz generation validated
* **Manual Testing** – Checked with multiple subjects and topics
* **Edge Case Testing** – Tested empty fields, long text inputs, and uncommon topics

### 11. Screenshots

Screenshots include:

* Application UI
* Example explanation output
* Example quiz output

### 12. Known Issues

* Model may occasionally give vague or repetitive responses
* Large model size can cause slower loading on low-resource systems

### 13. Future Enhancement

* Add **speech-to-text input** for accessibility
* Include **student progress tracking** features
* Extend to support **interactive diagrams and multimedia explanations**
* Optimize model for faster response time
* Add authentication for secure usage in schools/colleges