EduTutor AI: Personalized Learning Assistant Using IBM Granite LLM

1.Intoduction

Project title: Sustainable smart city assistant IBM granite LLM

MEMBERS	NAME	NM_ ID
Team Leader	Sundhar	BB6098C9DB9E58BAF9C4FB0403D8B5BC
Team Member	Sundhar Murthy	274E15812740954863 6493B0EA224E57
Team Member	Vallarasu	IFB8B9L0AE60E96BD3247421516FA64D
Team Member	Arun	967B46EE88C3FAA818C4A725ESDFD6DA

2. Project Overview

• Purpose

EduTutor AI is designed to empower students and learners with a personalized AI-powered tutor. By leveraging **IBM Granite LLM**, the assistant provides detailed explanations of concepts, generates quizzes, creates study plans, and solves academic problems step by step. The project makes advanced AI accessible in an educational setting, helping students learn interactively and teachers deliver personalized guidance.

• Features

1. **Concept Explanation**
    - Provides detailed explanations of academic concepts across multiple subjects.
    - Uses real-world examples to make learning more relatable.
    - Helps students understand both basic and advanced topics in a simple way.
2. **Quiz Generator**
    - Automatically generates quizzes with multiple question types (MCQ, True/False, Short Answer).
    - Includes correct answers at the end for self-assessment.
    - Helps teachers quickly create assessments and students to practice.
3. **Study Planner**
    - Creates personalized study timetables based on subject and duration.
    - Breaks topics into daily goals with tips for effective learning.
    - Encourages consistency and productivity in preparation for exams.
4. **Problem Solver**
    - Solves mathematical, logical, and analytical problems step by step.
    - Explains each step clearly so learners can follow the logic.
    - Supports a wide range of problems from basic equations to advanced reasoning.
5. **Interactive Learning Interface**

- Built with **Gradio**, featuring simple tabs for each feature.
- Copy-to-clipboard buttons for easy sharing of generated outputs.
- Intuitive and accessible design for both students and educators.

6. **AI-Powered Personalization**
   - Uses **IBM Granite LLM** to adapt explanations and solutions based on user input.
   - Can provide different levels of detail depending on the learner's background.
   - Future-ready for integration with student performance tracking.
7. **Cloud-Based Deployment**
   - Runs seamlessly on **Google Colab** with GPU support.
   - Provides a live demo link through Gradio for easy access.
   - No need for complex local setup, making it lightweight and accessible anywhere.

3. Architecture

1. Frontend (Gradio UI)

- Built with **Gradio**, offering a clean and interactive web interface.
- Organized into **four main tabs**:
  - 📚 Concept Explanation
  - ❓ Quiz Generator
  - 🗓️ Study Planner
  - 🧮 Problem Solver
- Includes **copy-to-clipboard** options for easy sharing.
- Runs directly in a browser through Google Colab links, requiring no local installation.

2. Backend (Python Runtime in Google Colab)

- Core application logic is written in **Python**.
- Execution environment: **Google Colab** with GPU/TPU support for faster model inference.
- Handles user queries, processes inputs, and generates AI-driven outputs.
- Includes utility functions for text generation, study scheduling, and step-by-step problem solving.

3. LLM Integration (IBM Granite on Hugging Face)

- Uses **ibm-granite/granite-3.2-2b-instruct** model hosted on Hugging Face.
- Provides natural language understanding and response generation.
- Model is optimized for educational purposes with strong reasoning and explanation ability.
- Tokenizer and model are automatically loaded and cached during execution in Colab.

4. Deployment Layer (Google Colab + Gradio Share)

- The app is launched via **Gradio's app.launch(share=True)** command.
- Generates a **public sharable link**, making the app accessible from any device with internet.
- No dedicated server hosting required, keeping the setup lightweight.
- Can be extended for **cloud deployment** (e.g., Hugging Face Spaces, Streamlit Cloud, or IBM Cloud).

5. Workflow

1. **User Input** → Student enters concept, topic, subject + duration, or problem.
2. **Frontend Processing** → Gradio captures the input from the relevant tab.
3. **Backend Processing** → Python functions format the query and send it to the IBM Granite model.
4. **LLM Response** → Granite model generates explanations, quizzes, plans, or solutions.
5. **Output Delivery** → Gradio displays results in a structured format (textboxes, planners, Q&A).

6. Extensibility

- Can integrate with **student performance tracking systems** for adaptive learning.
- Future scope includes **voice-based interaction**, **multilingual support**, and **mobile-friendly apps**.

4. Setup Instructions

Prerequisites

- Python 3.9+
- Google Colab account with GPU runtime enabled
- Hugging Face account (for accessing IBM Granite models)
- Basic familiarity with Python and Gradio

Installation Process

1. Open Google Colab.
2. Install required dependencies:
3. !pip install transformers torch gradio
4. Import model and tokenizer from Hugging Face.
5. Launch the Gradio app and share the demo link.

5.Folder Structure

Notebook-Based Structure (Google Colab)

- **EduTutor AI.ipynb** → Main notebook containing all the code, divided into the following sections:
    - **1. Configuration & Setup**
        - Install dependencies (transformers, torch, gradio).
        - Define constants: model name, max tokens, temperature.

- **2. Model Initialization**
  - Load IBM Granite model from Hugging Face (granite-3.2-2b-instruct).
  - Load tokenizer and configure padding tokens.
- **3. Core Functions**
  - generate_response(prompt) → Base function for AI responses.
  - concept_explanation(concept) → Explains concepts with examples.
  - quiz_generator(concept) → Generates quizzes with answers.
  - study_planner(subject, duration) → Creates study schedules.
  - problem_solver(problem) → Solves math/logic problems step by step.
- **4. Gradio UI Setup**
  - Build interface with four tabs (Concept, Quiz, Planner, Solver).
  - Define textboxes, buttons, and output areas.
- **5. Launch Application**
  - Start Gradio app with app.launch(share=True) for demo link.

6. Running the Application

Step-by-Step Instructions

1. **Open Google Colab**

Go to Google Colab.

Create a new notebook or upload the EduTutor_AI.ipynb file.

Import Libraries & Initialize Model

Import transformers, torch, and gradio.

Load **IBM Granite model (granite-3.2-2b-instruct)** and tokenizer from Hugging Face.

Configure model parameters:

MAX_TOKENS = 1024

TEMPERATURE = 0.7

Define Core Functions

generate_response() → Handles AI text generation.

concept_explanation() → Provides detailed explanations with examples.

quiz_generator() → Creates quizzes with answers.

study_planner() → Generates structured study schedules.

problem_solver() → Solves mathematical/logical problems step by step.

7. API Documentation

 Concept Explanation Tab:
Input: Topic name → Output: Detailed explanation with examples.

 Quiz Generator Tab:
Input: Topic name → Output: 5 quiz questions with answers.

 Study Planner Tab:
Input: Subject + Duration → Output: Study timetable with goals.

Problem Solver Tab:
Input: Math/logic problem → Output: Step-by-step solution.

during development.

8. Authentication

This project runs openly in Colab for demo purposes. For secure deployment, Hugging Face tokens and Colab API keys can be integrated.

9. User Interface

The Gradio interface is:

- **Minimalist** and user-friendly

- Organized into **tabs** for each feature
- Includes **copy-to-clipboard** buttons for generated results
- Runs seamlessly in the browser via Colab links

10. Testing

Testing included:

- **Unit Testing**: Validating each function (concept explanation, quiz generator, etc.)
- **Manual Testing**: Ensuring each tab runs correctly in Gradio.
- **Edge Case Handling**: Empty inputs, overly long text, or invalid queries.
- **Output Verification**: Checking generated quizzes, solutions, and study plans for accuracy and clarity.

11.screen shots