EduTutor AI- Personalized Learning Assistant Using IBM Granite LLM

Project Documentation

1.Introduction

- Project title : EduTutor AI
- Team Leader : K.Kinjal
- Team Member : Thasleema.U
- Team Member : S.Sabareshwari
- Team Member : S.Sivasakthi

2.Project Overview

- Purpose :

EduTutor AI is designed to act as an educational assistant that helps students learn concepts more effectively. It explains concepts in simple terms and generates quizzes for practice. The system uses IBM Granite LLM models to deliver high-quality explanations and interactive quizzes, making learning more engaging.

- Features :

Concept Explanation

Key Point: Understand any topic clearly

Functionality: Students enter a concept, and the assistant generates a detailed explanation with examples.

Quiz Generator

Key Point: Practice based learning

Functionality: Automatically generates 5 questions (MCQs, True/False, short answer) with an ANSWERS section.

Interactive UI

Key Point: User-friendly learning

Functionality: Build with Gradio ,includes tabs for different tasks.

Semantic Search & Knowledge Base

Key Point: Fast retrieval of relevant content.

Functionality: Index course materials using embeddings so students can ask natural-language queries and retrieve exact paragraphs or explanations.

Flexibility Across Subjects
Key Point: Multiple domain learning
Functionality: Works with science, maths, computer science, history and general knowledge topics.

Adaptive Responses
Key Point: Context-aware answers
Functionality: Adjusts explanations and quizzes on input length , topic and complexity.

Lightweight Development
Key Point: Easy to run
Functionality: Requires only python environment and Gradio interface to launch.

Multimodal Input Support
Key Point: Flexible data handling
Functionality: Accepts text, PDFs, and CSVs for document analysis and forecasting.

AI -Powered Responses
Key Point: Generative AI with IBM Granite
Functionality: Uses IBM Granite 3.2-2B instruct model to generate human-like answers and quizzes.

3. Architecture

Frontend (Gradio):
The user interface is implemented with gradio . It has two main tabs:

Concept Explanation tab → Input textbox + “Explain” button → AI-generated explanation output.

Quiz Generator tab → Input textbox + “Generate Quiz” button → Quiz questions with answers displayed in output.

Backend (Model and Functions): The backend handles all AI operations:

Python functions for concept explanation and quiz generation (concept_explanation, quiz_generator).

generate_response() → Core function that sends prompts to IBM Granite and decodes output.

concept_explanation() → Wraps generate_response to create detailed explanations.

quiz_generator() → Wraps generate_response to create structured quizzes.

Flow:

1. User enters a concept/topic in the UI.

2. Input is tokenized → passed to Granite LLM.

3. Model generates explanation or quiz.

4. Output is decoded, cleaned, and displayed in Gradio.

4. Setup Instructions

Prerequisites:

Google Colab, Python 3.9+ ,Hugging Face transformers

Liberaries: Torch (with CUDA if GPU available) , Gradio

Installation Process:

!pip install transformers gradio torch accelerate

Run in Colab

Upload your notebook/code.

Run the setup cell to install dependencies.

Run the Gradio app cell → link will be generated.

5. Folder Structure

```
EduTutorAI/
├── edututor_app.ipynb       # Main Colab notebook
├── models/                  # Hugging Face Granite model (cached)
├── utils/                   # (Optional future: helper functions)
├── outputs/                 # Save generated quizzes/answers
└── screenshots/             # UI screenshots
```

6. Running the Application

To start the project:

- Open the Colab notebook.
- Install dependencies and run all cells.

App starts automatically with Gradio.

- Concept Explanation Tab: Enter concept → Click Explain → Get detailed response.
- Quiz Generator Tab: Enter topic → Click Generate Quiz → Get questions + answers.

Frontend (Streamlit/Gradio):

The frontend is built with Streamlit (dashboard) and Gradio (prototype), providing an easy-to-use interactive web UI. It includes:

Student Dashboard → progress tracking and personalized study plans.

Chat Tutor → ask questions and receive AI-generated explanations.

Quiz Generator → automatic quizzes with answers.

Material Upload Page → teachers upload PDFs/notes

Backend (FastAPI):

The backend is powered by FastAPI, exposing REST endpoints for:

Document Processing → upload and convert course material into embeddings.

AI Tutor Interaction → concept explanations using IBM Granite LLM.

Quiz Generation → automatic quiz creation from study materials

Study Plan Generator → adaptive personalized learning past

Progress Tracking → stores results, tracks performance, and updates dashboards.

7. API Documentation

Backend APIs available include:

POST /chat/ask – Accepts a student query and responds with an AI-generated explanation (using IBM Granite LLM).

POST /materials/upload – Uploads study materials (PDFs/notes) and embeds them in Pinecone for retrievalGET /materials/search – Returns semantically similar content chunks related to the student's query.POST

/quiz/generate – Generates quizzes with answers based on uploaded study content.

generate_response(prompt, max_length)

→ Sends prompt to Granite model and returns generated text.

concept_explanation(concept)

→ Returns a detailed explanation for the given concept.

quiz_generator(concept)

→ Generates 5 quiz questions with answers.

8. Authentication

Query and history tracking for each student. Currently, the demo version runs in an open environment for ease of testing. Planned security features include:

Token-based authentication (JWT or API keys).

OAuth2 with IBM Cloud credentials for enterprise deployments.

Role-based access:

Student → access quizzes, explanations, study plans.

Teacher → upload materials, review AI-generated content, view student analytics.

Admin → manage roles, monitor usage, review logs.

Future enhancements:

Secure user sessions.

9. User Interface

The interface is minimalist, student-friendly, and accessible, focusing on ease of use for both learners and educators.

It include: Sidebar Navigation → easy movement between dash bard, quizzes, study plans, and uploads.

Student Dashboard → shows progress summary, mastery charts, and pending tasks.

Tutor Chat → interactive chat interface for real-time concept explanations.

Quiz Page → generates and conducts quizzes with immediate feedback.

The design prioritizes clarity, speed, and accessibility, with simple layouts, tooltips, and responsive elements.

10. Testing

Unit Testing: Verified individual functions (concept_explanation, quiz_generator).

Manual Testing: Tested multiple concepts (e.g., machine learning, physics, history).

Integration Testing → Checked end-to-end flow: upload → embed → query → AI answer → progress update.

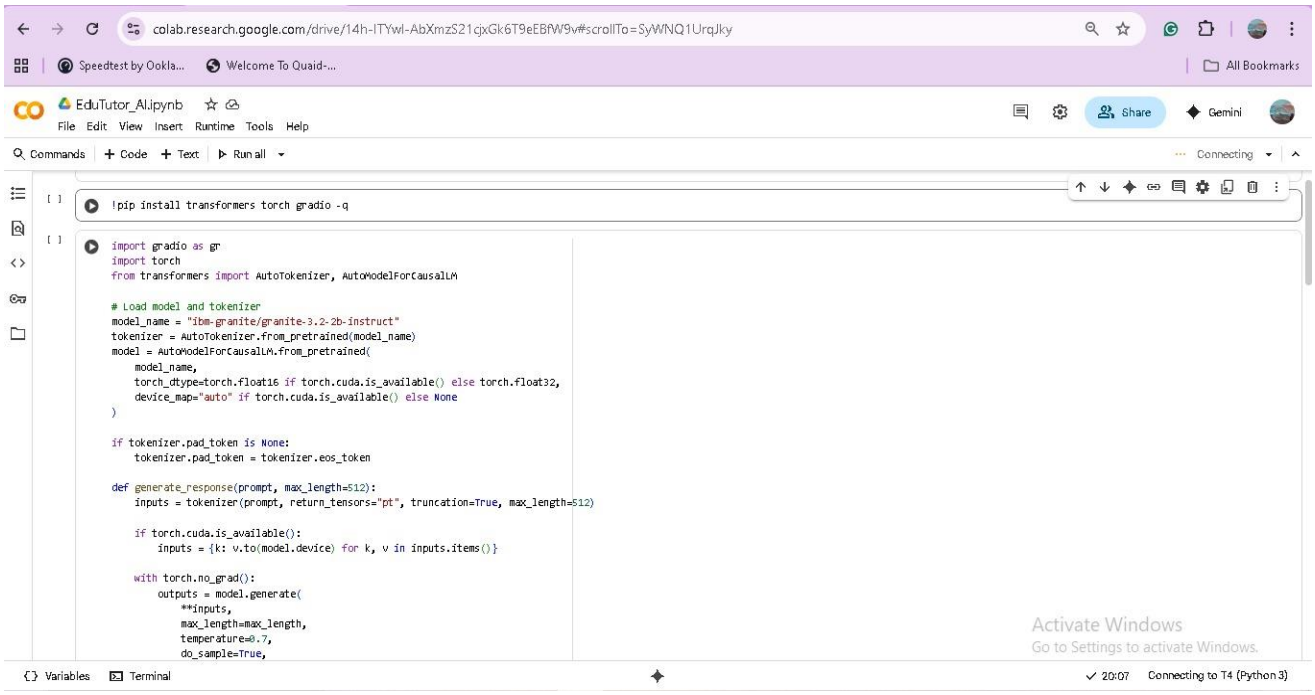
User Testing → Students and teachers tested chat, quizzes, dashboards; feedback used for improvements.

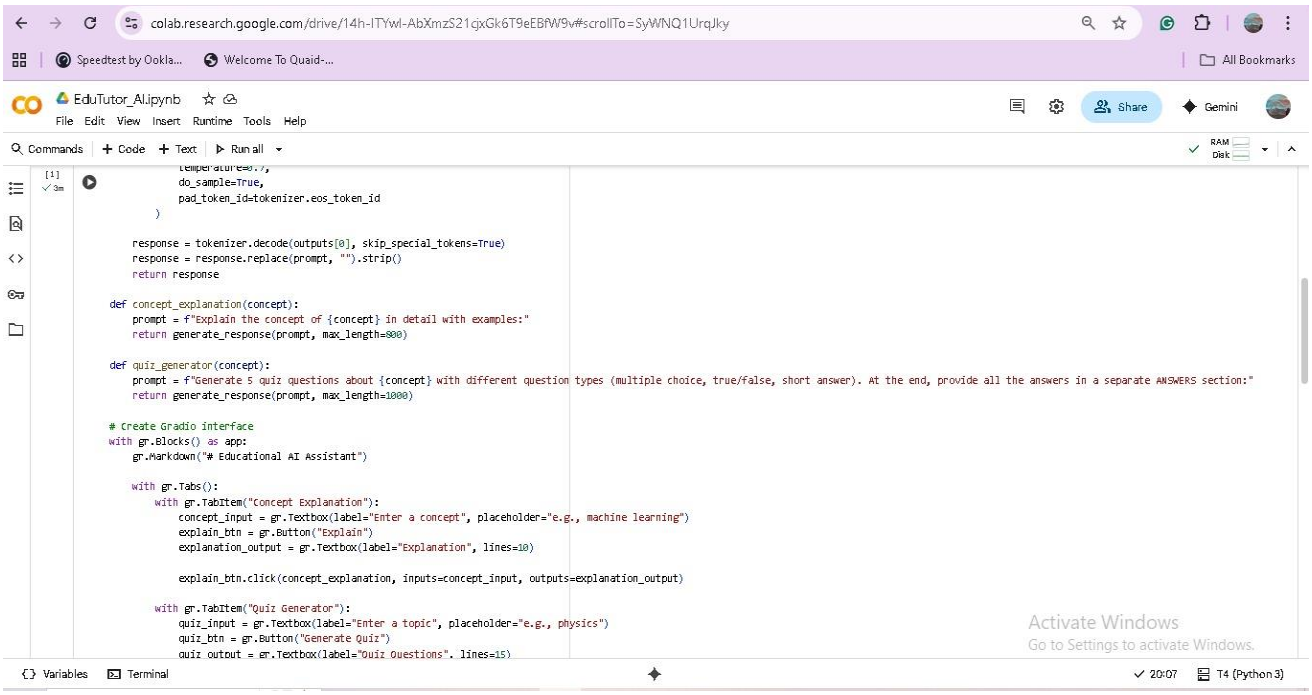
Performance Testing → Ensured fast response times and stable API behaviour.

LLM Testing → Reviewed explanations and quizzes for clarity, accuracy, and reduced hallucinations.

UI Testing → Confirmed responsive design and accessibility on multiple devices.

11.Screen shot





colab.research.google.com/drive/14h-ITYwl-AbXmzS21cjxGk6T9eEBfW9v#scrollTo=SyWNQ1UrqJky

EduTutor_AI.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[1] 3m

```python
# Create Gradio interface
with gr.Blocks() as app:
    gr.Markdown("# Educational AI Assistant")

    with gr.Tabs():
        with gr.TabItem("Concept Explanation"):
            concept_input = gr.Textbox(label="Enter a concept", placeholder="e.g., machine learning")
            explain_btn = gr.Button("Explain")
            explanation_output = gr.Textbox(label="Explanation", lines=10)

            explain_btn.click(concept_explanation, inputs=concept_input, outputs=explanation_output)

        with gr.TabItem("Quiz Generator"):
            quiz_input = gr.Textbox(label="Enter a topic", placeholder="e.g., physics")
            quiz_btn = gr.Button("Generate Quiz")
            quiz_output = gr.Textbox(label="Quiz Questions", lines=15)

            quiz_btn.click(quiz_generator, inputs=quiz_input, outputs=quiz_output)

app.launch(share=True)
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 8.88k/? [00:00<00:00, 504kB/s]
vocab.json: 777k/? [00:00<00:00, 17.1MB/s]
merges.txt: 442k/? [00:00<00:00, 8.38MB/s]
```


Variables Terminal ✓ 20:07 T4 (Python 3)

colab.research.google.com/drive/14h-ITYwl-AbXmzS21cjxGk6T9eEBfW9v#scrollTo=SyWNQ1UrqJky

EduTutor_AI.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 8.88k/? [00:00<00:00, 504kB/s]
vocab.json: 777k/? [00:00<00:00, 17.1MB/s]
merges.txt: 442k/? [00:00<00:00, 8.38MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 36.5MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 3.80kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 24.7kB/s]
config.json: 100% 786/786 [00:00<00:00, 27.7kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 1.53MB/s]
Fetching 2 files: 100% 2/2 [01:56<00:00, 116.38s/it]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:01<00:00, 45.4MB/s]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [01:55<00:00, 42.8MB/s]
Loading checkpoint shards: 100% 2/2 [00:18<00:00, 7.79s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 16.6kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://4be6564d85b9526d25.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (https://huggingface.co/spaces
```


Variables Terminal ✓ 20:07 T4 (Python 3)

4be6564d85b9526d25.gradio.live

Speedtest by Ookla... Welcome To Quaid-... All Bookmarks

Educational AI Assistant

Concept Explanation | Quiz Generator

Enter a concept

cloud computing

Explain

Explanation

Cloud computing is a revolutionary paradigm shift in information technology, fundamentally transforming how digital services are delivered, managed, and consumed. It represents a significant departure from traditional on-premises computing models, where organizations own and maintain their hardware and software infrastructure. Instead, cloud computing offers shared, scalable, and elastic resources that can be accessed via the internet, enabling users to provision and utilize computing power, storage, databases, network services, and various software applications on-demand.

At its core, cloud computing relies on three primary service models:

1. **Infrastructure as a Service (IaaS)**: This is the most fundamental level of cloud services, offering virtualized computing resources such as servers, storage, and networking components. Users can provision, configure, and manage these basic building blocks independently, just as if they were operating their own data center. Well-known IaaS providers include Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). For example, an e-commerce company might use AWS IaaS to launch virtual machines for their transaction processing, database management, and content delivery systems, thereby avoiding the upfront capital expenditure of purchasing, setting up, and maintaining physical servers.

2. **Platform as a Service (PaaS)**: In this model, cloud providers deliver a complete, managed software development platform, including middleware, runtime environments, and development tools. Users can then develop, test, deploy, and manage their applications without worrying about underlying infrastructure. PaaS enables rapid application development and agility in

4be6564d85b9526d25.gradio.live

Speedtest by Ookla... Welcome To Quaid-... All Bookmarks

Educational AI Assistant

Concept Explanation | Quiz Generator

Enter a topic

machine learning

Generate Quiz

Quiz Questions

1. Multiple Choice: Which of the following is NOT a type of machine learning?
   a) Supervised
   b) Unsupervised
   c) Semi-supervised
   d) Enthusiastic

2. True/False: Feature engineering is the process of transforming raw data into a more suitable format for machine learning algorithms.

3. Short Answer: Briefly explain the concept of overfitting in machine learning.

4. Multiple Choice: Which of the following techniques is BEST for handling missing data in a dataset?
   a) Filling with mean value
   b) Using decision tree-based models
   c) Applying the k-nearest neighbors method

12.Known Issues

No quiz difficulty control (ll questions are same level).

No offline mode (must load from Hugging Face).

Quiz formatting can sometimes be inconsistent.

Session data is not saved in Colab (no persistence).

12. Future Enhancements

Enable progress history tracking and performance analytics.

Support multiple languages (Hindi, Tamil, etc.).

Introduce gamification (points, leaderboards).

Add quiz difficulty levels (easy, medium, hard).

Provide export feature for quizzes in PDF/Word.

Add voice input/output for accessibility.