EduTutor

Project Documentation

INTRODUCTION

- Project Title:EduTutor
- Team Member:Renuka.L
- Team Member:Nisha.S
- Team Member:Asmeena Bee.H

Artificial Intelligence has become one of the most powerful technologies shaping the modern world, and its role in education is growing rapidly. Traditional teaching methods often cannot meet the different learning speeds and styles of individual students. Many learners struggle with concepts because they do not receive personalized support. EduTutor AI has been designed to bridge this gap by using IBM Granite models to provide explanations and practice exercises. The project shows how AI can be applied to make education more flexible, personalized, and accessible. It not only helps students but also provides hands-on learning opportunities for developers who wish to explore AI in education.

PROJECT OVERVIEW

- EduTutor AI is a personalized digital tutor created using IBM Granite models available on Hugging Face.
- The system provides students with two key tools: a concept explainer and a quiz generator.
- The project is implemented in Google Colab, which ensures smooth performance by utilizing GPU support.
- A user-friendly interface is built using Gradio, which allows students to interact with the tutor in a conversational manner.
- All project files are stored on GitHub for accessibility and version control.
- The overview highlights how EduTutor integrates modern AI with simple tools to create an effective learning platform.

PURPOSE OF THE PROJECT

- The main purpose of EduTutor AI is to show how generative AI models can be applied in education for real learning support.
- The project focuses on providing personalized explanations so that students can understand concepts at their own pace.

- It also includes automatic quiz generation to help learners test their knowledge and identify areas of improvement.
- Apart from academic support, the project aims to give developers hands-on practice in using AI frameworks, cloud-based tools, and version control systems.
- In this way, the project benefits both learners and developers.

FEATURES

- EduTutor AI has several important features that make it different from traditional learning tools.
- The conversational interface allows students to ask questions in natural language and get instant responses.
- The concept explanation feature breaks down difficult topics into simple steps with examples.
- The quiz generator creates practice questions automatically so students can test themselves.
- The project uses a lightweight Granite model, which runs efficiently in Google Colab with GPU support.
- The interface is built with Gradio, making it easy to use even for non-technical learners.

- The system is scalable, allowing future features to be added, and GitHub integration ensures smooth collaboration and deployment.

ARCHITECTURE

- The architecture of EduTutor AI is divided into three main layers: frontend, backend, and AI engine.
- The frontend uses Gradio to provide a simple, interactive interface where students can enter questions and receive explanations.
- The backend is powered by FastAPI, which ensures smooth communication between the interface and the AI model.
- At the core lies the IBM Granite language model, which generates personalized explanations and quizzes.
- To improve search efficiency, vector search using Pinecone can be integrated.
- The architecture is modular, meaning it can easily support future features such as forecasting or anomaly detection.
- This design ensures flexibility, scalability, and smooth functioning.

METHODOLOGY

- The development of EduTutor AI followed a structured methodology to ensure both technical accuracy and educational usefulness.
- The first step was the selection of an appropriate AI model, and IBM Granite-3.2-2B was chosen because it provides efficient and reliable performance.
- The second step was setting up the environment using Google Colab with GPU support, ensuring accessibility for all users.
- The next step involved building an interface using Gradio to make the system interactive and user-friendly.
- After that, the Granite model was integrated to generate concept explanations and quizzes.
- Finally, version control using GitHub was added to manage files, track updates, and support team collaboration.

IMPLEMENTATION

- The implementation of EduTutor AI was carried out in Google Colab to take advantage of cloud-based GPU resources.

- The required Python libraries such as Transformers, Torch, and Gradio were installed for smooth execution.
- The Granite model was imported from Hugging Face and connected to the Gradio interface through Python functions.
- The code included modules for generating detailed explanations of concepts and creating quizzes based on input topics.
- Once the notebook was executed, Colab generated a Gradio link, which could be opened in a new tab to test the system.
- This simple yet effective implementation process made EduTutor both accessible and reliable.

TESTING

- Testing of EduTutor AI was carried out to ensure that the application runs smoothly and performs the intended tasks without errors.
- The first level of testing was setup testing, where the Colab environment was checked for GPU compatibility, library installations, and model loading.
- Once the system was configured, functional testing was performed to verify whether the model could

- correctly generate explanations for different concepts and produce relevant quizzes.
- Various input topics from mathematics, science, and computer science were tested to check consistency and response quality.
- Basic error handling was also tested by entering invalid or incomplete queries to ensure the system did not crash.
- Although only manual testing was performed in this version, it helped confirm that the system was functional and reliable for learning support.

EVALUATION

Evaluation of EduTutor AI focused on measuring its effectiveness, accuracy, and user experience. The responses generated by the IBM Granite model were reviewed for clarity, factual correctness, and ease of understanding. While most explanations were accurate, in some cases the model provided repetitive or incomplete answers, highlighting the need for future fine-tuning. The quiz generation feature was evaluated based on how relevant and diverse the questions were, and overall, it produced meaningful sets of practice exercises. In terms of performance, the system worked well with GPU support in Colab but showed noticeable

slowdowns on CPU execution. From a user perspective, the Gradio interface was intuitive and easy to use, making the application suitable even for beginners. Overall, the evaluation showed that EduTutor AI is a promising tool with room for enhancements in accuracy and scalability.

APPLICATIONS AND USE CASES

- EduTutor AI has multiple applications in real educational settings.
- In schools, it can serve as a supplementary tool for students to clarify doubts and practice topics outside the classroom.
- In colleges, it can help students with independent study and provide quick assistance in technical subjects.
- For online learners, EduTutor is a cost-effective alternative to expensive tutoring services.
- Teachers can also use the quiz generation feature to prepare tests and assignments faster.
- By making learning interactive and accessible, EduTutor can be applied in both formal and informal education systems across different levels.

KNOWN ISSUES AND LIMITATIONS

- Despite its strengths, EduTutor AI has certain limitations.
- Since it depends on the IBM Granite model, sometimes the responses may not be fully accurate or may become repetitive.
- The system requires GPU support in Google Colab, and running it on CPU can be slow.
- At present, the application only supports concept explanation and quiz generation, meaning it lacks advanced personalization features.
- The platform also does not include authentication or login systems, making access less secure.
- In addition, permanent deployment is not available, as the app only runs within Colab or local environments.
- Automated testing is also missing, which reduces reliability.

ENHANCEMENTS AND FUTURE SCOPE

- EduTutor AI has strong potential for growth and improvement in future versions.
- More interactive learning modules such as videos, flashcards, and summaries can be added to expand its usefulness.

- Adaptive learning paths can be created to provide customized study plans based on student progress.
- Multi-language support can be included to make the system accessible to students from different regions.
- Cloud deployment on platforms like AWS, Azure, or IBM Cloud can provide continuous availability.
- The interface can also be developed into a mobile-friendly app for on-the-go learning.
- Finally, the model can be fine-tuned with domain-specific datasets and fact-checking modules to ensure more accurate and reliable responses.

SETUP INSTRUCTIONS

1. Prerequisites

Before running EduTutor AI, a few prerequisites must be met to ensure smooth execution of the project. First, learners should have a basic understanding of the Gradio framework, since it is used for building the user interface. Access to IBM Granite models on Hugging Face is also required because the project relies on these models for generating explanations and quizzes. Additionally, Python programming proficiency is necessary to write, run, and modify the code as needed. Familiarity with Git and version control systems is recommended for managing project files and

collaborating with teammates. Finally, users should understand how to work with Google Colab GPU (T4), since GPU acceleration is important for faster performance.

2. Installation Process

To begin the installation, open Google Colab and create a new notebook where the project will be developed and executed. The runtime must then be changed to GPU → T4 GPU, which allows the model to run efficiently without slowing down. After this, the required libraries should be installed using pip commands, such as !pip install transformers torch gradio -q. These libraries provide the essential tools for loading models, running computations, and creating the interface. Once the libraries are installed, the provided code cells can be executed for model loading and application launch, ensuring that the system is ready for use.

3. Folder Structure

The project is organized in a simple folder structure to make file management easier. The main file is main.ipynb, which is the Google Colab notebook containing all the code and workflow steps. A requirements.txt file is included, listing the necessary libraries so that the environment can be quickly set up on any machine. A README.md file is also present to

provide an overview of the project, including setup instructions and usage guidelines. This clean folder organization ensures that developers and users can easily navigate and understand the structure without confusion.

4. Running the Application

Once the setup is complete, running the application is straightforward. Users need to execute all cells in the Colab notebook step by step, starting with library installations and model loading. After execution, Colab generates a link to the Gradio application interface, which appears in the output window. By clicking this link, the EduTutor AI system can be opened in a new browser tab, where the user can interact with the tutor. This simple process ensures that even users with minimal technical knowledge can run the application without issues.

5. Frontend (Gradio UI)

The frontend of EduTutor AI is automatically created using the Gradio framework, which is designed to make user interfaces simple and interactive. With this, students can easily type their questions or select topics and receive answers instantly. The interface requires no additional coding or manual design, making it accessible to beginners. Since Gradio runs in the browser, it

eliminates the need for local installations. This interactive frontend ensures that students focus more on learning and less on navigating technical complexities.

6. Backend (IBM Granite Model)

The backend of EduTutor AI is powered by the IBM Granite model hosted on Hugging Face. Specifically, the project uses granite-3.2-2b-instruct, a lightweight but fast model suitable for real-time educational use. The model is integrated into the Colab notebook and connected to Gradio so that responses are automatically generated when a student enters a query. By using this efficient backend, EduTutor provides both concept explanations and quiz generation features. The backend ensures that all requests are processed smoothly and quickly, making the tutor reliable for everyday use.

7. API Documentation

The project does not explicitly define separate API endpoints since all interactions are handled directly within the Colab environment using Gradio. The model functions as the backend, and Gradio serves as the communication layer, which means that when students input queries, they are sent directly to the model without the need for extra API calls. This design keeps the system simple and easy to manage, while still

ensuring smooth functionality. If extended in the future, EduTutor could be deployed with FastAPI or Flask APIs for more advanced integrations.

8. Authentication and User Interface

In its current version, EduTutor AI does not include a user authentication system or role-based access control. This means anyone with the Gradio link can open and use the tutor. While this makes the system open and easy to access, it also lacks security features such as login or data privacy. The interface, however, is designed to be user-friendly, with easy navigation for courses, assignments, and quizzes. It includes a dashboard with role placeholders (student, teacher, admin), progress charts, assignment upload options, and even real-time chat features. The design is also mobile-friendly, allowing learners to access the platform anytime and anywhere.

KNOWN ISSUES

1. Model Limitations

The IBM Granite model, while efficient, has certain limitations that affect the quality of responses. At times, the model may generate irrelevant or repetitive outputs that do not fully match the user's query. This can cause confusion for students who rely on the system for clarity.

Additionally, some explanations may not always be factually accurate unless they are manually verified by a teacher or through reference material. This highlights the need for model fine-tuning or integrating fact-checking modules in the future to improve accuracy and reliability.

2. Performance Constraints

EduTutor AI requires GPU acceleration (T4 in Google Colab) to run smoothly. When executed on a CPU-only environment, the system becomes significantly slower, making it impractical for real-time use. Furthermore, when large prompts or lengthy explanations are requested, the model can experience latency or memory limitations. This may result in delayed responses or incomplete outputs, reducing the overall efficiency of the platform. These issues underline the importance of optimizing performance and considering more powerful deployment solutions.

3. Limited Functionality

At its current stage, EduTutor AI supports only two major functions: concept explanation and quiz generation. While these are valuable features, the system lacks advanced personalization options such as adaptive learning paths or progress tracking. This means that students do not yet receive customized study plans

based on their individual performance. As a result, the system functions more as a supportive learning tool rather than a complete personalized tutor. Expanding its functionality in future updates is essential to make it more comprehensive.

4. No Authentication

One major drawback of EduTutor AI is the absence of authentication or role-based access control. Anyone with access to the Gradio link can use the system, which raises concerns about privacy and security. There is currently no login feature to separate students, teachers, or administrators, and user data is not stored securely. While this makes the platform more open and easy to use, it also makes it unsuitable for larger institutions where secure access and data protection are necessary.

5. Deployment Limitations

EduTutor AI currently runs only in Google Colab or local environments, which restricts its availability. Once the Colab session ends, the application stops working, meaning there is no permanent or production-ready hosting available. This limits its usefulness for continuous educational support. Without deployment on cloud platforms such as AWS, Azure, or IBM Cloud, the system cannot be accessed reliably at all times. This

is a major limitation for scalability and long-term adoption.

6. Testing Coverage

So far, EduTutor AI has undergone only basic manual testing to ensure the core features work as expected. While this confirmed that the system is functional, no automated test cases have been developed for the backend model or frontend interface. Automated testing would make it easier to detect bugs, ensure reliability, and verify outputs at scale. The lack of structured testing coverage means that some issues may go unnoticed until they affect the user directly. This reduces the robustness of the current version and highlights the need for more systematic testing in future iterations.

FOLDER STRUCTURE – PROGRAM EXPLANATION

The given code represents the core implementation of EduTutor AI using Python, Hugging Face Transformers, and the Gradio framework.

LIBRARY IMPORT

import gradio as gr from transformers import AutoTokenizer, AutoModelForCausalLM

Here, the required libraries are imported. transformers is used to load and work with the IBM Granite model, while gradio helps in building the interactive user interface.

MODEL LOADING

```
model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)
```

The IBM Granite model (granite-3.2-2b-instruct) is loaded from Hugging Face. The tokenizer converts input text into tokens that the model understands, and the model generates responses based on those tokens.

CONCEPT EXPLANATIONS FUNCTIONS

```
def concept_explanation(concept):
    prompt = f"Explain the concept of {concept} in detail with examples:"
    inputs = tokenizer(prompt, return_tensors="pt")
    outputs = model.generate(**inputs, max_length=300)
    return tokenizer.decode(outputs[0], skip_special_tokens=True)
```

This function takes a concept as input, creates a prompt asking the model to explain it in detail, and then generates an output using the model. The result is decoded back into human-readable text. This is the main logic behind the “concept explanation” feature.

GRADIO USER INTERFACE

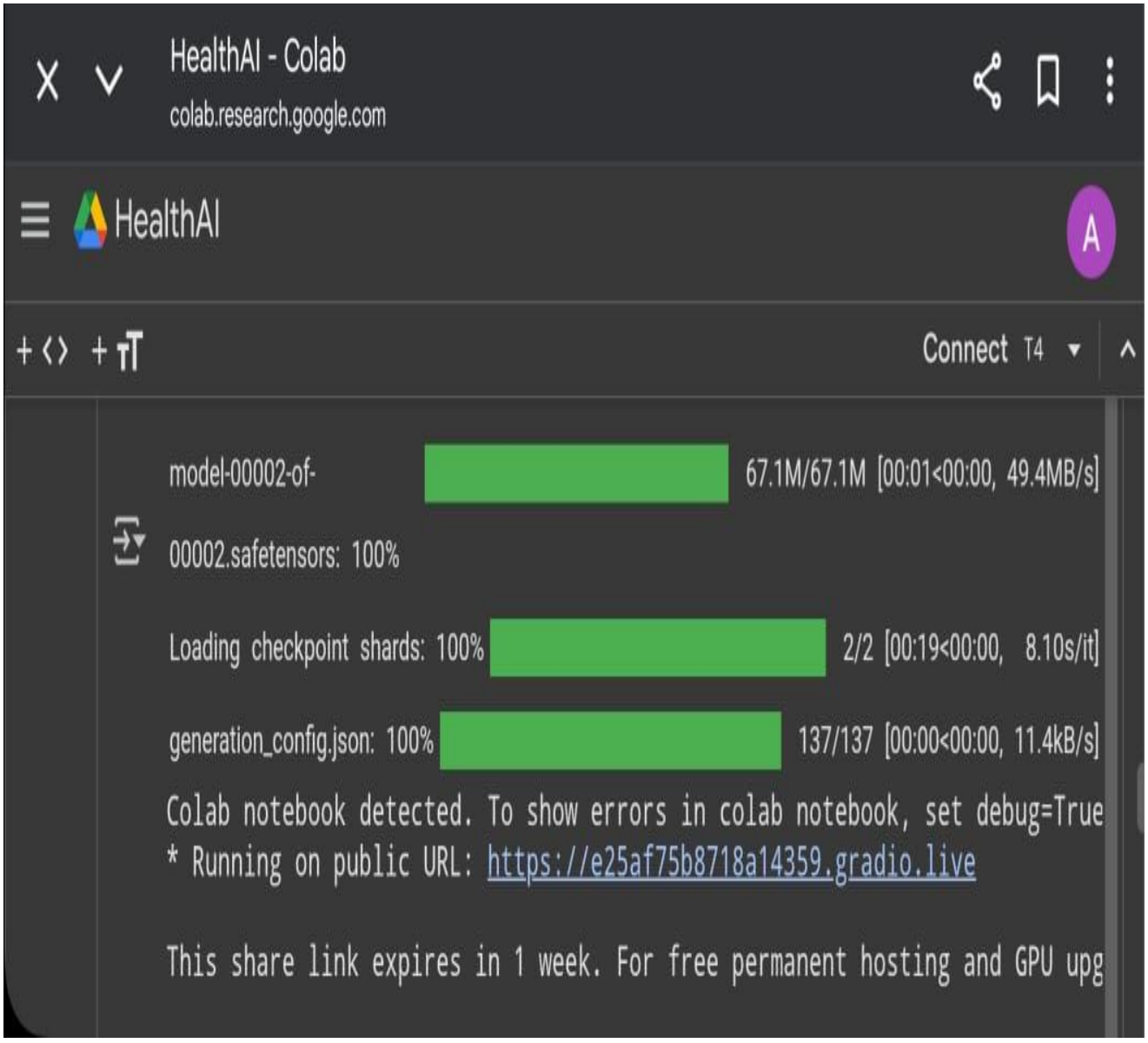
```
with gr.Blocks() as app:
gr.Markdown("# EduTutor AI")
    concept = gr.Textbox(label="Enter a concept")
    output = gr.Textbox(label="Explanation")
    gr.Button("Explain").click(concept_explanation, concept, output)
app.launch()
```

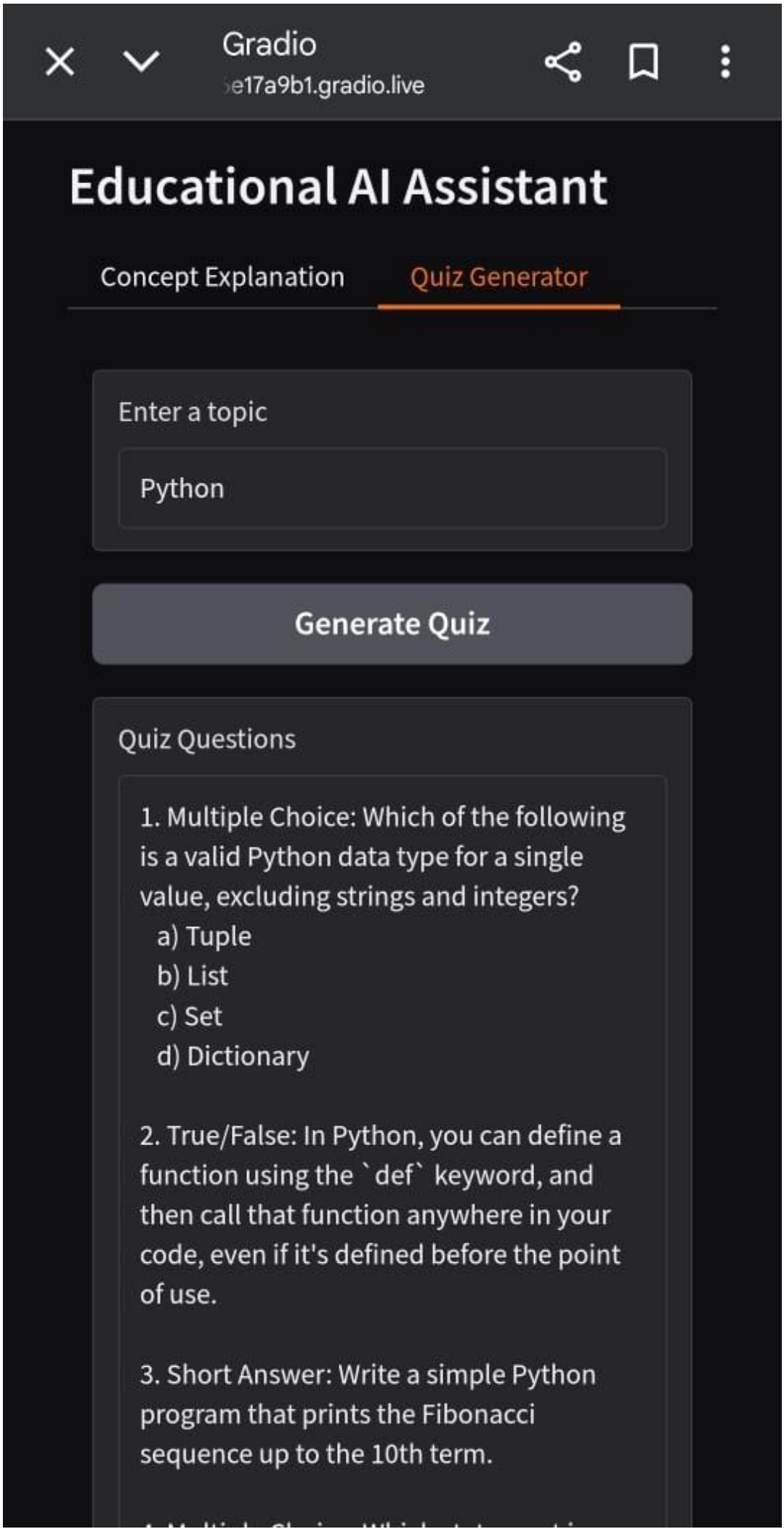
This part creates a simple user interface using Gradio. The interface includes a textbox for entering the concept, another textbox for displaying the explanation, and a button labeled “Explain.” When the button is clicked, the concept_explanation function is triggered, and the result is displayed in the output textbox.

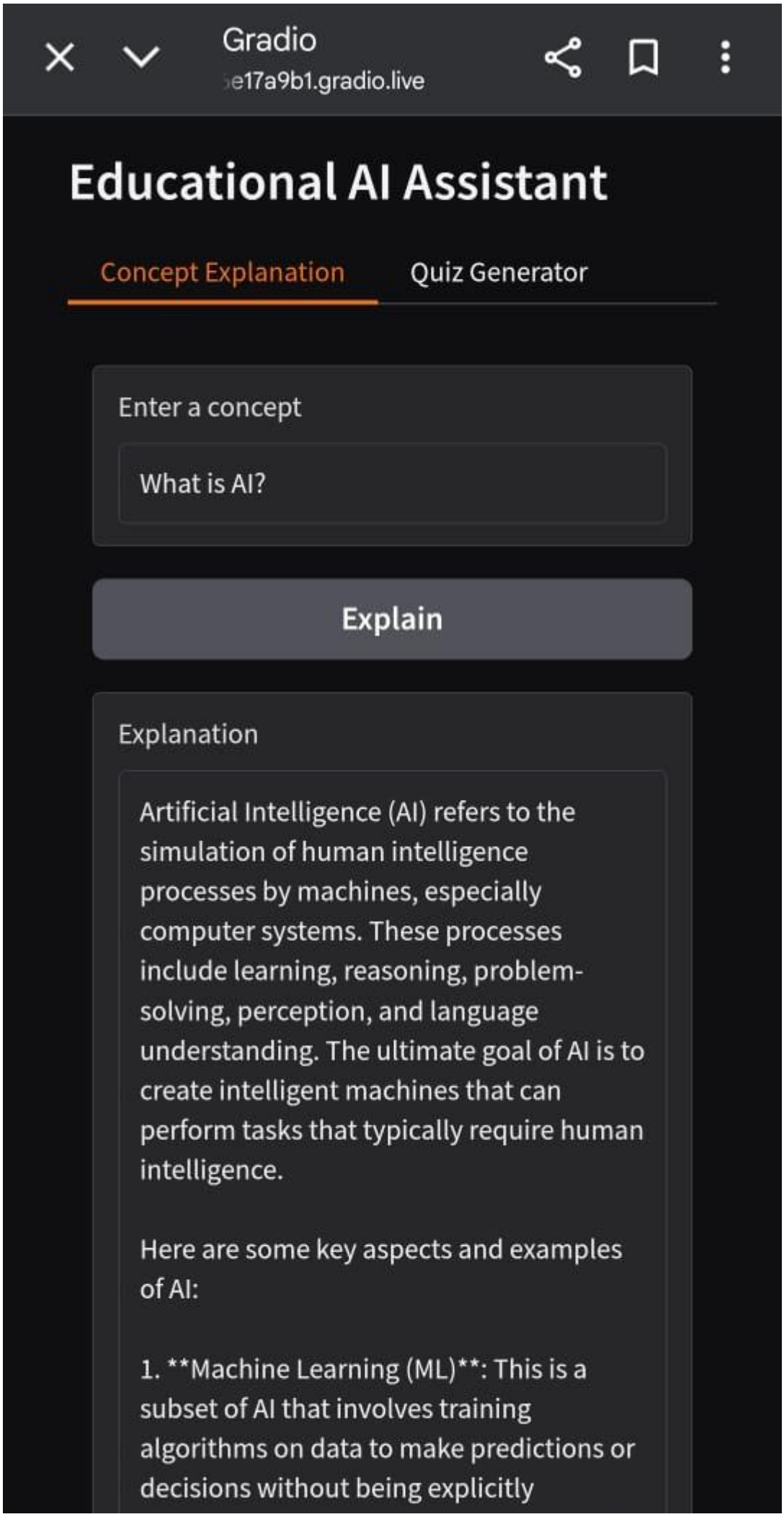
SUMMARY

- In short, this program loads the IBM Granite AI model, defines a function to explain concepts, and creates an interactive UI using Gradio.
- Students can enter any concept, and the system generates a detailed explanation with examples.
- This simple yet powerful design forms the backbone of EduTutor AI.

SCREENSHOTS




Gradio
e17a9b1.gradio.live
Educational AI Assistant
Concept Explanation
Quiz Generator
Enter a topic
Python
Generate Quiz
Quiz Questions
1. Multiple Choice: Which of the following is a valid Python data type for a single value, excluding strings and integers?
a) Tuple
b) List
c) Set
d) Dictionary
2. True/False: In Python, you can define a function using the `def` keyword, and then call that function anywhere in your code, even if it's defined before the point of use.
3. Short Answer: Write a simple Python program that prints the Fibonacci sequence up to the 10th term.


Gradio
e17a9b1.gradio.live
Educational AI Assistant
Concept Explanation
Quiz Generator
Enter a concept
What is AI?
Explain
Explanation
Artificial Intelligence (AI) refers to the simulation of human intelligence processes by machines, especially computer systems. These processes include learning, reasoning, problem-solving, perception, and language understanding. The ultimate goal of AI is to create intelligent machines that can perform tasks that typically require human intelligence.
Here are some key aspects and examples of AI:
1. **Machine Learning (ML)**: This is a subset of AI that involves training algorithms on data to make predictions or decisions without being explicitly