Edu Tutor AI:Personalized Learning

Project Documentation

1.Introduction:

- Project title: Edu Tutor AI:Personalized Learning
- Team member: Deepika S
- Team member: Durga devi T
- Team member: Devi Priya.S
- Team member: Divya.D

2. Project Overview:

Purpose:

The purpose of this project is to build an educational AI application that assists students and learners by using artificial intelligence to make studying more interactive and effective. It helps in understanding complex concepts by providing clear explanations and also supports learning through quizzes generated automatically on different topics. This system reduces the manual effort of teachers in preparing study material and quizzes, while also helping learners quickly access relevant knowledge. Overall, the project improves productivity, saves time, and makes learning more engaging by serving as a smart educational assistant.

Features:

- **Concept Explanation:**

Users can input any concept, and the system will generate a clear and detailed explanation, making it easier to understand.

- **Quiz Generation:**

The system can automatically generate quiz questions based on the given topic, which helps learners in self-assessment and practice.

- **User Interface:**

The interface is simple and user-friendly, with two tabs that is one for concept explanation and the other for quiz generation.

- **Customizable:**

Users can provide any concept or topic of their choice, and the system generates explanations or quizzes accordingly.

- **Saves Time:**

It quickly provides study material and quizzes, reducing the manual effort for both students and teachers.

3. Architecture:

- **Frontend (Gradio):**

The frontend is developed using Gradio, which provides a simple and interactive web-based interface. Users can enter concepts or topics, and the system will display either explanations or quiz questions.The interface is organized into two main tabs:

1. Concept Explanation Tab – For entering a concept and generating a detailed explanation.
2. Quiz Generator Tab – For entering a topic and generating quiz questions automatically.

- **Backend (Google Colab + Python):**

The backend runs on Google Colab, where the Python environment handles model loading, request processing, and interaction with the AI model. It processes inputs, communicates with the IBM Granite model, and returns structured results back to the Gradio interface.

- **LLM Integration (IBM Granite):**

The project integrates the IBM Granite LLM for natural language understanding and generation. The model is responsible for providing detailed explanations of concepts and generating quiz questions based on topics entered by the user.

4. Setup Instructions:

Prerequisites:

- A Google account to access Google Colab.
- A stable internet connection to install models and libraries from the cloud.
- A Hugging Face account for loading the IBM Granite model.

Installation Process:

- Open Google Colab.
- Create a new notebook.
- Change the runtime to T4-GPU.
- In the first cell, install the required libraries.
- After installation, copy the project code into the next cell and run it.

5. Folder Structure:

Since the project runs entirely in Google Colab, there is only one main file used: educationalAI.ipynb

Structure:

project/

└── educationalAI.ipynb

6. Running the Application:

- Execute the code cell with app.launch(share=True).
- Colab will display a public Gradio link like:

(Running on public URL: https://xxxxx.gradio.live)

- Click the link to open the web app in a new tab.
- The interface contains two main functionalities:

Concept Explanation → Enter a concept → Click Explain → Get a detailed explanation.

- Quiz Generation → Enter a topic → Click Generate Quiz → AI generates quiz questions.

7. API Documentation:

Backend APIs available include:

POST /explain-concept – Accepts a concept as input and responds with an AI-generated explanation.

POST /generate-quiz – Accepts a topic and generates quiz questions related to the input.

POST /upload-text – Allows users to upload raw text for processing and explanation.

Each endpoint is tested and documented within the Colab environment for quick inspection and trial during development.

8. User Interface:

- **Google Colab Notebook** – The entire project runs inside a single Colab notebook with a Gradio interface.

- Two Tabs in Gradio are:
  - **Concept Explanation** – In this tab, users can enter a concept, and the AI generates a clear and detailed explanation.
  - **Quiz Generator** – In this tab, users enter a topic, and the AI generates relevant quiz questions for practice.
- **Textboxes for Input/Output** – Simple boxes are provided to type the concept/topic and to display the AI-generated explanations or quiz questions.
- **Buttons to Trigger Actions** – "Explain" and "Generate Quiz" buttons execute the AI model and show the results.
- **Public/Local Link** – Gradio provides a shareable link so that the interface can be accessed from any browser or device.

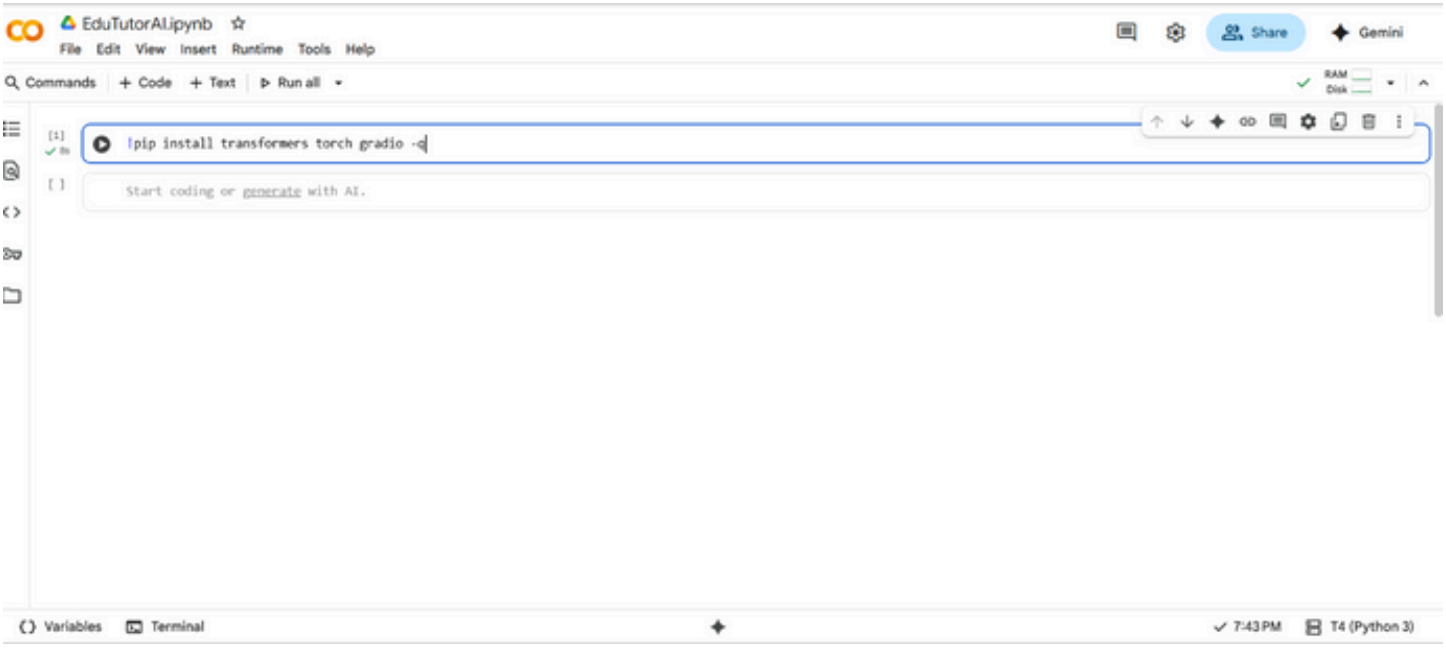
9. Testing:

- **Unit Testing:** For individual functions like generate_response() to ensure correct AI outputs.
- **API Testing:** By sending test prompts through Gradio's backend and validating responses.
- **Manual Testing:** For verifying input/output quality, checking explanation accuracy, and quiz relevance.
- **Edge Case Handling:** Tested with empty inputs, long prompts, irrelevant queries, and unusual topics to ensure robust handling.

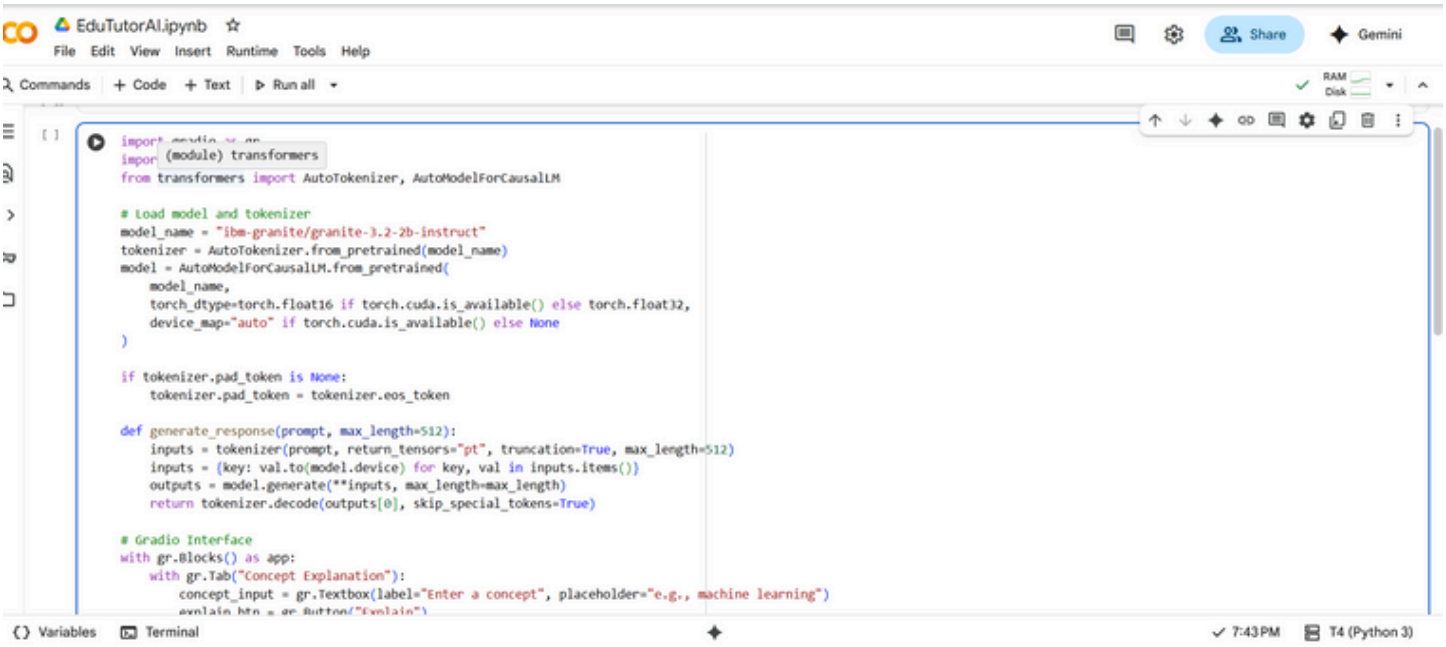
12.Screenshots:

Program:

First cell,


EduTutorAI.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
!pip install transformers torch gradio -q
Start coding or generate with AI.
Variables
Terminal
7:43 PM
T4 (Python 3)

Second cell,


EduTutorAI.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
(module) transformers
from transformers import AutoTokenizer, AutoModelForCausalLM
# Load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
model_name,
torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
device_map="auto" if torch.cuda.is_available() else None
)
if tokenizer.pad_token is None:
tokenizer.pad_token = tokenizer.eos_token
def generate_response(prompt, max_length=512):
inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)
inputs = {key: val.to(model.device) for key, val in inputs.items()}
outputs = model.generate(**inputs, max_length=max_length)
return tokenizer.decode(outputs[0], skip_special_tokens=True)
Gradio Interface
with gr.Blocks() as app:
with gr.Tab("Concept Explanation"):
concept_input = gr.Textbox(label="Enter a concept", placeholder="e.g., machine learning")
Variables
Terminal
7:43 PM
T4 (Python 3)

EduTutorAI.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Run all

RAM Disk

```
    tokenizer.pad_token = tokenizer.eos_token

def generate_response(prompt, max_length=512):
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)
    inputs = {key: val.to(model.device) for key, val in inputs.items()}
    outputs = model.generate(**inputs, max_length=max_length)
    return tokenizer.decode(outputs[0], skip_special_tokens=True)

# Gradio Interface
with gr.Blocks() as app:
    with gr.Tab("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(generate_response, inputs=concept_input, outputs=explanation_output)

    with gr.Tab("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(generate_response, inputs=quiz_input, outputs=quiz_output)

app.launch(share=True)
```

Variables Terminal

7:43 PM T4 (Python 3)

EduTutorAI.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Run all

RAM Disk

```
app.launch(share=True)
```

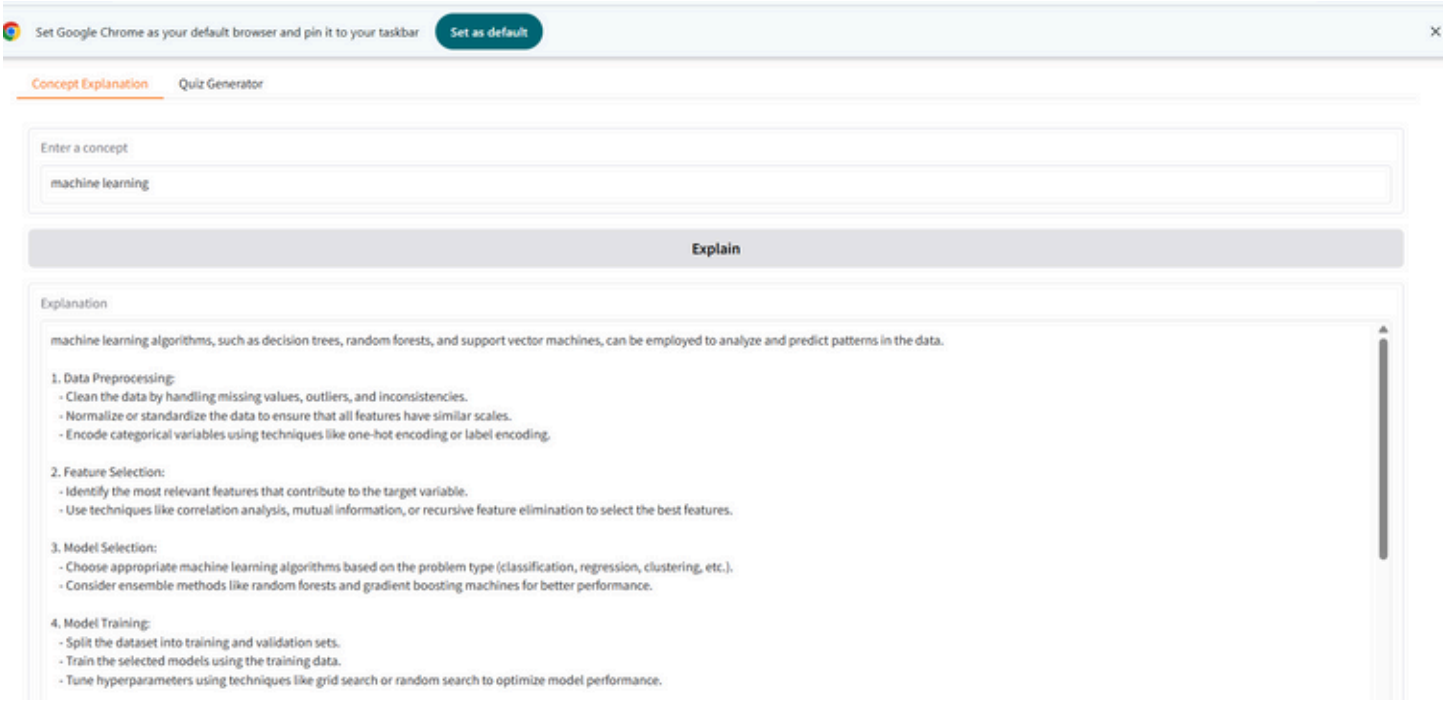
```
vocab.json: 777k/? [00:00<00:00, 12.4MB/s]
merges.txt: 442k/? [00:00<00:00, 25.1MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 94.7MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 3.49kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 33.5kB/s]
config.json: 100% 786/786 [00:00<00:00, 31.9kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 3.05MB/s]
Fetching 2 files: 100% 2/2 [02:31<00:00, 151.02s/it]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [02:30<00:00, 57.0MB/s]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:05<00:00, 11.3MB/s]
Loading checkpoint shards: 100% 2/2 [00:22<00:00, 9.34s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 9.20kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://1782cbb9c4cc17e737.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 (
```

Variables Terminal

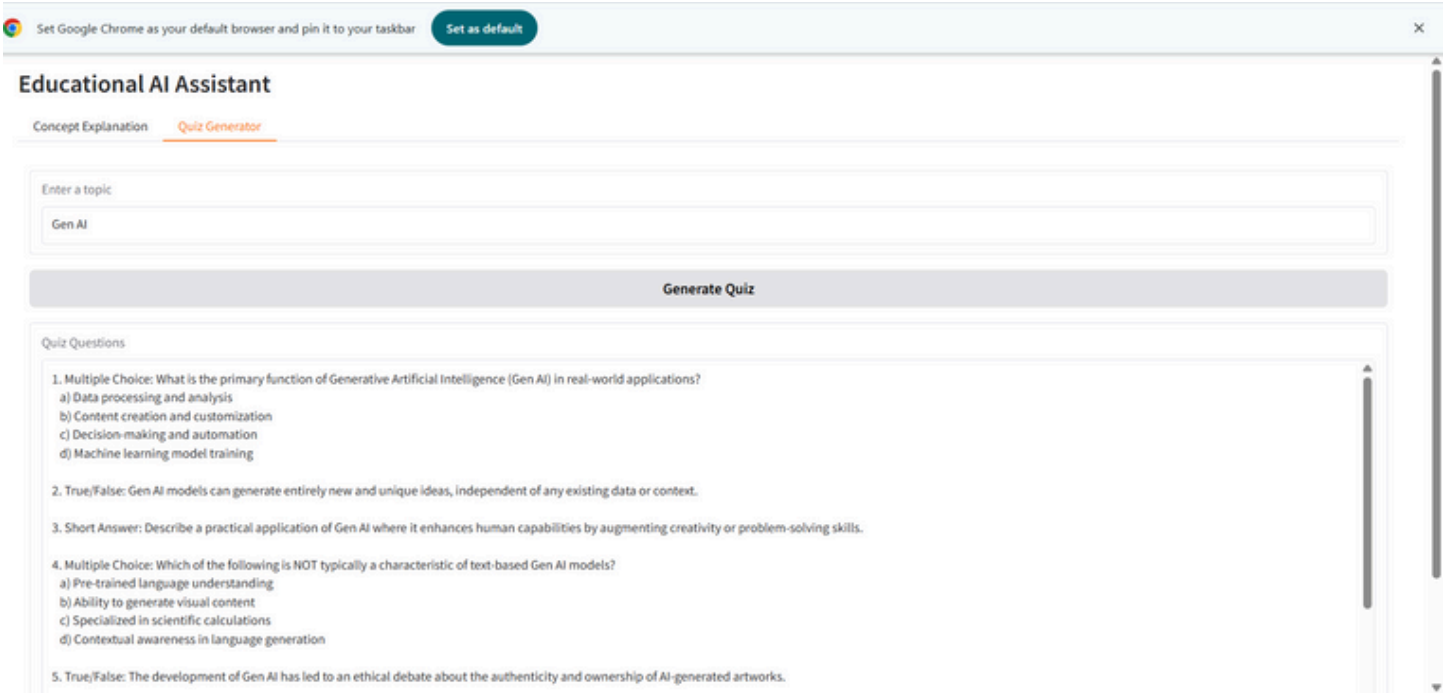
7:54 PM T4 (Python 3)

Output:

First view of the screen, **Concept Explanation.**



Second view of the screen, **Quiz Generator.**



10. Known Issues:

- May run slowly if internet connection is weak or model size is large.
- Sometimes generates explanations or quiz questions that are inaccurate or incomplete.
- No login or authentication, so anyone with the Gradio link can access the application.
- Limited to features supported by the IBM Granite model.
- Requires Google Colab to run, so it cannot work offline.

11. Future Enhancements:

Some of the future enhancements that can be added are:

- Add user authentication (login system) for better security.
- Provide a download option to save generated explanations or quizzes.
- Improve the AI model to give more accurate and detailed results.
- Add real-time collaboration, so multiple learners or teachers can use it together.
- Allow direct deployment as a standalone website or mobile app for easier access.
- Improve UI with better formatting, layouts, and themes.
- Add support for more input formats like DOCX and TXT files.