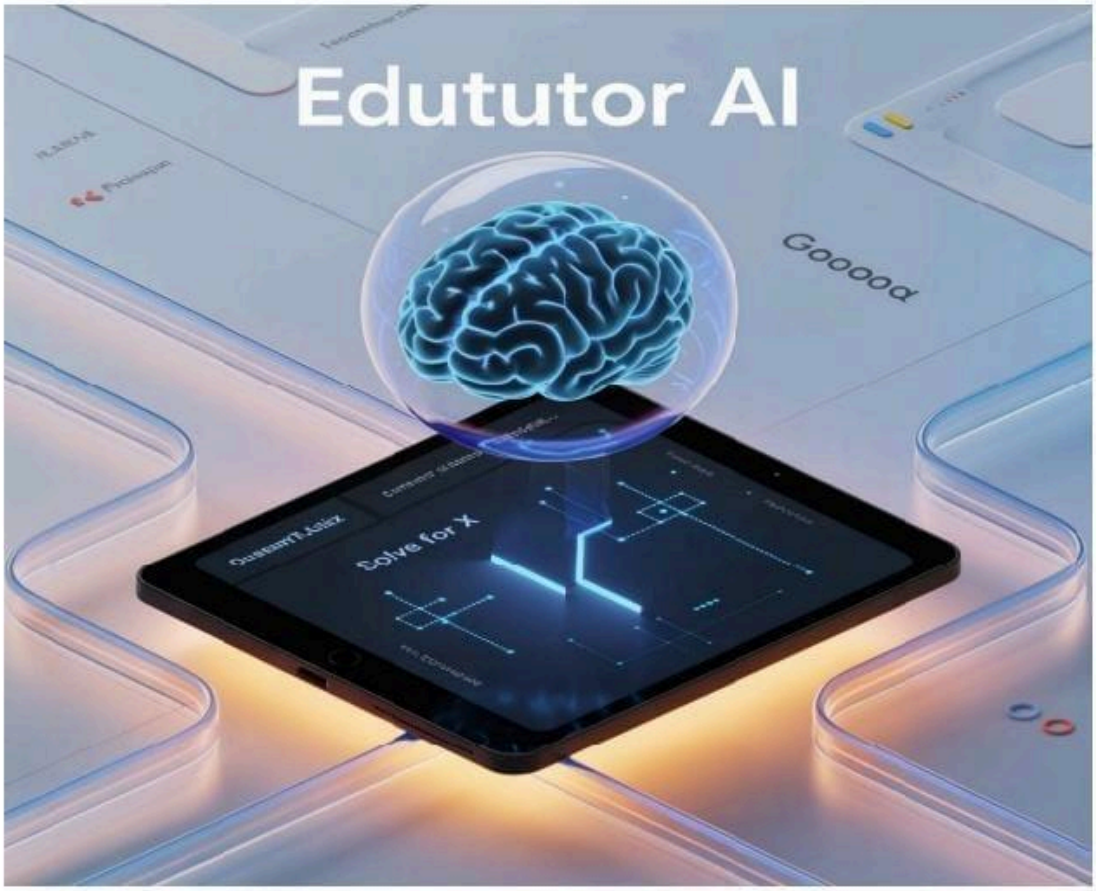
Edu Tutor AI: Personalized Learning

Generative AI with IBM



EDU TUTOR AI

TEAM MEMBERS

- Preethi S
- Sharmila P
- Mahalakshmi M
- Shamitha G K

1. PROJECT OVERVIEW

PURPOSE:

Edututor is an intelligent e-learning assistant designed to personalize education for students and support teachers with smart tools. By leveraging AI and interactive modules, Edututor enhances learning experiences through adaptive lessons, real-time feedback, and engaging quizzes. For educators, it provides analytics dashboards, automated grading support, and resource recommendations. Ultimately, Edututor bridges technology and education to make learning more effective, inclusive, and accessible.

FEATURES:

Conversational Learning: Students interact using natural language to ask questions, clarify

doubts, and receive explanations.

Personalized Lessons: Adaptive content tailored to student performance and learning speed.

Quiz & Assessment Generator: Automatic creation of quizzes with instant feedback.

Analytics Dashboard: Provides teachers insights into student progress and areas needing improvement.

Resource Recommendation: Suggests learning materials, videos, and practice exercises.

Collaborative Tools: Group projects and peer-to-peer interaction support.

2. ARCHITECTURES

Frontend (React/Streamlit): Provides an interactive UI for students and teachers.

Backend(FastAPI/Django): Handles APIs for quizzes, analytics, and recommendations.

AI Models: Powers personalization, grading, and adaptive learning using NLP and ML.

Database: Stores user progress, resources, and activity logs securely.

3. SETUP INSTRUCTIONS

Choose a IBM Granite Model from Hugging Face.
Using “granite 3.2-instruct” .
Running Application in Google collab.

4 .RUNNING

Start backend server ➢ Launch frontend dashboard ➢ Register/login as student or teacher ➢
Upload/assign resources and quizzes ➢ Track progress via analytics dashboard.

5. APPLICATIONS

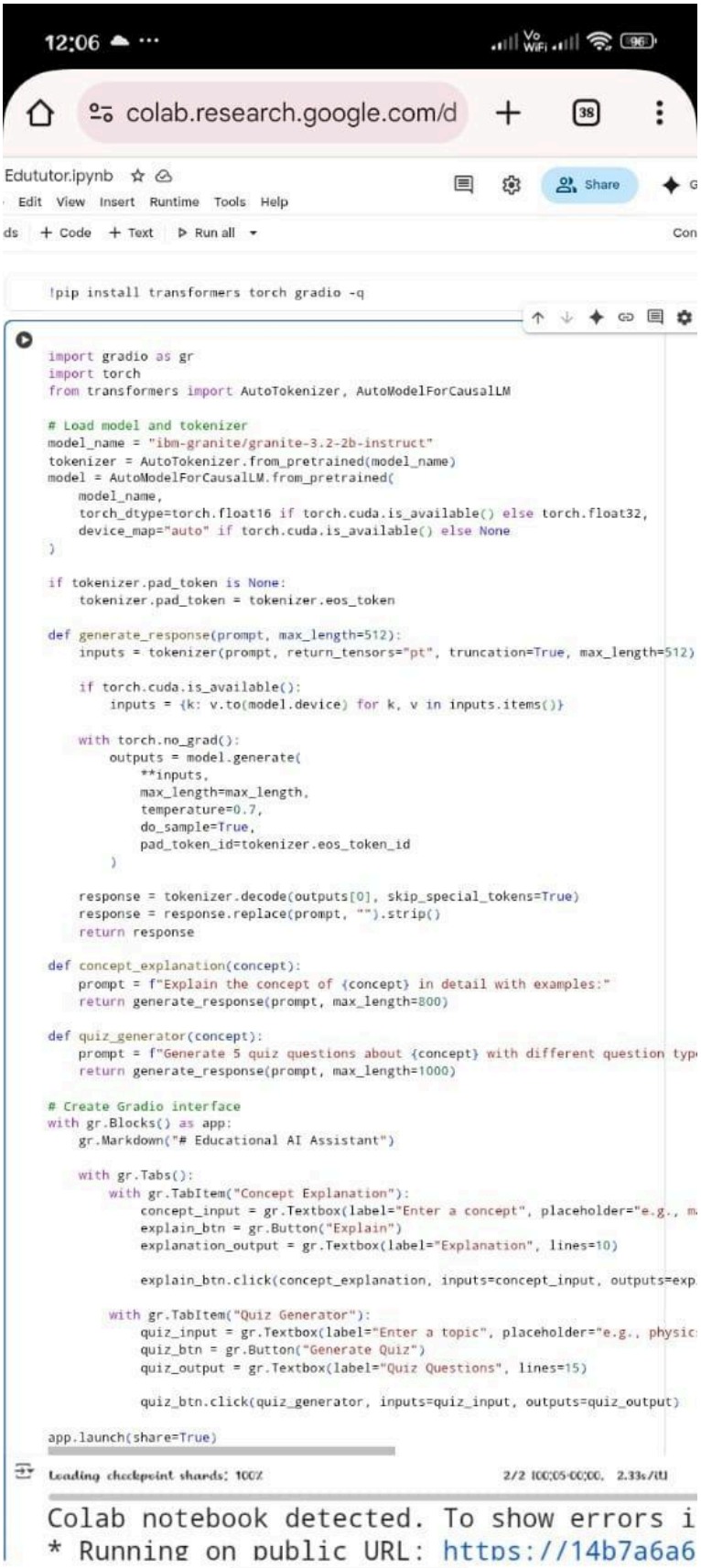
- **Student-friendly dashboard**
- **AI-powered doubt assistant**
- **Smart quiz and test creation**
- **Analytics for progress tracking**

6. FUTURE ENHANCEMENTS

- **Multilingual support for global education**
- **Integration with AR/VR for immersive learning**
- **Adaptive learning paths using AI-driven insights**

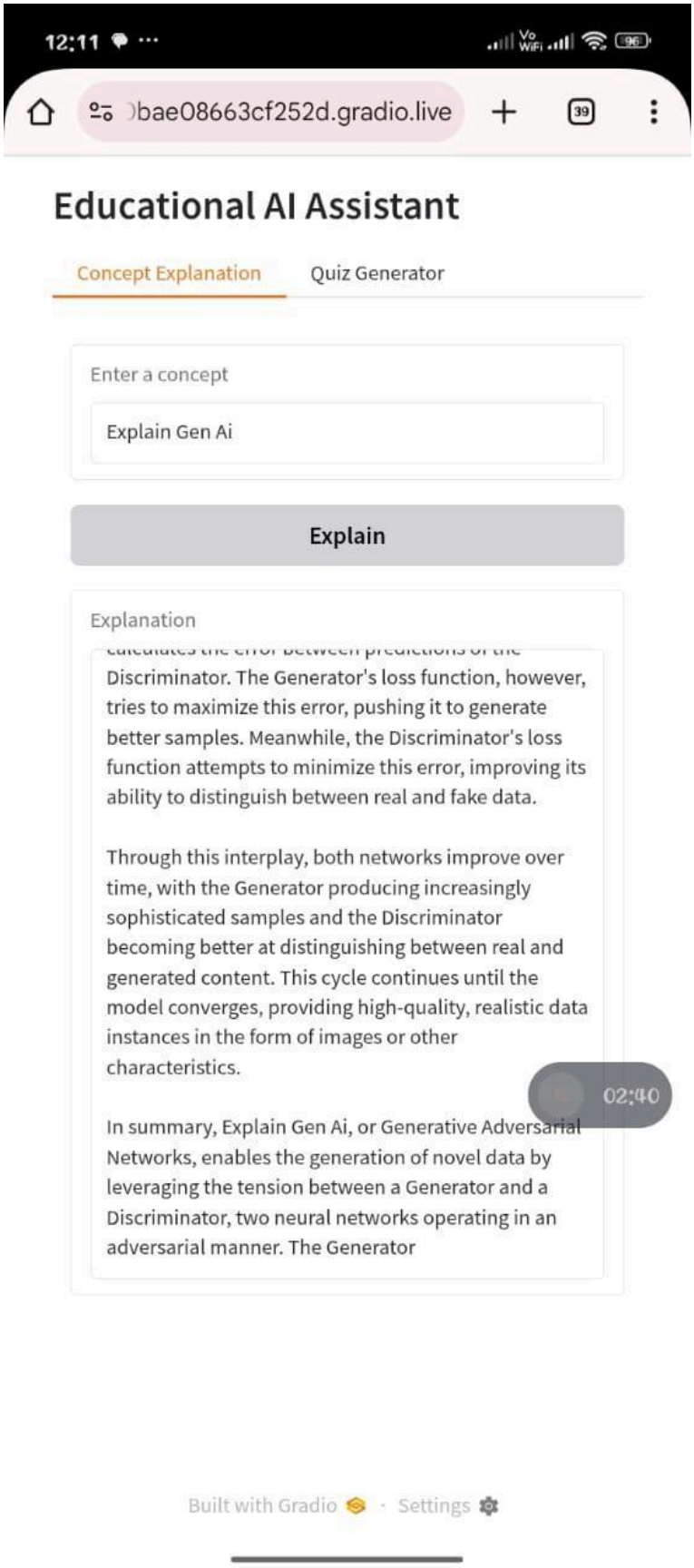
* Teacher support with automated grading and content suggestion

7. CODE SCREENSHOTS


12:06
colab.research.google.com/d
Edututor.ipynb
Edit View Insert Runtime Tools Help
Share
+ Code + Text Run all
!pip install transformers torch gradio -q
import gradio as gr
import torch
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)
if torch.cuda.is_available():
inputs = {k: v.to(model.device) for k, v in inputs.items()}
with torch.no_grad():
outputs = model.generate(
**inputs,
max_length=max_length,
temperature=0.7,
do_sample=True,
pad_token_id=tokenizer.eos_token_id
)
response = tokenizer.decode(outputs[0], skip_special_tokens=True)
response = response.replace(prompt, "").strip()
return response
def concept_explanation(concept):
prompt = f"Explain the concept of {concept} in detail with examples:"
return generate_response(prompt, max_length=800)
def quiz_generator(concept):
prompt = f"Generate 5 quiz questions about {concept} with different question typ
return generate_response(prompt, max_length=1000)
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., m
explain_btn = gr.Button("Explain")
explanation_output = gr.Textbox(label="Explanation", lines=10)
explain_btn.click(concept_explanation, inputs=concept_input, outputs=exp
with gr.TabItem("Quiz Generator"):
quiz_input = gr.Textbox(label="Enter a topic", placeholder="e.g., physic
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)
Loading checkpoint shards: 100%
2/2 [00:05<00:00, 2.33s/it]
Colab notebook detected. To show errors i
* Running on public URL: https://14b7a6a6

8. OUTPUT


12:10
colab.research.google.com/d
Edututor.ipynb
Edit View Insert Runtime Tools Help
Share
Gemini
+ Code + Text Run all
pad_token_id=tokenizer.eos_token_id
response = tokenizer.decode(outputs[0], skip_special_tokens=True)
response = response.replace(prompt, "").strip()
return response
def concept_explanation(concept):
prompt = f"Explain the concept of {concept} in detail with examples:"
return generate_response(prompt, max_length=800)
def quiz_generator(concept):
prompt = f"Generate 5 quiz questions about {concept} with different question types (multiple
return generate_response(prompt, max_length=1000)
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 learni
explain_btn = gr.Button("Explain")
explanation_output = gr.Textbox(label="Explanation", lines=10)
explain_btn.click(concept_explanation, inputs=concept_input, outputs=explanation_outp
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)
special_tokens_map.json: 100%
701/701 [00:00<00:00, 31.3kB/s]
config.json: 100%
786/786 [00:00<00:00, 45.6kB/s]
`torch_dtype` is deprecated! Use `dtype` instead
model.safetensors.index.json:
29.5k/? [00:00<00:00, 2.62MB/s]
Fetching 2 files: 100%
2/2 [01:43<00:00, 103.67s/it]
model-00002-of-
00002.safetensors: 100%
67.1M/67.1M [00:00<00:00, 18.1MB/s]
model-00001-of-
00002.safetensors: 100%
5.00G/5.00G [01:43<00:00, 72.4MB/s]
Loading checkpoint shards: 100%
2/2 [00:19<00:00, 8.25s/it]
generation_config.json: 100%
137/137 [00:00<00:00, 18.3kB/s]
Colab notebook detected. To show errors in colab
* Running on public URL: https://ade0bae08663cf2
This share link expires in 1 week. For free perm


12:11
bae08663cf252d.gradio.live
39
Educational AI Assistant
Concept Explanation
Quiz Generator
Enter a concept
Explain Gen Ai
Explain
Explanation
Discriminator. The Generator's loss function, however, tries to maximize this error, pushing it to generate better samples. Meanwhile, the Discriminator's loss function attempts to minimize this error, improving its ability to distinguish between real and fake data.
Through this interplay, both networks improve over time, with the Generator producing increasingly sophisticated samples and the Discriminator becoming better at distinguishing between real and generated content. This cycle continues until the model converges, providing high-quality, realistic data instances in the form of images or other characteristics.
02:40
In summary, Explain Gen Ai, or Generative Adversarial Networks, enables the generation of novel data by leveraging the tension between a Generator and a Discriminator, two neural networks operating in an adversarial manner. The Generator
Built with Gradio
Settings

9. CODE LINK

https://github.com/preethiammu2005/Digital-protofolio/blob/main/IBM%20PROJECT

10. DEMO VIDEO

https://drive.google.com/file/d/1COfVPyVjr37E9MN2cF74WCQrZ769W2h0/view?usp=drivesdk