EduTutor AI – Intelligent Learning Assistant

Project Report

TEAM LEADER: NIVEDHA N

TABLE OF CONTENTS

1.Abstract

This project presents EduTutor AI - an intelligent educational assistant built using Streamlit,IBM WatsonX AI. Leveraging IBM Granite language models, the system provides concept explanations and generates quizzes dynamically for diverse academic topics. The tool is designed to support students, educators, and lifelong learners by offering clear, contextual explanations as well as self-assessment through quizzes. With a user-friendly web interface powered by Gradio, EduTutor AI promotes interactive and personalized learning experiences.

2.Objectives

- To provide on-demand concept explanations in simple, detailed language with example.
- To dynamically generate quizzes for learners, including multiple formats .
- To create a user-friendly AI assistant interface accessible via the web.
- To utilize transformer-based LLMs for ensuring contextual, accurate, and high-quality educational content.

- To support self-paced learning and improve student engagement through interactive AI tools.

3.System Requirements

Hardware Requirements:

Minimum:

- CPU: Quad-core
- RAM: 8GB
- Storage: 10GB free

Software Requirements:

Programming Language: Python 3.9+

Libraries/Frameworks:

- transformers (for LLMs)
- torch (PyTorch for model inference)
- gradio (for web UI)

4.Implementation

- The system uses IBM Granite 3.2-2B Instruct model, loaded through Hugging Face's transformers. The tokenizer and model are initialized with device compatibility .
- Takes a user-entered concept, generates a detailed explanation with examples.
- Accepts a topic and produces 5 mixed-type questions (MCQs, True/False, short answer) along with an ANSWER section for self-evaluation.
- Uses model.generate() with parameters like temperature=0.7 for creativity, max_length for limiting output size, and do_sample=True for varied responses.
- Post-processes the text to remove prompt echoes and keep clean output.
- Developed using Gradio Blocks & Tabs, providing two sections: Concept Explanation and Quiz Generator.
- Learners can input a topic/concept and instantly receive explanations/quizzes in a structured format.

- The application is launched locally with app.launch(share=True), enabling web access via a public Gradio link.

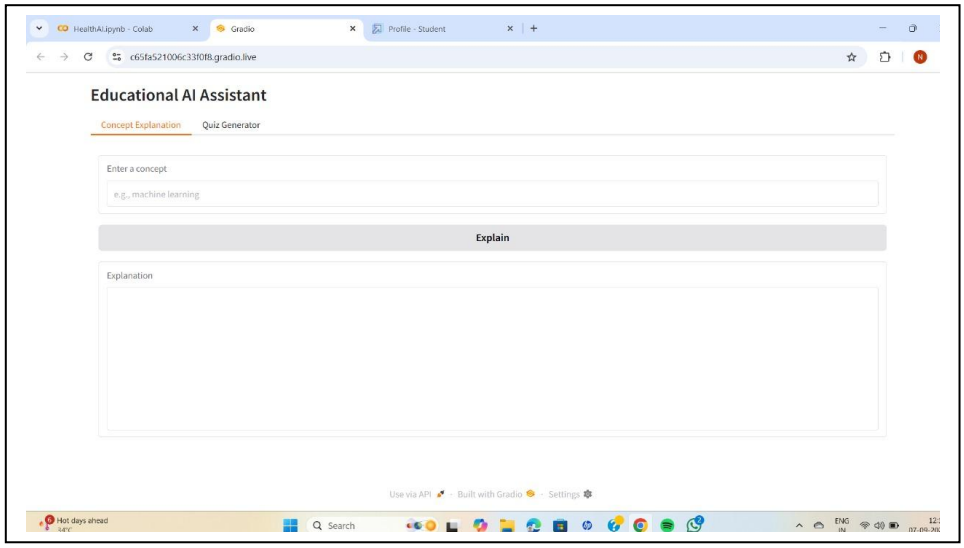
5.Appendix: Source Code

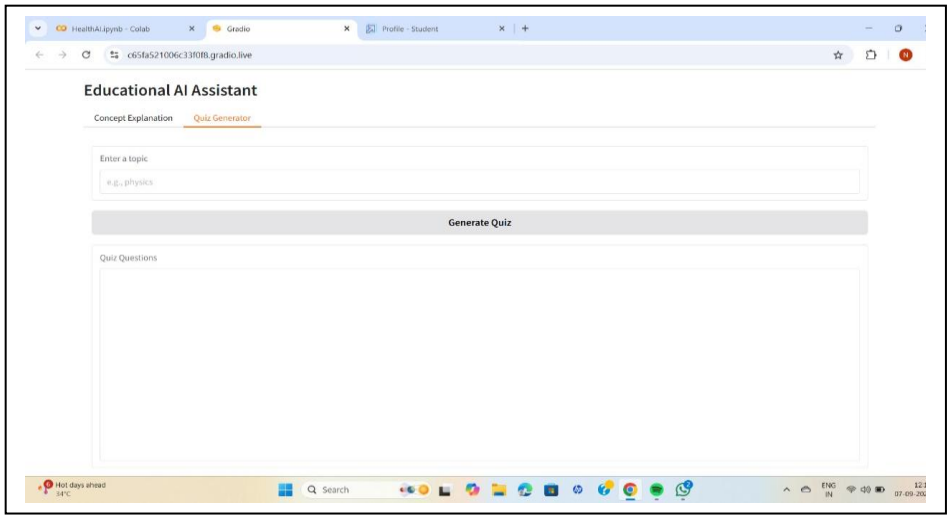
```
import gradio as gr

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 types (multiple choice, true/false, short answer). At the end, provide all the answers in a separate ANSWERS section:"
    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 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)
```

6.Screenshots





7.Conclusion

EduTutor AI demonstrates the effective use of large language models as personalized learning assistants. Its dual features of concept tutoring and quiz generation make it a versatile academic aid for students across disciplines. By combining generative AI with an interactive interface, the system offers both knowledge reinforcement and self-assessment opportunities. Future enhancements may include multilingual support, adaptive difficulty levels for quizzes, and integration with educational platforms for broader classroom use.