PROJECT DOCUMENTATION

PROJECT TITLE

Edu Tutor AI: Personalized Learning

1. Introduction

Project Title : **EduTutor AI: Personalized Learning Assistant**

Team Leader : SAMSON JEBAKUMAR S

Team Members: SATHYANARAYANAN CP

Team Members: LOKESH S

Team Members: DEEPAK KUMAR S

2. Project Overview

- **Purpose:**

The purpose of EduTutor AI is to create a simple, accessible, and personalized AI learning assistant that explains complex concepts and generates quizzes, making education more engaging and interactive.

- **Importance:**

Traditional learning methods can be static and less adaptive to individual learners. EduTutor AI enhances accessibility, ensures personalized learning, and supports both teachers and students through AI-generated content.

3. Problem Statement

Challenges:

- Students often struggle to understand complex topics without detailed explanations.
- Teachers need tools to generate quizzes quickly for assessments.
- Existing platforms are either costly or lack personalization.

Need:

❖ A user-friendly educational assistant that explains concepts with examples and generates quizzes instantly using AI models.

4. Objectives of the Project

- To build an AI-powered assistant for education using IBM Granite models.
- To provide detailed concept explanations with examples.
- To generate quizzes with multiple question types.
- To design a user-friendly interface using Gradio in Google Colab.
- To deploy the application for easy access by students and teachers.

5. Literature Review

AI in Education:

Artificial Intelligence supports personalized learning, content generation, and adaptive tutoring systems.

NLP in Learning Tools:
Natural Language Processing models simplify complex concepts and create interactive learning materials.

Existing Solutions:
Many e-learning platforms exist, but they lack domain-specific AI support and quiz generation capabilities tailored for flexible learning.

6. Methodology

Tools & Technologies:

- Language: Python
- Framework: Gradio
- Library: HuggingFace Transformers
- Model: IBM Granite 3.2B Instruct
- Platform: Google Colab (T4 GPU)
- Version Control: GitHub

Workflow:

1. User enters a concept or topic.
2. Model processes the input and generates explanations or quizzes.
3. Output is displayed via Gradio interface.
4. Application can be shared through Colab or deployed.

7. System Architecture

[User Input] → [IBM Granite Model] → [AI Output: Explanation/Quiz] → [Gradio Interface]

8. Implementation

8.1 **Concept Explanation**

Input: User enters a concept.

Output: AI generates detailed explanation with examples.

8.2 Quiz Generator Input:

User enters a topic.

Output: AI generates 5 quiz questions (MCQ, True/False, Short Answer) with answers.

8.3 Gradio Interface

Two tabs are created:

- Concept Explanation
- Quiz Generator

9. Sample Outputs

Example 1: Concept Explanation

Input: Machine Learning

Output: AI-generated detailed explanation with real-world examples.

Example 2: Quiz Generator

Input: Physics

Output: 5 quiz questions of different types along with answers.

10. Results & Discussion

EduTutor AI successfully provides meaningful concept explanations and quiz questions. It helps students understand topics better and allows teachers to generate assessments quickly.

11. Advantages

- User-friendly and interactive.
- Uses cloud-based IBM Granite model.
- Reduces teacher workload by automating quiz creation.
- Helps students learn concepts in detail.

12. Limitations

- Requires internet and Colab for execution.
- May occasionally generate repetitive content.
- Accuracy depends on the model performance.

13. Future Enhancements

- Add voice-based query support.
- Integrate gamified learning features.
- Provide multilingual explanations.
- Enable offline access.
- Deploy as a mobile or web app.

14. Conclusion

EduTutor AI demonstrates how Generative AI can enhance education by offering personalized learning experiences. It provides detailed explanations and quiz generation features, making it a valuable tool for both students and teachers.

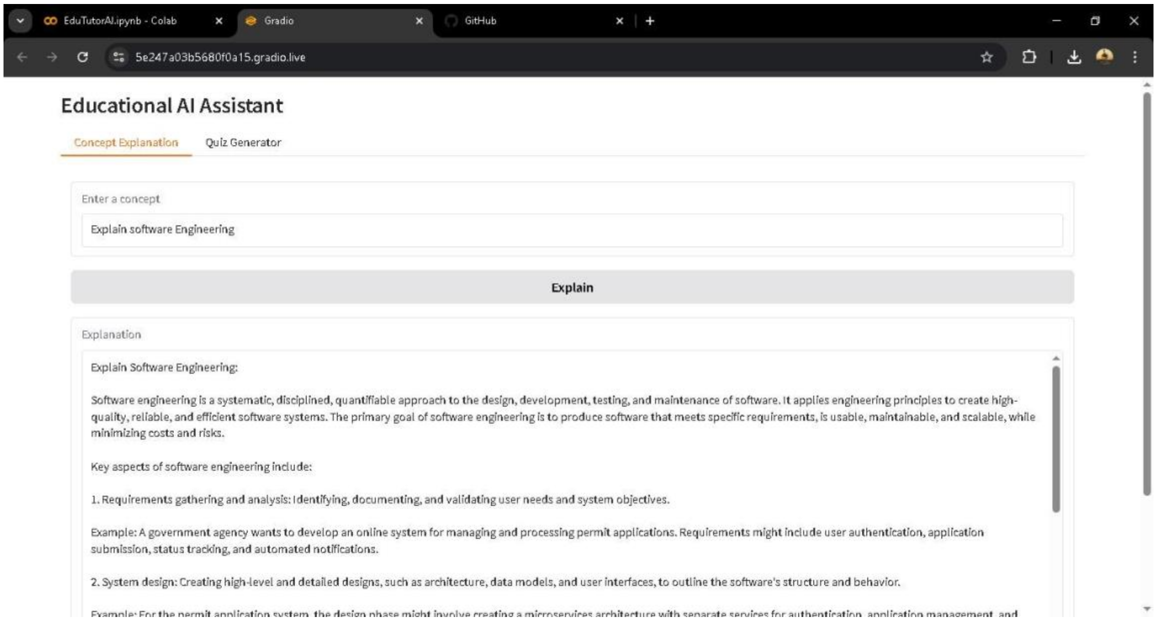
15. Program Code

```
# -*- coding: utf-8 -*-

"""EduTutorAI.ipynb


Automatically generated by Colab.


Original file is located at
```

```
https://colab.research.google.com/drive/1m7sTXRhXNPtn37t15KBWsvusu
5 70ixB9
"""


!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_toke
            n_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)
```

OUTPUT :



Enter a concept

Explain software Engineering

Explanation

Explain Software Engineering:

Software engineering is a systematic, disciplined, quantifiable approach to the design, development, testing, and maintenance of software. It applies engineering principles to create high-quality, reliable, and efficient software systems. The primary goal of software engineering is to produce software that meets specific requirements, is usable, maintainable, and scalable, while minimizing costs and risks.

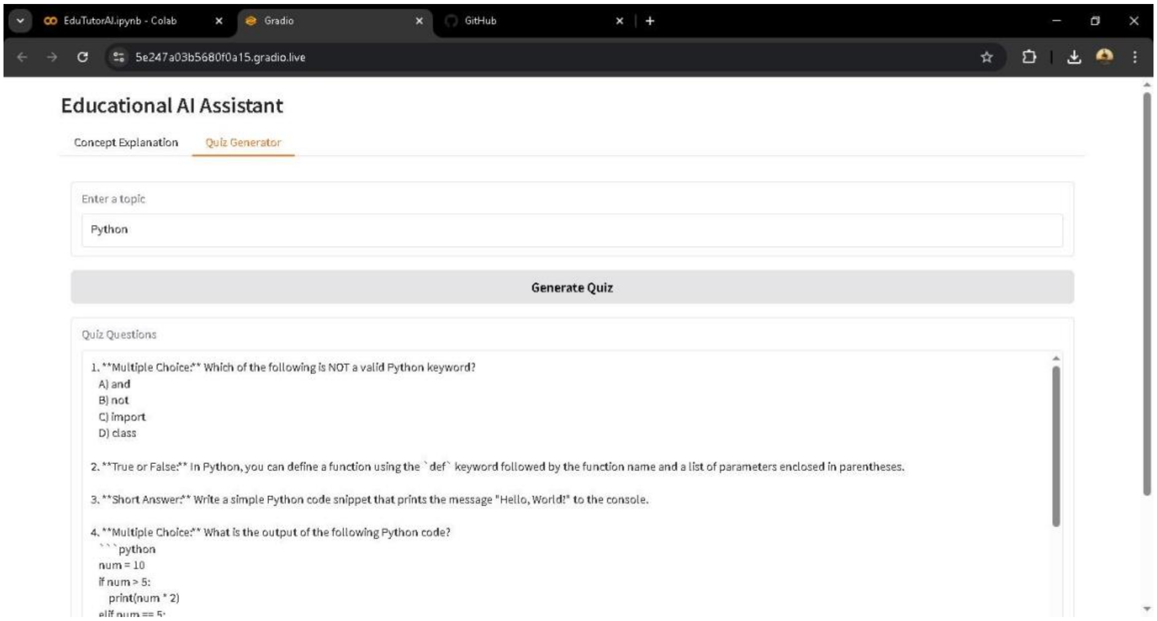
Key aspects of software engineering include:

1. Requirements gathering and analysis: Identifying, documenting, and validating user needs and system objectives.

Example: A government agency wants to develop an online system for managing and processing permit applications. Requirements might include user authentication, application submission, status tracking, and automated notifications.

2. System design: Creating high-level and detailed designs, such as architecture, data models, and user interfaces, to outline the software's structure and behavior.

Example: For the permit application system, the design phase might involve creating a microservices architecture with separate services for authentication, application management, and



Educational AI Assistant

Enter a topic

Python

Quiz Questions

~~~
1. **Multiple Choice:** Which of the following is NOT a valid Python keyword?
  A) and
  B) not
  C) import
  D) class

2. **True or False:** In Python, you can define a function using the `def` keyword followed by the function name and a list of parameters enclosed in parentheses.

3. **Short Answer:** Write a simple Python code snippet that prints the message "Hello, World!" to the console.

4. **Multiple Choice:** What is the output of the following Python code?
  ```python
  num = 10
  if num > 5:
     print(num * 2)
  elif num == 5:
~~~