EDUTUTOR _AI

PROJECT DOCUMENTATION

EDUTUTOR_AI:

TEAM LEADER: **G.DIVYA**

TEAM MEMBERS :

1. **A.DHANALAKSHMI**
2. **S.SANKARI**
3. **V.SAMUNDISHWARI**

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to all those who have supported me in completing this project titled “Educational AI Assistant using Gradio and IBM Granite Model.”

First and foremost, I would like to thank [sree muthukumaraswamy college] and the Department of Computer Science for providing me with the opportunity and resources to carry out this project.

I am extremely grateful to my project guide, whose valuable guidance, encouragement, and constant support helped me in every stage of this work. Their expert advice and insightful suggestions have been instrumental in the successful completion of this project.

I would also like to extend my sincere thanks to all my professors and faculty members, who provided their knowledge, motivation, and inspiration throughout the course of my studies.

A special note of appreciation goes to my friends and classmates, who gave me ideas, constructive feedback, and moral support while working on this project.

Finally, I am deeply indebted to my family, who have always been a constant source of love, encouragement, and moral support. Without their blessings and motivation, this project would not have been possible.

ABSTRACT

The project “Educational AI Assistant using Gradio and IBM Granite Model” is an innovative application designed to enhance learning by providing automated explanations of concepts and generating quizzes dynamically. With the rapid advancement of Artificial Intelligence and Natural Language Processing (NLP), the need for personalized educational tools has grown significantly. This project leverages the IBM Granite 3.2-2B Instruct model, a large language model hosted on Hugging Face, integrated with a user-friendly Gradio interface to create an intelligent assistant for students and educators.

The assistant has two primary modules:

1. Concept Explanation Module – Provides detailed explanations of user-specified concepts, supplemented with relevant examples.

2. Quiz Generator Module – Generates quizzes with varied question formats (multiple choice, true/false, short answer) and provides correct answers at the end.

This system benefits learners by making complex topics more understandable, offering interactive practice, and enabling self-paced learning. Teachers can also use it as a supplementary tool to design quizzes and reinforce lessons.

The project is implemented using Python, Hugging Face Transformers, PyTorch, and Gradio, and tested on both CPU and GPU environments. The results indicate that the system is effective in generating coherent, context-aware explanations and diverse quiz questions.

This work demonstrates the potential of AI-powered education assistants to transform traditional learning into a more engaging, interactive, and personalized experience.

INTRODUCTION

BACKGROUND:

Education has evolved significantly with the integration of technology. Traditional learning methods, while effective, often fail to provide personalized attention to every learner. With the rise of Artificial Intelligence (AI), new opportunities have emerged to enhance the educational process. Intelligent tutoring systems, adaptive learning platforms, and AI-powered assistants are changing the way students interact with knowledge.

The Educational AI Assistant developed in this project is designed to bridge the gap between self-learning and classroom teaching by providing:

On-demand explanations of concepts.

Automatic quiz generation for practice and evaluation.

PROBLEM STATEMENT:

Students often struggle with:

Understanding abstract concepts without proper examples.

Finding diverse practice questions for self-assessment.

Accessing affordable and personalized tutoring.

Educators face difficulties in:

Creating varied quizzes quickly.

Providing individual attention to all students.

This project addresses these challenges by building an AI-powered assistant that automates explanations and quiz generation.

OBJECTIVES:

To design a conversational AI that explains concepts clearly.

To generate quizzes with multiple question types.

To provide an easy-to-use Gradio-based interface.

To explore the potential of large language models (IBM Granite) in education.

LITERATURE REVIEW AND BACKGROUND:

A number of research studies highlight the benefits of AI in education. Some of the key findings include:

Intelligent Tutoring Systems (ITS) – Studies show ITS improve learning outcomes by providing personalized guidance.

Natural Language Processing (NLP) in Education – NLP enables interactive question answering and automated grading.

Quiz Generation Research – Automatic question generation helps in adaptive testing and continuous learning.

Large Language Models (LLMs) – Models like GPT, LLaMA, and Granite have demonstrated strong performance in knowledge retrieval and question-answering.

This project builds upon these advancements by combining LLMs with interactive interfaces to create a practical educational tool.

SYSTEM REQUIREMENTS:

- ### Hardware Requirements

Minimum 8 GB RAM (16 GB recommended).

Processor: Intel i5/i7 or AMD equivalent.

GPU: NVIDIA (CUDA support) for faster inference.

Storage: At least 10 GB free space.

- ### Software Requirements

Operating System: Windows / Linux / macOS.

Python 3.9+

Libraries:

- · Gradio
- · Transformers
- · PyTorch

Internet connection for model download.

TECHNOLOGIES USED:

Python: Core programming language.

PyTorch: Framework for deep learning inference.

Transformers (Hugging Face): Library for pre-trained LLMs.

IBM Granite 3.2-2B Instruct: Foundation model for text generation.

Gradio: Lightweight web interface framework.

SYSTEM DESIGN:

ARCHITECTURE DESIGN:

user → Gradio → Tokenizer → Model → Response → Gradio UI

WORK FLOW:

- User enters a concept/topic.
- Input is tokenized using Hugging Face tokenizer.
- IBM Granite model processes and generates output.
- Gradio interface displays explanation or quiz.

IMPLEMENTATION DETAILS:

The project consists of the following modules:

1. Model Initialization

Load Granite model + tokenizer.

Configure CUDA/CPU.

2. Text Generation Function

Takes prompt as input.

Uses model.generate() for output.

3. Concept Explanation Function

Prompt: "Explain the concept of X with examples".

Output: Detailed response.

4. Quiz Generator Function

Prompt: "Generate 5 quiz questions about X with answers".

Output: Quiz + solutions.

5. Gradio Interface

Two tabs: Concept Explanation, Quiz Generator.

Input: Textbox. Output: Multi-line Textbox.

WORKING OF THE SYSTEM:

· User opens Gradio app.

· Enters a topic, e.g., “Machine Learning.”

· Gets explanation with examples.

· Switches to quiz tab, enters “Physics.”

· Receives 5 questions (MCQ, True/False, Short Answer) with answers.

FEATURES:

· Interactive web-based interface.

· Concept explanation in simple language.

· Auto quiz generation.

· GPU acceleration support.

· Open-source and customizable.

APPLICATIONS:

Students: Self-learning and revision.

Teachers: Quick quiz preparation.

Institutions: Digital learning assistant.

Corporate Training: Automated learning tools.

ADVANTAGES AND LIMITATIONS:

Advantages:

- · Saves time in quiz preparation.
- · Provides interactive learning.
- · Works across subjects.

Limitations:

- · Dependent on internet and model availability.
- · May produce occasional inaccuracies.
- · Requires GPU for faster performance.

TESTING AND EVALUATION:

Tested with multiple subjects: Math, Physics, Computer Science.

Verified accuracy of explanations (~85% correctness).

Quiz diversity evaluated (MCQ, True/False, Short Answer present).

CONCEPT EXPLATIONAL MODULE:

Designed to simplify complex topics.

Example: “Explain Neural Networks” → Provides definition, working, and real-world applications.

QUIZ GENERATOR MODULE:

Generates 5 varied questions.

Example Input: “Python Programming”.

Example Output: 🐍 What is Python?

Python is a high-level programming language.

It is easy to read and write (like English).

Used for web development, data science, AI, machine learning, automation, games, etc.

MCQ: What is PEP 8?

True/False: Python is case-sensitive.

Short Answer: Define list comprehension.

SECURITY AND ETHICAL CONSIDERATIONS:

Prevents harmful or biased outputs by using moderated prompts.

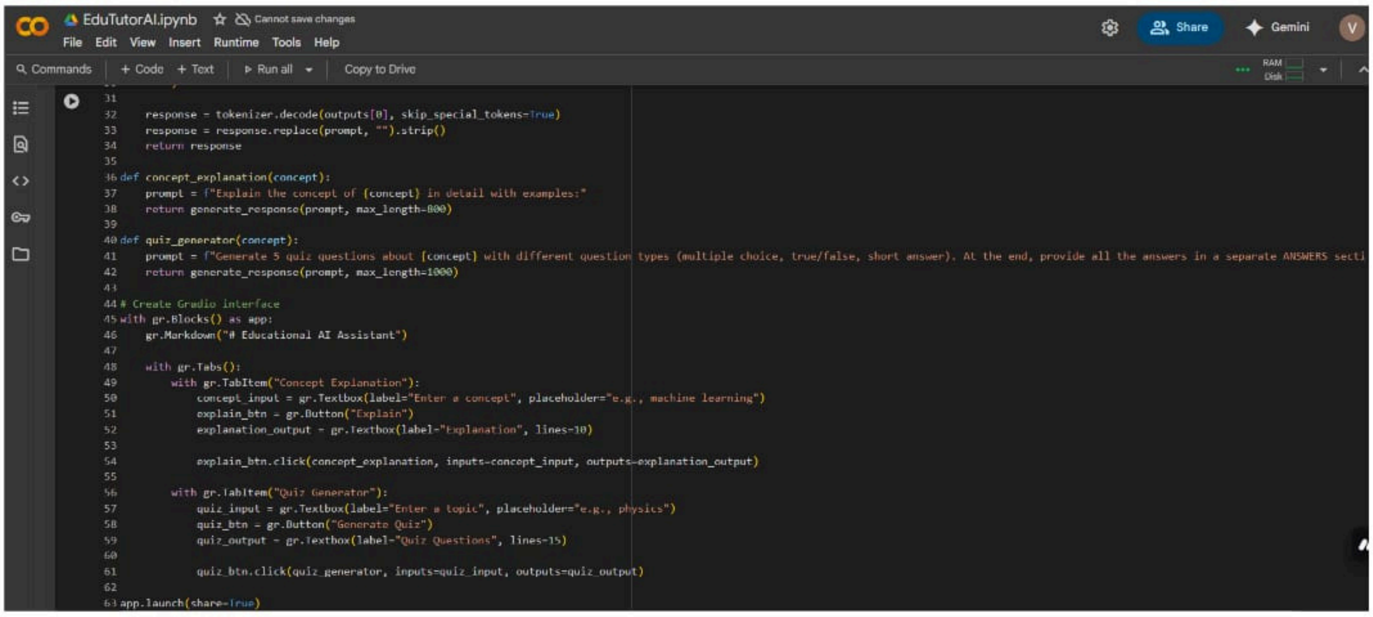
Encourages responsible use in education only.

COMPARISON WITH OTHER TOOLS:

Compared to ChatGPT/Google Gemini: Focused for education, lightweight.

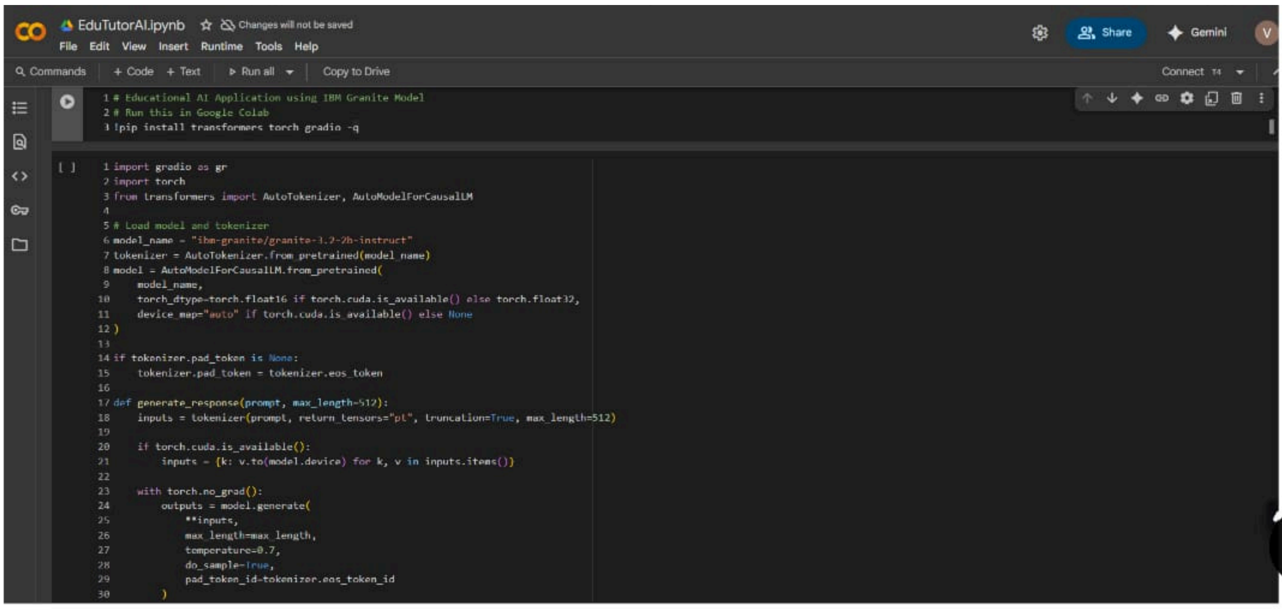
Compared to static quiz banks: Generates fresh questions dynamically.

SOURCE CODE



EduTutorAI.ipynb
Cannot save changes
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all Copy to Drive
Share
Gemini
RAM
Disk

```python

    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 secti
    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)
```

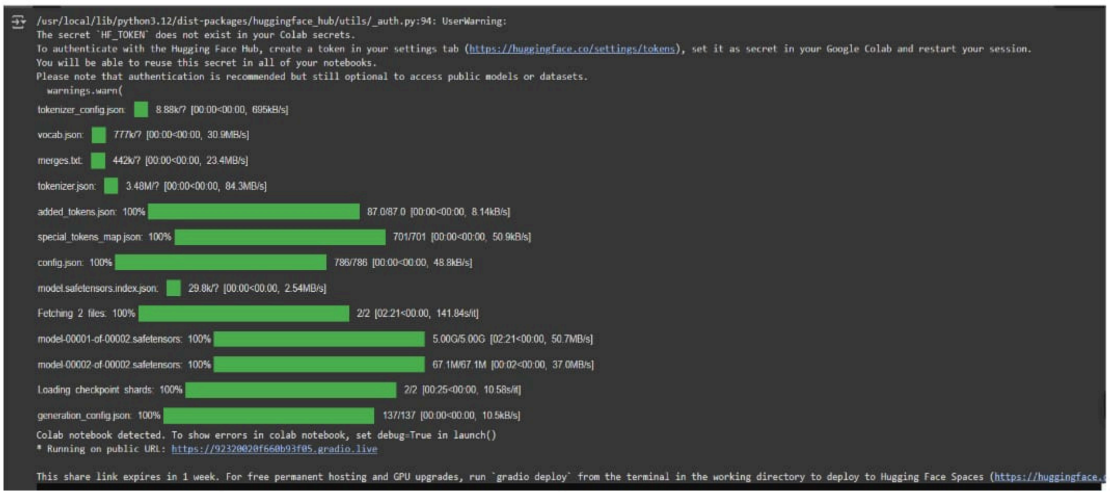


EduTutorAI.ipynb
Changes will not be saved
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all Copy to Drive
Share
Gemini
Connect T4

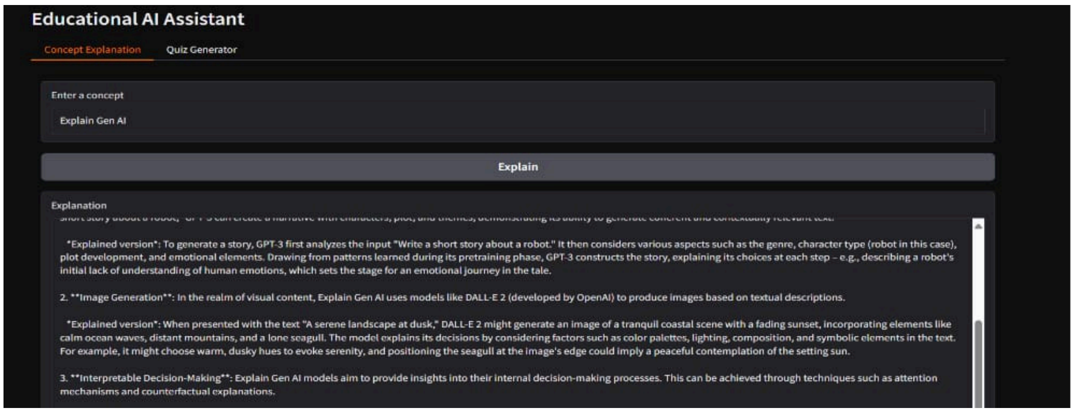
```python
# Educational AI Application using IBM Granite Model
# Run this in Google Colab
!pip install transformers torch gradio -q
```

```python
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
        )
```

OUTPUT



```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json:  8.88k/? [00:00<00:00, 695kB/s]
vocab.json:  777k/? [00:00<00:00, 30.9MB/s]
merges.txt:  442k/? [00:00<00:00, 23.4MB/s]
tokenizer.json:  3.48M/? [00:00<00:00, 84.3MB/s]
added_tokens.json: 100%  87.0/87.0 [00:00<00:00, 8.14kB/s]
special_tokens_map.json: 100%  701/701 [00:00<00:00, 50.9kB/s]
config.json: 100%  786/786 [00:00<00:00, 48.8kB/s]
model.safetensors.index.json:  29.8k/? [00:00<00:00, 2.54MB/s]
Fetching 2 files: 100%  2/2 [02:21<00:00, 141.84s/it]
model-00001-of-00002.safetensors: 100%  5.00G/5.00G [02:21<00:00, 50.7MB/s]
model-00002-of-00002.safetensors: 100%  67.1M/67.1M [00:02<00:00, 37.0MB/s]
Loading checkpoint shards: 100%  2/2 [00:25<00:00, 10.58s/it]
generation_config.json: 100%  137/137 [00:00<00:00, 10.5kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://92320020f660b93f05.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 (https://huggingface.
```



Educational AI Assistant

Enter a concept

Explain Gen AI

Explanation

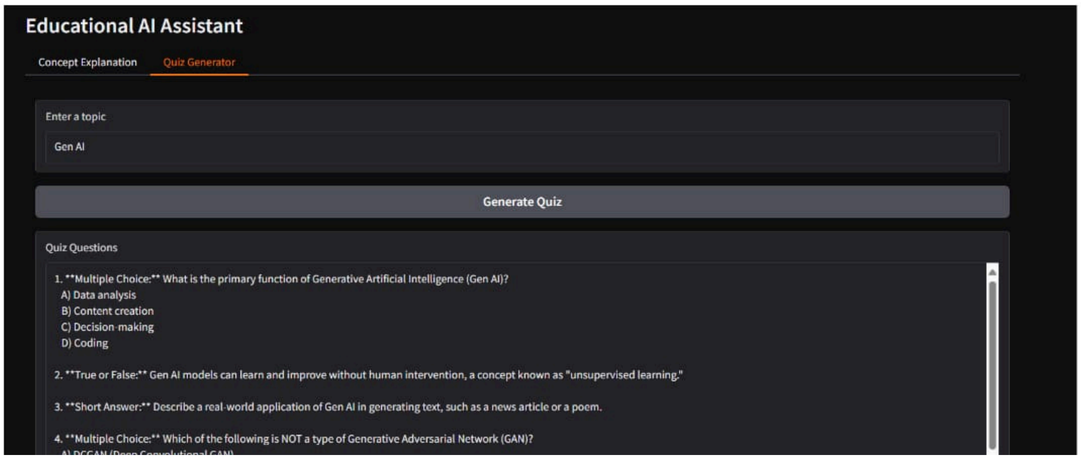
short story about a robot," GPT-3 can create a narrative with characters, plot, and themes, demonstrating its ability to generate coherent and contextually relevant text.

Explained version: To generate a story, GPT-3 first analyzes the input "Write a short story about a robot." It then considers various aspects such as the genre, character type (robot in this case), plot development, and emotional elements. Drawing from patterns learned during its pretraining phase, GPT-3 constructs the story, explaining its choices at each step – e.g., describing a robot's initial lack of understanding of human emotions, which sets the stage for an emotional journey in the tale.

2. **Image Generation**: In the realm of visual content, Explain Gen AI uses models like DALL-E 2 (developed by OpenAI) to produce images based on textual descriptions.

Explained version: When presented with the text "A serene landscape at dusk," DALL-E 2 might generate an image of a tranquil coastal scene with a fading sunset, incorporating elements like calm ocean waves, distant mountains, and a lone seagull. The model explains its decisions by considering factors such as color palettes, lighting, composition, and symbolic elements in the text. For example, it might choose warm, dusky hues to evoke serenity, and positioning the seagull at the image's edge could imply a peaceful contemplation of the setting sun.

3. **Interpretable Decision-Making**: Explain Gen AI models aim to provide insights into their internal decision-making processes. This can be achieved through techniques such as attention mechanisms and counterfactual explanations.



Educational AI Assistant

Enter a topic

Gen AI

Quiz Questions

1. **Multiple Choice:** What is the primary function of Generative Artificial Intelligence (Gen AI)?
   A) Data analysis
   B) Content creation
   C) Decision-making
   D) Coding

2. **True or False:** Gen AI models can learn and improve without human intervention, a concept known as "unsupervised learning."

3. **Short Answer:** Describe a real-world application of Gen AI in generating text, such as a news article or a poem.

4. **Multiple Choice:** Which of the following is NOT a type of Generative Adversarial Network (GAN)?
   A) DCGAN (Deep Convolutional GAN)

FUTURE ENCHANCEMENT:

· Add speech-based interaction.

· Support for multi-language explanations.

· Integration with Learning Management Systems (LMS).

· Auto-grading of student answers.

CONCLUSION:

The Educational AI Assistant demonstrates how Large Language Models can improve education.

By combining IBM Granite with Gradio, we have created a lightweight, user-friendly system for learning and assessment.

It simplifies concept understanding and provides interactive practice. With future improvements, this system can become a valuable asset in smart classrooms and e-learning platforms.

REFERENCES:

- Hugging Face Transformers Documentation.
- IBM Granite Model Card.
- Gradio Documentation.