EDUCATIONAL AI: PERSONALIZED LEARNING GENERATIVE AI WITH IBM

1.INTRODUCTION:

PROJECT TITLE: EDUCATIONAL AI: PERSONALIZED LEARNING GENERATIVE AI WITH IBM

TEAM LEADER: RAGHUL S

TEAM MEMBER:

* KIRAN KUMAR M
* SHANKAR C D
* DINESHKUMAR R

1.PROJECT OVWERVIEW:

EduTutor AI is an intelligent, AI-powered educational assistant designed to enhance personalized learning for students and provide efficient teaching support for educators**.** The system leverages Natural Language Processing (NLP), Machine Learning (ML), and Interactive User Interfaces to deliver customized learning experiences, instant query resolution, and adaptive content recommendations.

The platform acts as a virtual tutor, capable of understanding student queries in natural language, explaining academic concepts in detail, generating practice questions, and offering step-by-step solutions. It also supports teachers by providing automated assessment tools, study material generation, and student progress tracking.

By combining AI-driven tutoring, real-time assistance, and data analytics, EduTutor AI bridges the gap between traditional classroom teaching and modern digital education.

Key Features

Personalized Learning – Adapts content based on student performance and preferences.

AI-Powered Q&A – Answers academic queries instantly with accurate explanations

Content Generation – Creates study notes, quizzes, and practice exercises automatically.

Progress Tracking – Monitors student performance and provides insights for improvement.

Multi-Platform Support – Accessible via web and mobile for flexible learning.

2. Architecture

1. User Interface Layer

Web/Mobile App (React, Flutter, HTML/CSS/JS)

Provides students and teachers with a friendly interface to:

Ask questions

Access notes/quizzes

View progress reports

Handles input/output in text/voice

2. Application Layer

API Gateway / Backend Server (Flask, Django, or FastAPI)

Manages requests between users and the AI engine.

Functions include:

User authentication & role management (student/teacher)

Query routing to AI modelsHandling content generation requests

Storing results in the database

3. AI Engine Layer

Natural Language Processing (NLP) Module

Processes student queries (tokenization, intent recognition).

Knowledge Base & Retrieval System

Uses textbooks, notes, and academic content stored in DB/Cloud.

Machine Learning/LLM Models

Pre-trained transformer (GPT, BERT, Granite, or fine-tuned models).

Generates explanations, answers, and summaries.

Content Generator

Auto-creates quizzes, study notes, and exercises.

4. Data Management Layer

Database (SQL/NoSQL)

Stores user profiles, progress data, and learning materials.

Content Repository

Stores study notes, question banks, and generated materials.

Analytics Engine

Tracks student performance and generates reports.

5. Cloud & Deployment Layer

Hosting Services (AWS, Azure, or GCP)

Ensures scalability, availability, and security.

Containerization (Docker, Kubernetes)

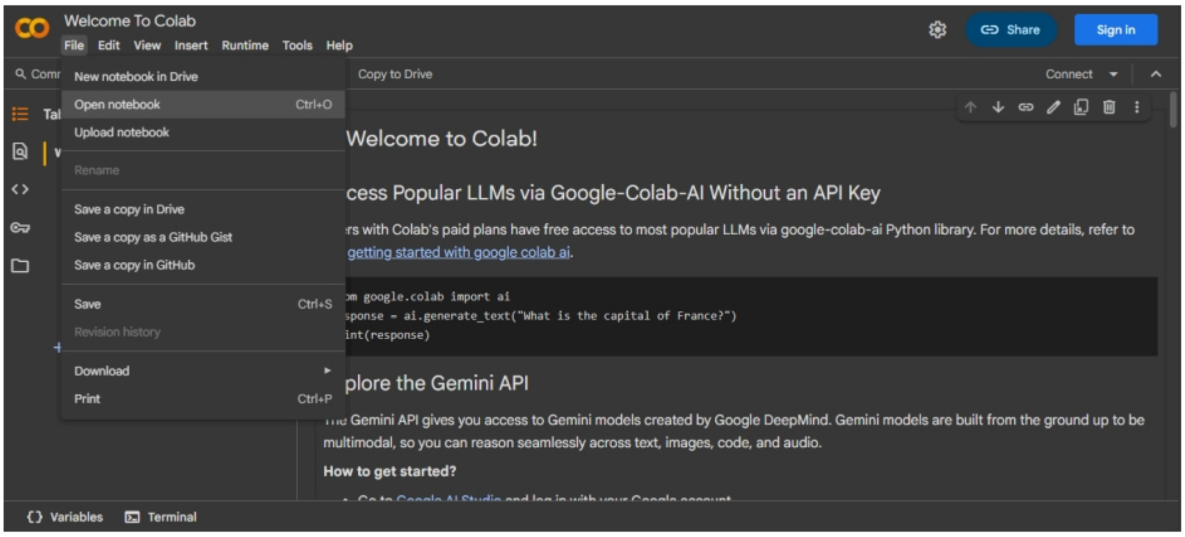
For smooth deployment and scaling.

Monitoring & Logging.

2. TOOLS AND TECHNOLOGIES USED

Tools and Libraries Used:

- **Google Colab** – For cloud-based Python execution.



Welcome to Colab!

cess Popular LLMs via Google-Colab-AI Without an API Key

rs with Colab's paid plans have free access to most popular LLMs via google-colab-ai Python library. For more details, refer to getting started with google colab ai.

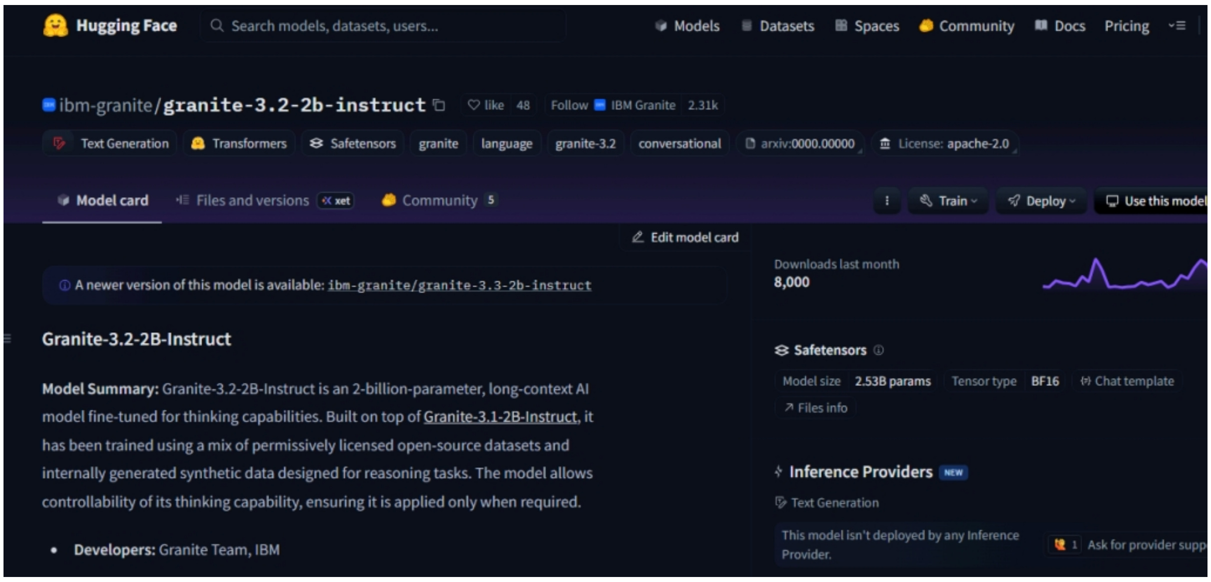
```
om google.colab import ai
sponse = ai.generate_text("What is the capital of France?")
int(response)
```

plore the Gemini API

Gemini API gives you access to Gemini models created by Google DeepMind. Gemini models are built from the ground up to be multimodal, so you can reason seamlessly across text, images, code, and audio.

How to get started?

- **Hugging Face Transformers** – For loading IBM Granite model.



A newer version of this model is available: ibm-granite/granite-3.3-2b-instruct

Granite-3.2-2B-Instruct

Model Summary: Granite-3.2-2B-Instruct is an 2-billion-parameter, long-context AI model fine-tuned for thinking capabilities. Built on top of Granite-3.1-2B-Instruct, it has been trained using a mix of permissively licensed open-source datasets and internally generated synthetic data designed for reasoning tasks. The model allows controllability of its thinking capability, ensuring it is applied only when required.

- **Developers:** Granite Team, IBM

This model isn't deployed by any Inference Provider.

- **Gradio** – To build a web-based user interface.
- **Torch** – For model execution with CUDA/GPU support.

3. IMPLEMENTATION

- **Model Setup:**

model_name = "ibm-granite/granite-3.2-2b-instruct"

tokenizer = AutoTokenizer.from_pretrained(model_name)

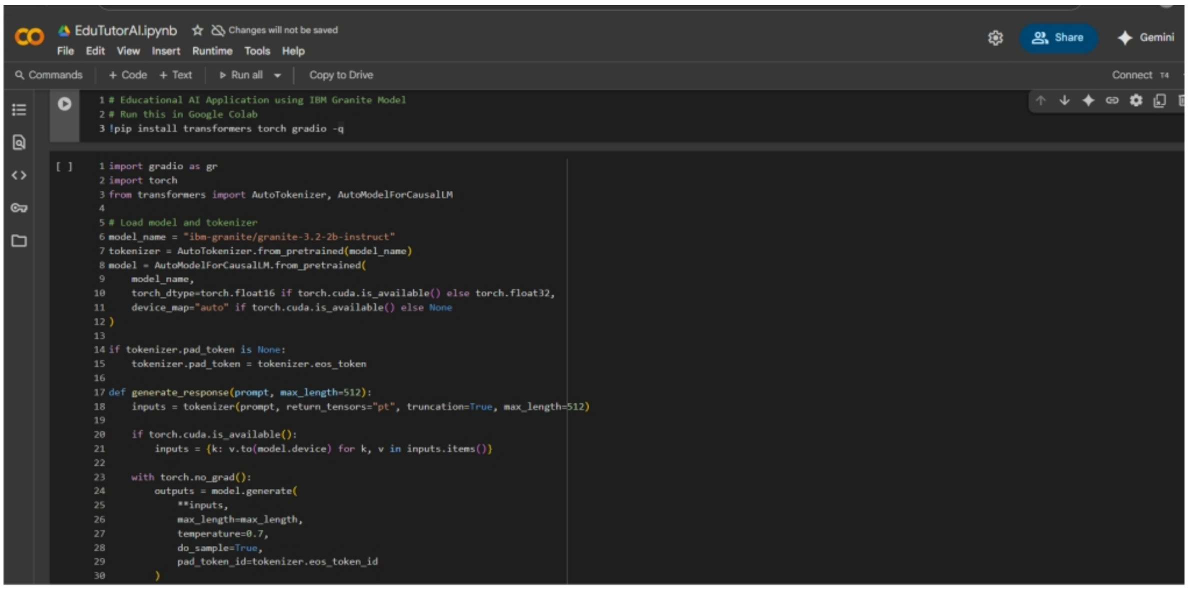
model = AutoModelForCausalLM.from_pretrained(model_name, ...)

- **Function to Generate Responses:**

def generate_response(prompt, max_length=512):

...

```
    outputs = model.generate(...)

    return decoded_output
```


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

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

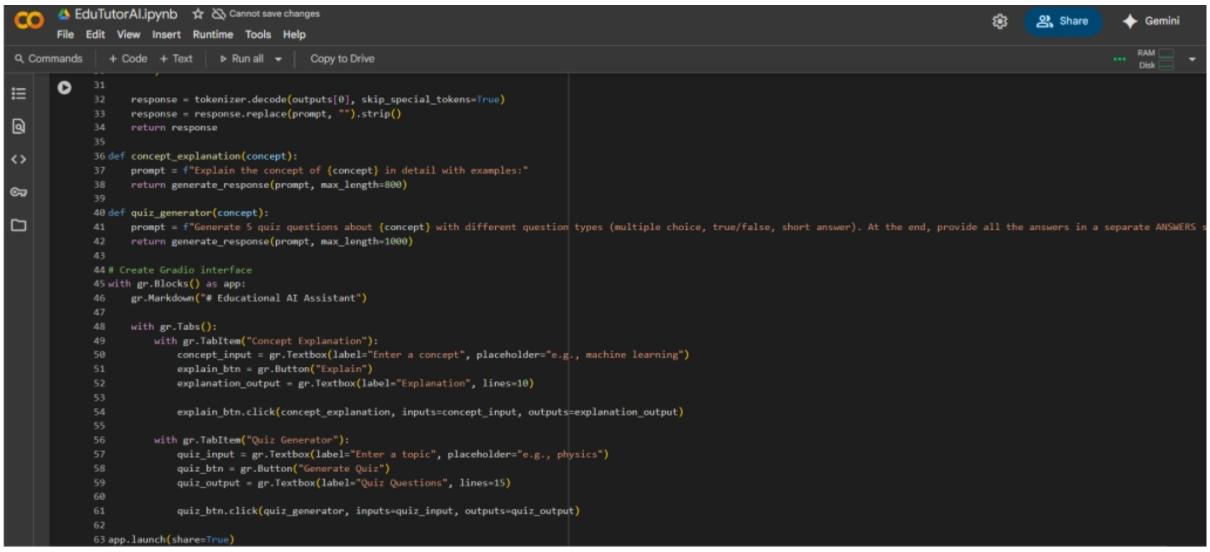
```
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# 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
        )
```

- **Concept Explanation:**

```
def concept_explanation(concept):

    prompt = f"Explain the concept of {concept} in detail with
examples:"

    return generate_response(prompt, max_length=800)
```

- **Quiz Generation:**

```
def quiz_generator(concept):

    prompt = f"Generate 5 quiz questions about {concept}..."

    return generate_response(prompt, max_length=1000)
```


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 s
    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)
```

4. USER INTERFACE AND FLOW

Gradio Interface Setup:

- Two tabs:
  - *Concept Explanation*
  - *Quiz Generator*

```
with gr.Tabs():
    with gr.TabItem("Concept Explanation"):
        ...
    with gr.TabItem("Quiz Generator"):
        ...
```

Execution Environment:

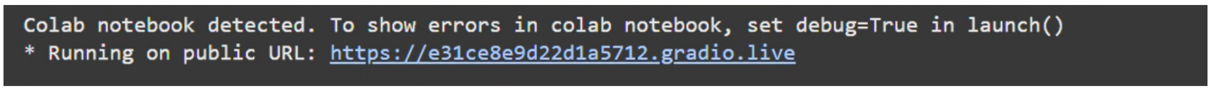
The application was executed using a **T4 GPU** runtime in **Google Colab**, ensuring fast model inference.

Authen tication Warning

While the app works without a Hugging Face token, using a token (HF_TOKEN) can improve reliability by accessing private models or increasing rate limits.

5. OUTPUT DEMONSTRATIONS

Click on the URl to open the Gradio Application click on the link. You can View the Application is running in the other tab



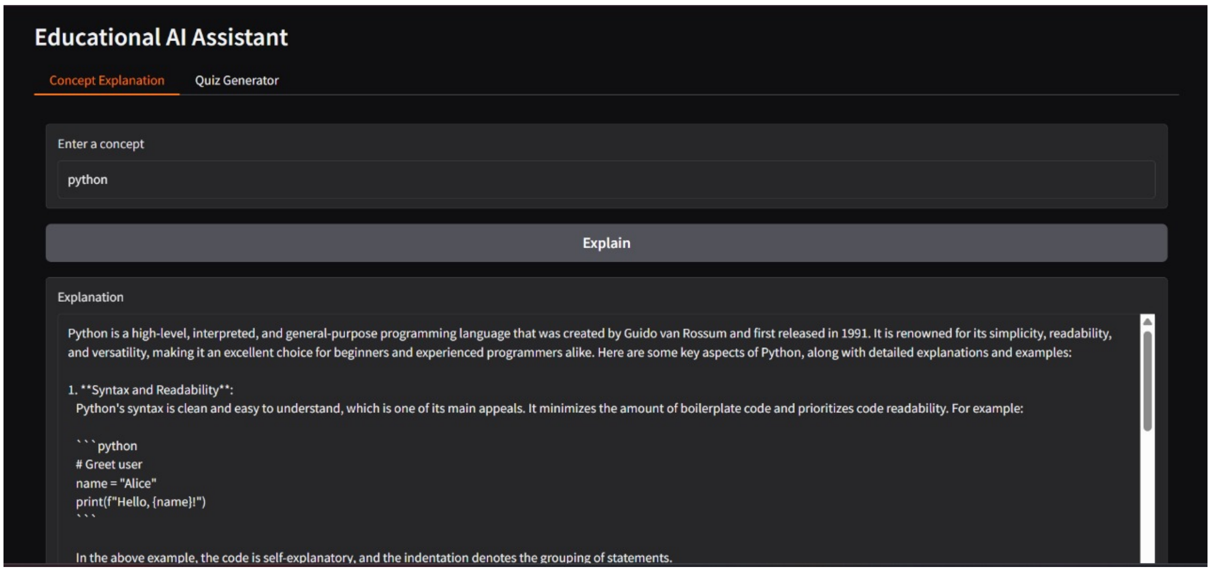
```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://e31ce8e9d22d1a5712.gradio.live
```

1. Concept Explanation Output:

User Input: python

Generated Output:

Explanation on Python’s syntax, code readability, history, and usage, with code samples.



python

Python is a high-level, interpreted, and general-purpose programming language that was created by Guido van Rossum and first released in 1991. It is renowned for its simplicity, readability, and versatility, making it an excellent choice for beginners and experienced programmers alike. Here are some key aspects of Python, along with detailed explanations and examples:

1. **Syntax and Readability**:
  Python's syntax is clean and easy to understand, which is one of its main appeals. It minimizes the amount of boilerplate code and prioritizes code readability. For example:

~~~
  ```python
  # Greet user
  name = "Alice"
  print(f"Hello, {name}!")
  ```
~~~

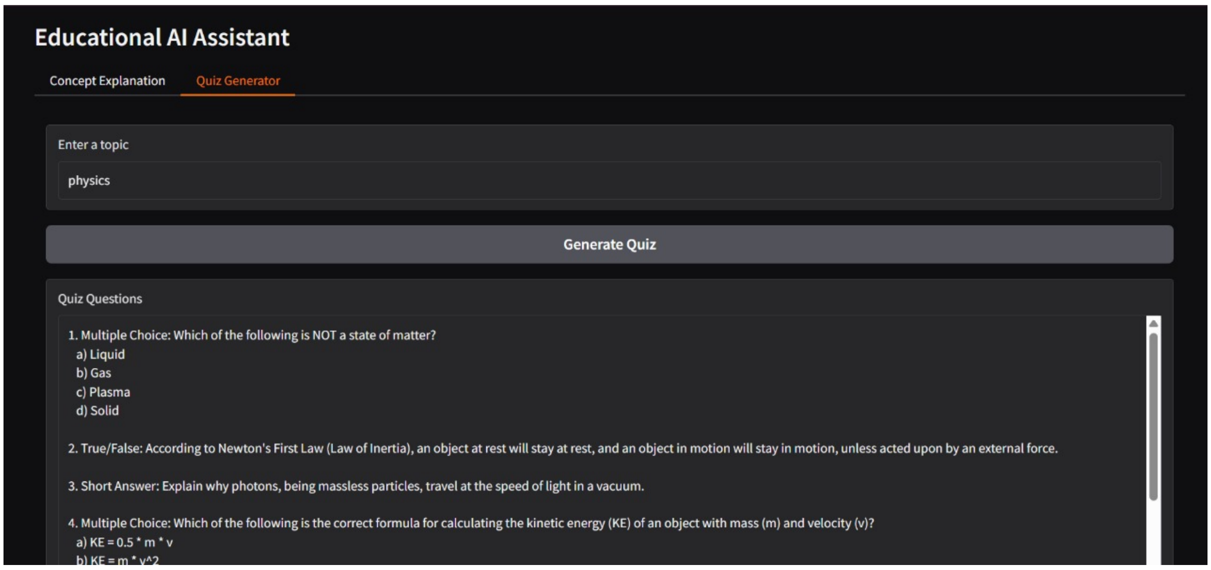
  In the above example, the code is self-explanatory, and the indentation denotes the grouping of statements.

2. Quiz Generator Output:

User Input: physics

Generated Output:

A list of 5 quiz questions (MCQs, True/False, Short Answer) and an **ANSWERS** section.



physics

1. Multiple Choice: Which of the following is NOT a state of matter?
  a) Liquid
  b) Gas
  c) Plasma
  d) Solid

2. True/False: According to Newton's First Law (Law of Inertia), an object at rest will stay at rest, and an object in motion will stay in motion, unless acted upon by an external force.

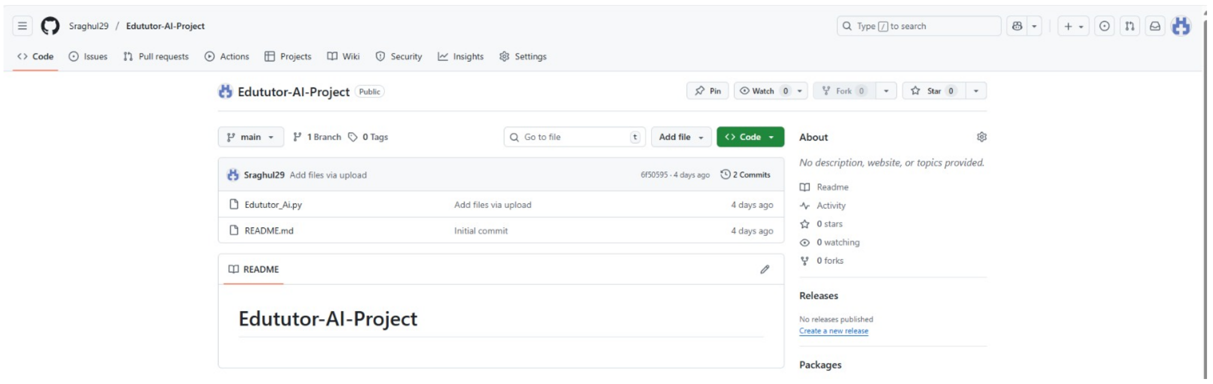
3. Short Answer: Explain why photons, being massless particles, travel at the speed of light in a vacuum.

4. Multiple Choice: Which of the following is the correct formula for calculating the kinetic energy (KE) of an object with mass (m) and velocity (v)?
  a) KE = 0.5 * m * v
  b) KE = m * v^2

6. GITHUB DEPLOYMENT

GitHub Repository

The complete source code, including model setup, Gradio interface, and executable scripts, has been uploaded to a public GitHub repository for easy access, collaboration, and version control.



No description, website, or topics provided.

Sraghul29 Add files via upload 6f50595 · 4 days ago 2 Commits

Edututor_Ai.py	Add files via upload	4 days ago
README.md	Initial commit	4 days ago

Edututor-AI-Project

No releases published
Create a new release

Repository Link:

https://github.com/Sraghul29/Edututor-AI-Project.git

Contents of the Repository:

File Name	Description
README.md	Provides project overview and instructions.
edututorai.py	Python script containing full Gradio app code.

This deployment ensures transparency, allows others to contribute, and enables continuous improvement and tracking of project evolution.

7. CONCLUSION & FUTURE SCOPE

Conclusion

This AI-based Educational Assistant demonstrates how powerful LLMs like IBM Granite can enhance the educational experience. The system is capable of dynamic text generation, adaptive learning support, and content creation, all through a user-friendly interface.

8. FUTURE ENHANCEMENTS:

- Add multilingual support.
- Integrate voice-based interaction using Speech-to-Text APIs.
- Save quiz results to a database for student progress tracking.
- Allow teachers to fine-tune question formats.
- Add model-switching (option to choose between open LLMs like Mistral, LLaMA, or GPT).

9. REFERENCES

1. IBM Granite Model: https://huggingface.co/ibm-granite/granite-3.2-2b-instruct
2. Hugging Face Transformers Library: https://huggingface.co/docs/transformers
3. Gradio Documentation: https://www.gradio.app/
4. PyTorch Documentation: https://pytorch.org/docs/stable/index.html
5. Google Colab: https://colab.research.google.com/