EDUTUTOR AI PROJECT DOCUMENTATION

1. INTRODUCTION

PROJECT TITLE: Edututor AI: Intelligent Personalized Learning Platform

TEAM MEMBER: R.Gayathri

TEAM MEMBERS: P.Rosemary

TEAM MEMBERS: J.Thanuja

TEAM MEMBERS: K.Gopika

2. PROJECT OVERVIEW

"Edututor AI" is an intelligent tutoring and learning assistance system that leverages artificial intelligence to provide personalized education experiences.Unlike traditional e-learning systems, Edututor AI adapts to each learner’s pace, style, and preferences, making education more accessible, engaging, and effective.

It supports both students (as learners) and teachers (as facilitators), offering tools for real-time feedback, automated grading, learning path recommendations, and interactive support.

Key Features

Automation:

Automates repetitive educational tasks such as grading, scheduling, and assessment generation, reducing the workload for educators.

Personalization:

Uses AI to tailor lessons, exercises, and study plans to match each student's abilities, progress, and learning style.

Data-Driven Insights:

Analyzes learning patterns, strengths, and weaknesses of students to provide actionable insights to teachers and institutions.

Natural Language Processing (NLP):

Powers AI-driven tutors and chatbots to enable natural conversations with students for answering questions, providing explanations, and assisting with study material.

3. ARCHITECTURE

The architecture of Edututor AI integrates AI-driven learning engines with a student-centric design to enhance accessibility, fairness, and adaptability in education.

Core Architectural Layers and Components:

1.Data Collection Layer

Sources: Student performance data, e-learning platforms, educational resources, user feedback.

Governance: Ensures compliance with FERPA/GDPR to protect student data privacy.

Processing: Stores and processes learning records in scalable cloud environments.

2. AI & Analytics Layer

Adaptive learning models: Recommend personalized content.

Predictive analytics: Forecast student performance and dropout risks.

NLP engines: Enable chatbot tutoring and question-answering systems.

3. Application Layer

Interfaces for students, teachers, and administrators.

Microservices-based structure for modularity (assessment service, recommendation engine, chatbot).

4. SET OF INSTRUCTIONS

To build and run Edututor AI, the following skills and tools are required:

1. Gradio framework knowledge: For building interactive AI interfaces.
2. IBM Granite / Hugging Face Models: For NLP-based tutoring.
3. Python programming proficiency: Core development language.

4. Version control with Git: For collaboration and project tracking.

5. Google Colab / GPU knowledge: For model training and testing.

5. RUNNING THE APPLICATION

After training and validating the Edututor AI models, the system can be deployed in:

Web applications (student dashboards, teacher portals).

Mobile platforms (lightweight tutoring apps).

Learning Management Systems (LMS) integration (e.g., Moodle, Canvas).

6. USER INTERFACE

The Edututor AI UI should be:

Student-friendly: Simple navigation, gamified elements, accessibility support.

Teacher-friendly: Dashboards for progress tracking, automated grading results, insights.

Mobile-first: Optimized for phones and tablets for accessibility in diverse environments.

7. TESTING

Testing in Edututor AI covers both software and AI-specific challenges:

Accuracy Testing: Verifying AI recommendations and predictions.

Fairness Testing: Ensuring unbiased treatment across different student groups.

Usability Testing: Checking ease of use for both students and teachers.

Performance Testing: Stress-testing under heavy usage (large classrooms).

8. KEY BENEFITS & APPLICATIONS

Personalized Learning: Tailored lessons and practice.

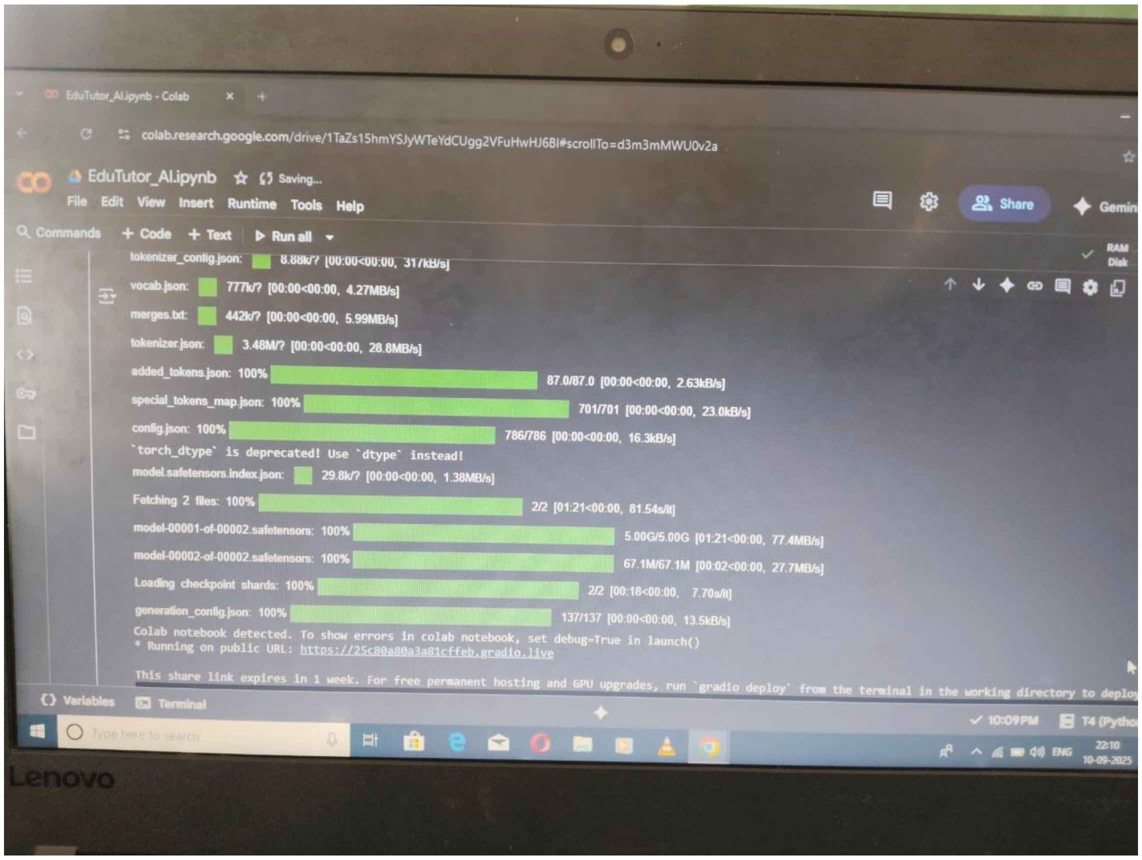
Automated Tutoring: AI chatbot answers queries instantly.

Efficient Grading: Automatic evaluation of quizzes and assignments.

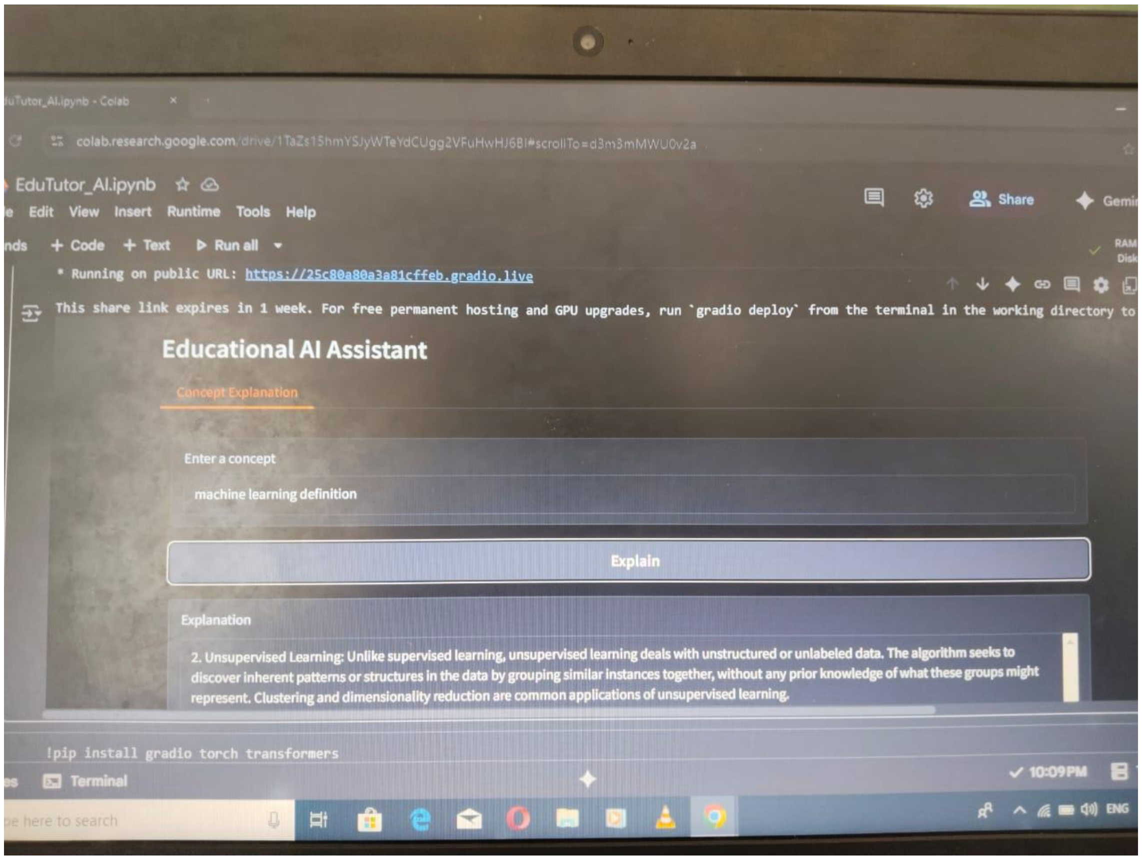
Data-Driven Teaching: Insights for teachers to focus on weak areas.

24/7 Accessibility: On-demand learning outside school hours.

9. SCREENSHOTS



```
tokenizer_config.json:  8.88k/? [00:00<00:00, 317kB/s]
vocab.json:  777k/? [00:00<00:00, 4.27MB/s]
merges.txt:  442k/? [00:00<00:00, 5.99MB/s]
tokenizer.json:  3.48M/? [00:00<00:00, 28.8MB/s]
added_tokens.json: 100%  87.0/87.0 [00:00<00:00, 2.63kB/s]
special_tokens_map.json: 100%  701/701 [00:00<00:00, 23.0kB/s]
config.json: 100%  786/786 [00:00<00:00, 16.3kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json:  29.8k/? [00:00<00:00, 1.38MB/s]
Fetching 2 files: 100%  2/2 [01:21<00:00, 81.54s/it]
model-00001-of-00002.safetensors: 100%  5.00G/5.00G [01:21<00:00, 77.4MB/s]
model-00002-of-00002.safetensors: 100%  67.1M/67.1M [00:02<00:00, 27.7MB/s]
Loading checkpoint shards: 100%  2/2 [00:18<00:00,  7.70s/it]
generation_config.json: 100%  137/137 [00:00<00:00, 13.5kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://25c80a80a3a81cffeb.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
```



```
* Running on public URL: https://25c80a80a3a81cffeb.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
```

Educational AI Assistant

Concept Explanation

Enter a concept

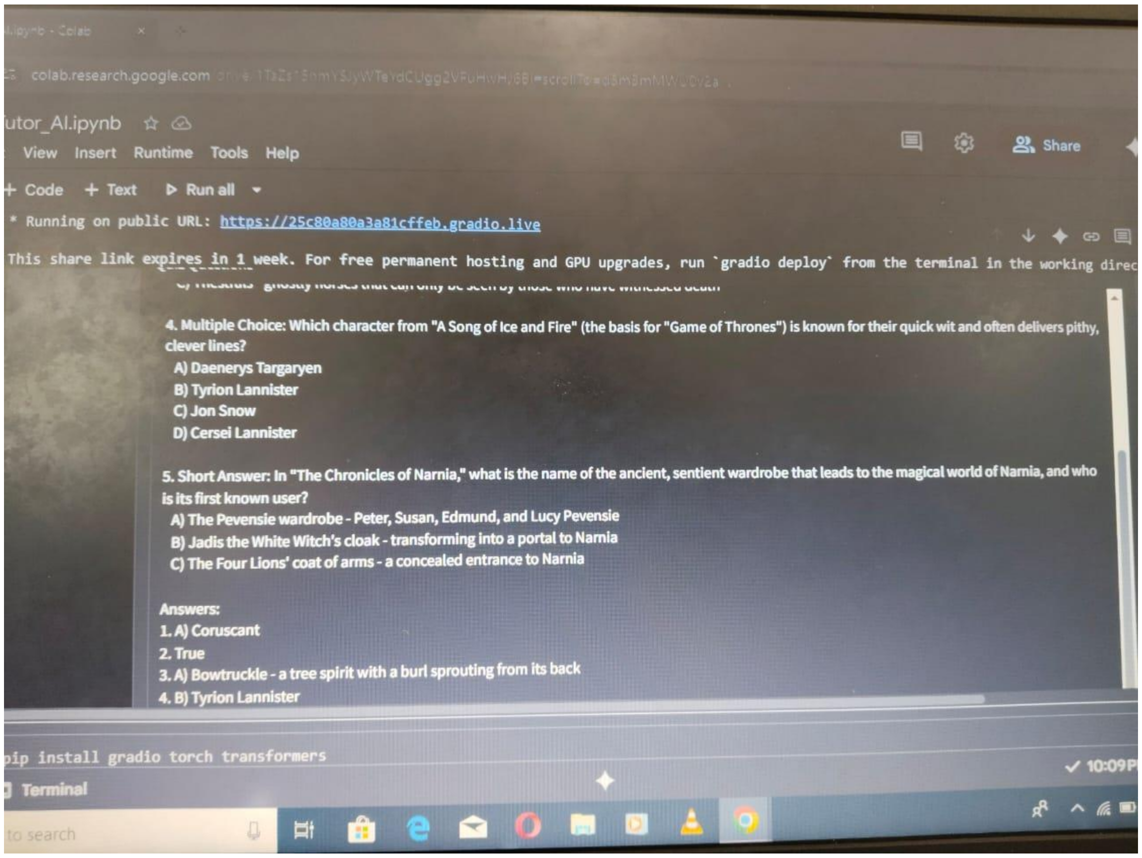
machine learning definition

Explain

Explanation

2. Unsupervised Learning: Unlike supervised learning, unsupervised learning deals with unstructured or unlabeled data. The algorithm seeks to discover inherent patterns or structures in the data by grouping similar instances together, without any prior knowledge of what these groups might represent. Clustering and dimensionality reduction are common applications of unsupervised learning.

```
!pip install gradio torch transformers
```



```
* Running on public URL: https://25c80a80a3a81cffeb.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 direc
```

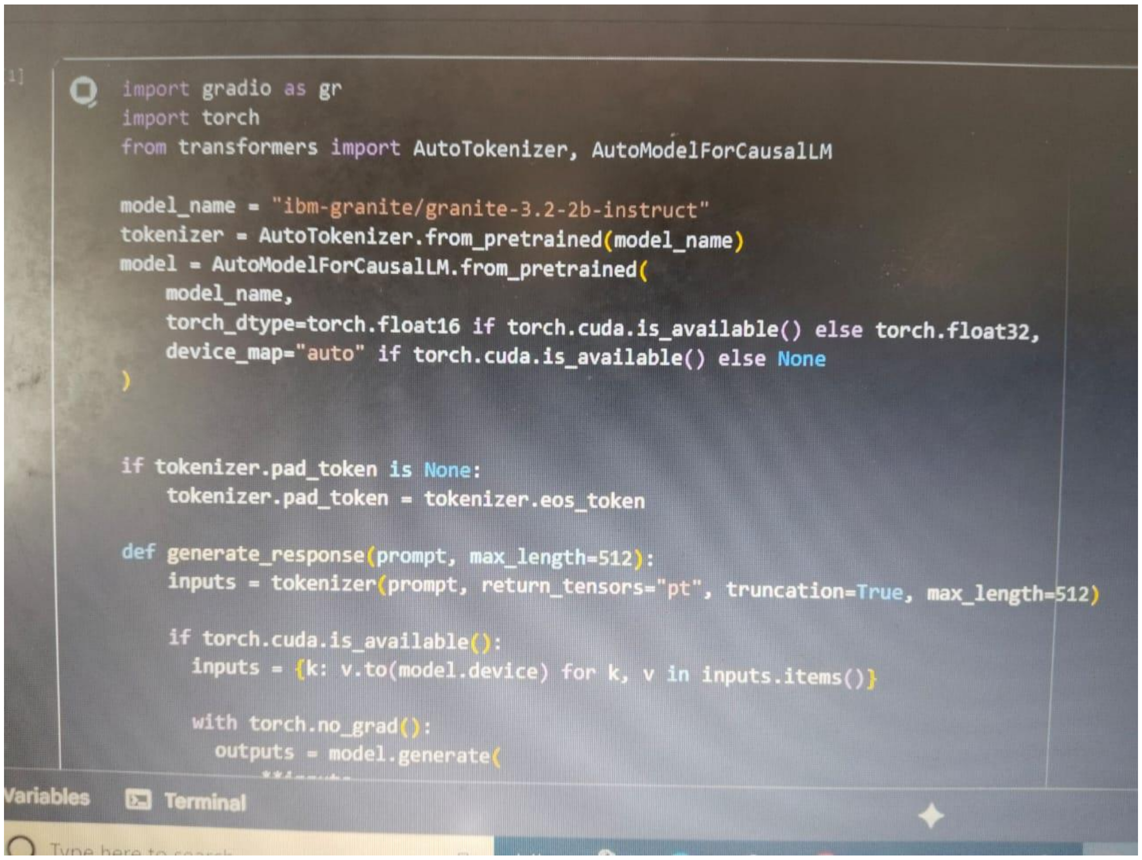
4. Multiple Choice: Which character from "A Song of Ice and Fire" (the basis for "Game of Thrones") is known for their quick wit and often delivers pithy, clever lines?
   A) Daenerys Targaryen
   B) Tyrion Lannister
   C) Jon Snow
   D) Cersei Lannister

5. Short Answer: In "The Chronicles of Narnia," what is the name of the ancient, sentient wardrobe that leads to the magical world of Narnia, and who is its first known user?
   A) The Pevensie wardrobe - Peter, Susan, Edmund, and Lucy Pevensie
   B) Jadis the White Witch's cloak - transforming into a portal to Narnia
   C) The Four Lions' coat of arms - a concealed entrance to Narnia

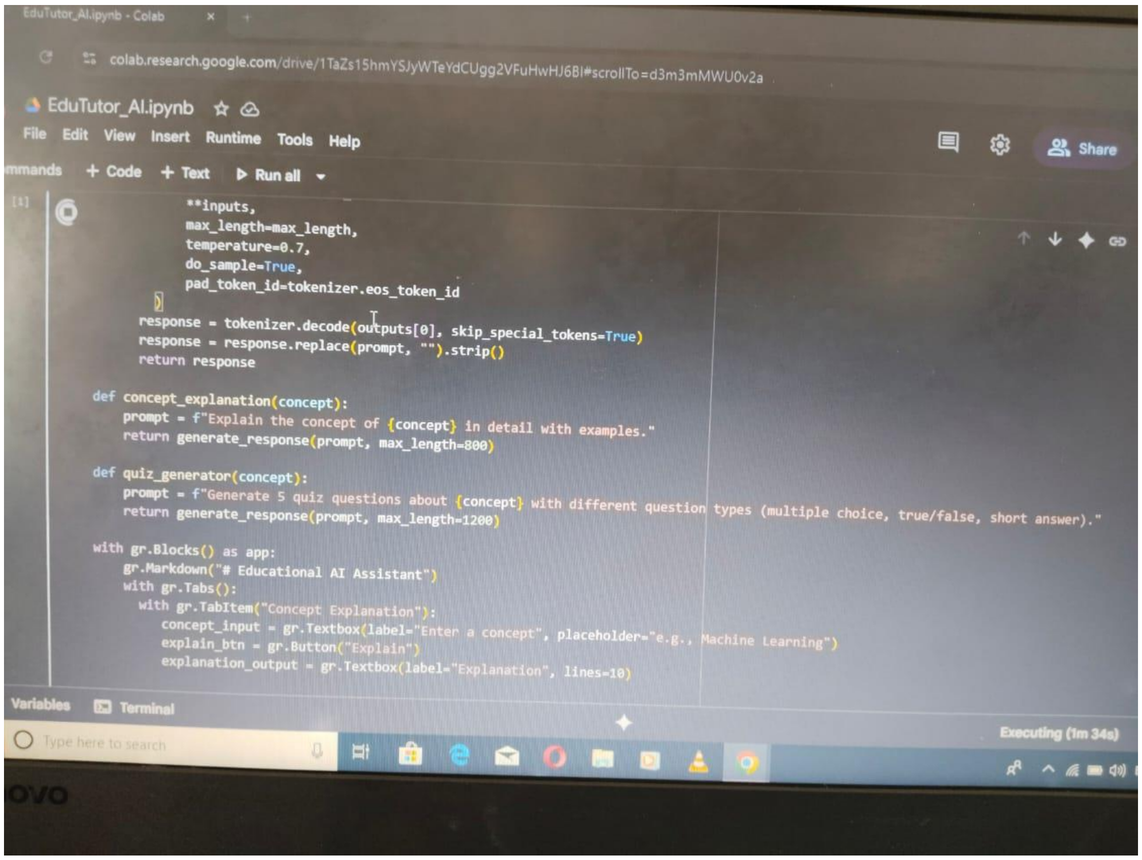
Answers:
1. A) Coruscant
2. True
3. A) Bowtruckle - a tree spirit with a burl sprouting from its back
4. B) Tyrion Lannister

```
pip install gradio torch transformers
```

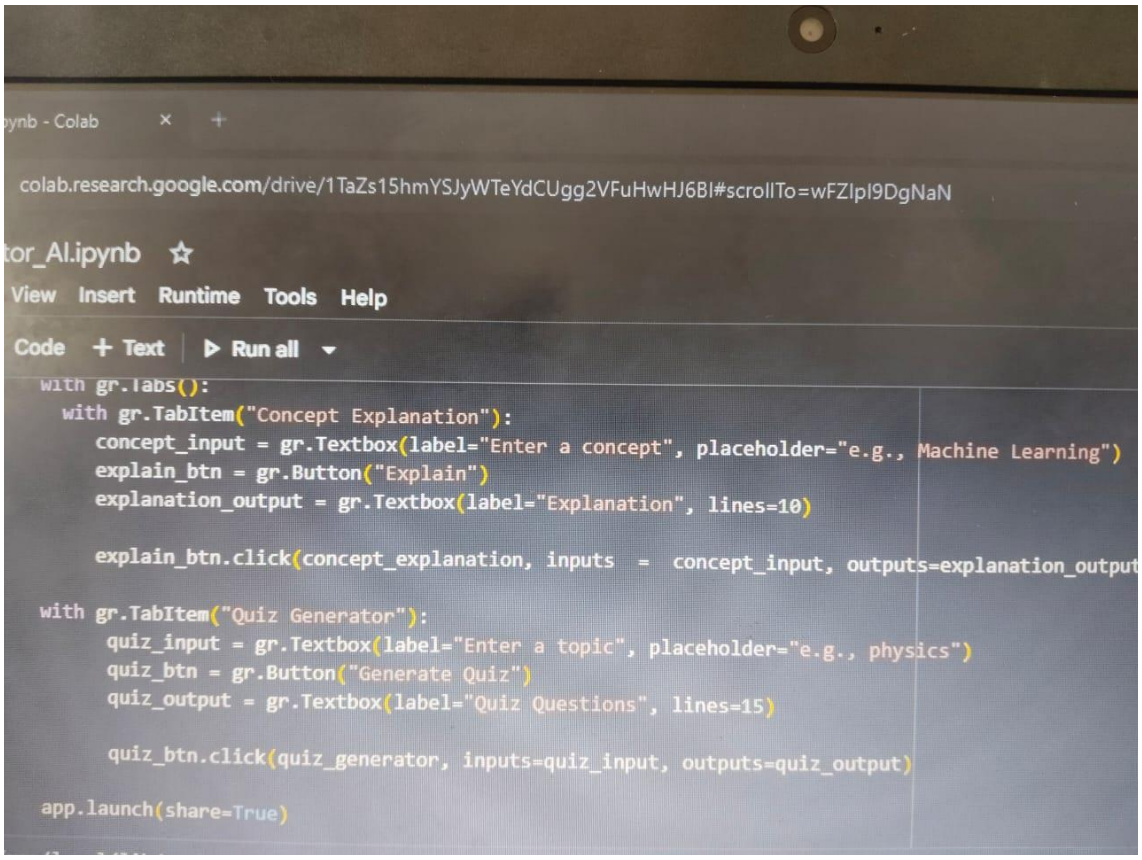
10. CODING



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

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
```



```
                **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)."
    return generate_response(prompt, max_length=1200)

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)
```


ynb - Colab
colab.research.google.com/drive/1TaZs15hmYSJyWTeYdCUgg2VFuHwHJ6Bl#scrollTo=wFZIpl9DgNaN
tor_AI.ipynb
View Insert Runtime Tools Help
Code + Text Run all

```
  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)
```

11.Known Issue:

EduTutor AI, a personalized learning tool, has several known issues that are common to AI-powered educational platforms. Here are some potential concerns:
Bias in AI Algorithms: The AI model may replicate prejudices present in the training data, which can lead to unfair treatment of certain students. Regular inspection and fine-tuning of algorithms are necessary to minimize bias.
Data Privacy and Security: EduTutor AI handles sensitive student information, which must be protected according to regulations like GDPR or FERPA. Ensuring the security of student data is crucial.

Reliability and Credibility of AI-Generated Content: The platform's AI-generated content may not always be accurate or of high quality, which can impact learning outcomes. Continuous evaluation and improvement of the AI model are essential.
Accessibility and Equity: Not all students have equal access to technology and internet connectivity, which can create disparities in learning experiences. EduTutor AI should be designed to accommodate diverse learning needs and environments.
Integration with Existing Systems: Seamless integration with Learning Management Systems (LMS) like Google Classroom is vital for EduTutor AI's effectiveness. Technical issues or compatibility problems can hinder the learning experience.
Teacher Training and Acceptance: Educators may require training to effectively use EduTutor AI and interpret its insights. Resistance to adopting new technology can also impact the platform's success.

Some potential technical issues with EduTutor AI include :
Model Limitations: The primary model used, google/flan-t5-xl, may have limitations in certain subjects or contexts.
Local Inference and API Usage: Technical issues may arise when using local inference or Hugging Face Inference API.
System Requirements: Ensuring compatibility with various devices and browsers is essential for smooth functionality.

12.Future Enhancement:

EduTutor AI has several potential future enhancements that can improve its functionality and user experience. Some of these enhancements include :
User Login/Authentication System: Implementing a secure login system to protect user data and track progress.

AEduTutor AI has several potential future enhancements that can improve its functionality and user experience. Some of these enhancements include :
User Login/Authentication System: Implementing a secure login system to protect user data and track progress.

Admin Dashboard: Creating a dashboard for administrators to upload and modify questions, track user performance, and manage the platform.

Detailed Quiz Analytics: Providing insights into user performance, including time taken, category-wise scores, and areas for improvement.

Multi-Language Support: Expanding the platform to support multiple languages, such as Telugu, Hindi, and English.

Voice Input and Text-to-Speech Output: Integrating voice input and text-to-speech features to make the platform more accessible.

Database Integration: Using a database like Firebase or Supabase to store quiz data and user information.

Model Switching Interface: Allowing users to choose between different AI models for personalized learning.

Live Leaderboards & Progress Tracking: Displaying leaderboards and tracking user progress to encourage engagement and motivation.

Additionally, EduTutor AI can explore features like :

AI-powered Flashcards: Creating digital flashcards to aid in memorization and retention.

Personalized Learning Paths: Using AI to tailor learning paths to individual users' needs and abilities.

Intelligent Tutoring Systems: Providing real-time feedback and guidance to users.
Automated Grading and Assessment: Using AI to grade assignments and provide constructive feedback.

Virtual and Augmented Reality Integration: Incorporating VR/AR technology to create immersive learning experiences.

These enhancements can help EduTutor AI become a more comprehensive and effective learning platform.dmin Dashboard: Creating a dashboard for administrators to upload and modify questions, track user performance, and manage the platform.
Detailed Quiz Analytics: Providing insights into user performance, including time taken, category-wise scores, and areas for improvement.

Multi-Language Support: Expanding the platform to support multiple languages, such as Telugu, Hindi, and English.

Voice Input and Text-to-Speech Output: Integrating voice input and text-to-speech features to make the platform more accessible.

Database Integration: Using a database like Firebase or Supabase to store quiz data and user information.

Model Switching Interface: Allowing users to choose between different AI models for personalized learning.

Live Leaderboards & Progress Tracking: Displaying leaderboards and tracking user progress to encourage engagement and motivation.

Additionally, EduTutor AI can explore features like :
AI-powered Flashcards: Creating digital flashcards to aid in memorization and retention.

Personalized Learning Paths: Using AI to tailor learning paths to individual users' needs and abilities.