**GOVERNMENT ARTS AND SCIENCE COLLEGE**

POONTHOTAM STREET, THERADI, THIRUVOTTIYUR

CHENNAI – 600 019

**Topic : Citizen AI – Intelligent Citizen Engagement Platform**

A NAAN MUDHALVAN PROJECT

**Submitted in Partial Fulfillment for the Award of**

**BATCHELOR OF COMPUTER APPLICATION**

BY

**JAGAN S (212304863)**

**IYANNAR V (212304862)**

**HEMANATHAN R (212304861)**

**YAWANTH P (212304856)**

**EBI R (212304857)**

**TABLE OF CONTENTS**

1**.**Introduction

2.Project Overview

3.Architecture

4.Setup Instruction

5.Folder /Notebook Structure

6.Running the Application

7.Code Overview

8.Testing

9.User Interface

10.Future Enhancements

11.Screenshot

12.Known Issues

**ABSTRACT**

Edutor AI is an educational assistant designed to enhance the learning experience by providing detailed explanations and interactive quizzes on various topics. Built using Hugging Face’s powerful language models and deployed through Google Colab, this tool offers real-time responses to student queries without the need for complex setups. With a simple and user-friendly interface powered by Gradio, users can easily access in-depth concept explanations and practice quizzes that are automatically generated. The assistant aims to support self-learning, foster engagement, and assist students in better understanding academic concepts by providing personalized educational content in an efficient and scalable manner.

**1.Introduction:**

In today’s fast-paced world, students often struggle to find reliable and instant educational resources that cater to their individual learning needs. Edutor AI addresses this challenge by leveraging state-of-the-art artificial intelligence models to offer a digital learning companion. Using pretrained models from the Hugging Face library and the cloud computing environment of Google Colab, Edutor AI enables students to explore complex topics, clarify doubts, and practice through quizzes—all in a seamless and interactive manner.

The primary objective of this project is to simplify learning and make educational support more accessible to students across various subjects. By providing detailed explanations and quiz questions on demand, the assistant helps users reinforce their knowledge and build confidence. The integration with Gradio ensures that the interface is intuitive and easy to use, allowing learners to interact with the system through text inputs and receive immediate feedback.

This documentation outlines the architecture, setup instructions, functionalities, and future improvements for Edutor AI, demonstrating its potential to transform the way students approach self-guided learning.

**2. Project Overview:**

**Purpose:**

The purpose of Edutor AI is to support students and learners by providing concept explanations and quiz generation using advanced AI models. By leveraging the Hugging Face Transformers library and Google Colab, the assistant delivers real-time answers, explanations, and personalized quizzes. The tool aims to simplify learning and improve educational engagement through interactive AI-powered tools.

**Features:**

Concept Explanation

Key Point: Detailed learning support

Functionality: Provides in-depth explanations with examples for various topics.

Quiz Generator

Key Point: Interactive practice tool

Functionality: Generates quizzes with multiple question types (MCQs, true/false, short answers) along with solutions.

User-Friendly Interface

Key Point: Easy access through Gradio

Functionality: Simple and clean UI allowing users to input queries and view responses.

Real-Time Interaction

Key Point: Immediate feedback

Functionality: Provides answers dynamically based on user inputs.

Scalable Architecture

Key Point: Extendable platform

Functionality: Can be scaled with additional topics, quizzes, or AI models.

**3. Architecture:**

**Frontend (Gradio):**

The frontend is built with Gradio to create an intuitive interface where users can input a concept or topic and receive responses instantly. The interface is structured with tabs for concept explanations and quiz generation.

**Backend (Hugging Face Transformers):**

The backend uses pretrained models from Hugging Face to process natural language queries and generate educational content. The models are loaded using the AutoTokenizer and AutoModelForCausalLM classes. It handles requests, processes inputs, and returns AI-generated outputs.

**Execution Environment (Google Colab):**

Google Colab provides a cloud-based environment with GPU support to run the models efficiently. Users only need internet access to interact with the assistant without setting up a local development environment.

**4. Setup Instructions:**

**Prerequisites:**

Google Account to access Google Colab.

Internet access.

No installation required apart from necessary Python libraries (transformers, torch, gradio).

**Steps:**

1. Open the provided Colab notebook.

2. Run the first cell to install required libraries:

!pip install transformers torch gradio -q

3. Load the model and tokenizer using the provided code.

4. Run all cells sequentially to initialize the app.

5. Access the Gradio interface and start interacting with the assistant.

**5. Folder / Notebook Structure:**

Since this is a Colab-based project, the structure is simplified:

Edutor AI Notebook

Model loading

Function definitions (generate\_response, concept\_explanation, quiz\_generator)

Gradio interface setup

Dependencies

transformers – Model loading and text processing.

torch – Efficient tensor operations, GPU usage.

gradio – Interactive user interface.

**6. Running the Application:**

1. Open the Colab notebook.

2. Install required libraries.

3. Load the model and tokenizer.

4. Launch the Gradio app using:

app.launch(share=True)

5. Use the interface to:

Enter a topic and get explanations.

Generate quizzes with answers.

6. Share the app link with others for interactive learning.

**7. Code Overview:**

import gradio as gr

import torch

from transformers import AutoTokenizer, AutoModelForCausalLM

# Load model and tokenizer

model\_name = "ibm-granite/granite-3.2-2b-instruct" # Replace with appropriate Hugging Face model

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)

with gr.Blocks() as app:

gr.Markdown("# Edutor AI – Educational 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)

**8. Testing:**

**Test Cases:**

Test with different topics such as "machine learning", "photosynthesis", "algebra".

Check whether explanations include examples.

Ensure quizzes have at least 5 questions with varying formats.

Verify that answers are clearly separated at the end.

Validate performance in both CPU and GPU modes.

**9. User Interface:**

Minimalist and accessible layout using Gradio.

Two main tabs: "Concept Explanation" and "Quiz Generator".

Inputs and outputs are clearly labeled.

Instant feedback with formatted results.

Shareable link for collaborative learning.

**10. Future Enhancements:**

Add voice input/output for better accessibility.

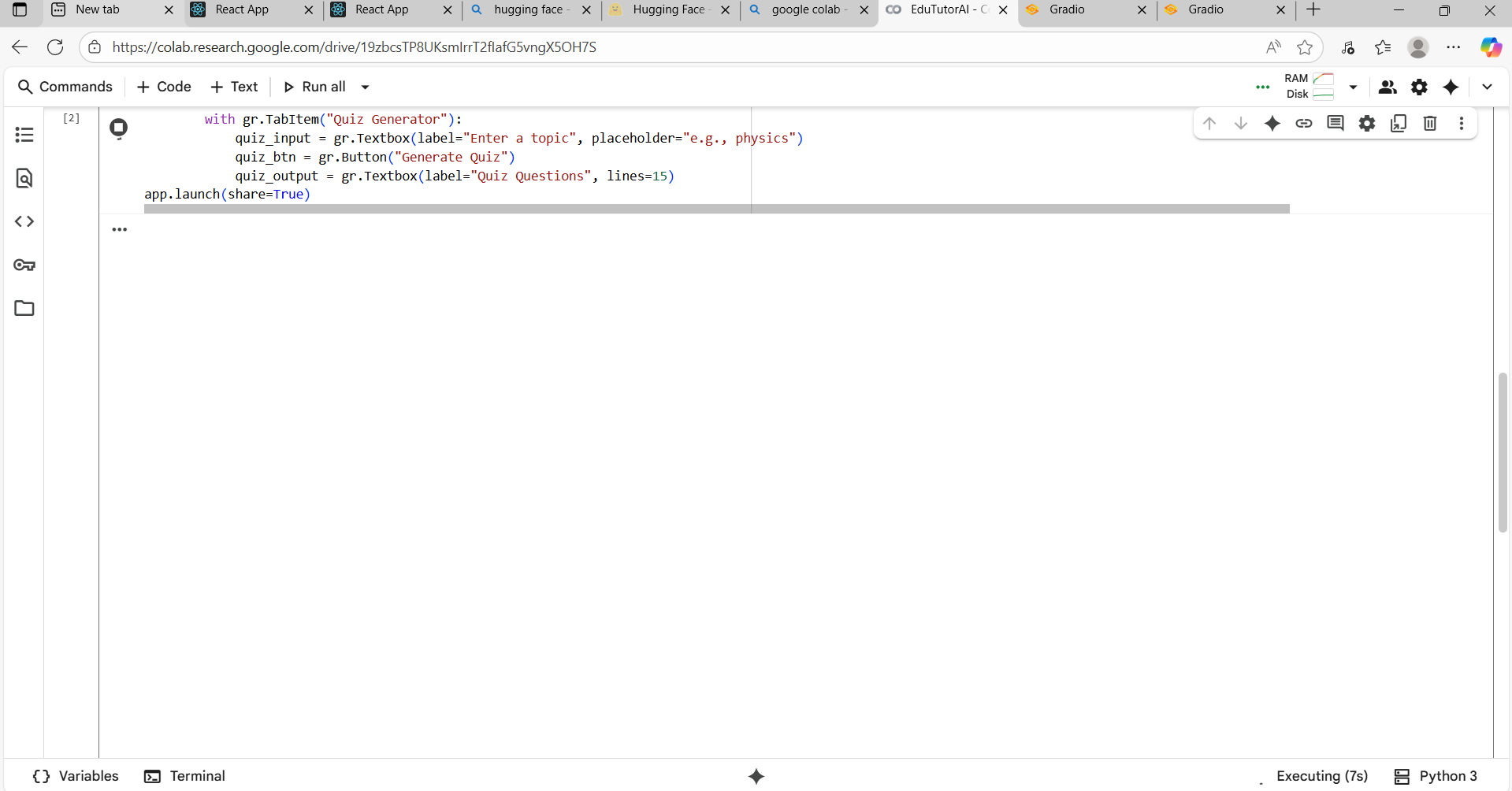
Support additional languages beyond English.

Include user history and progress tracking.

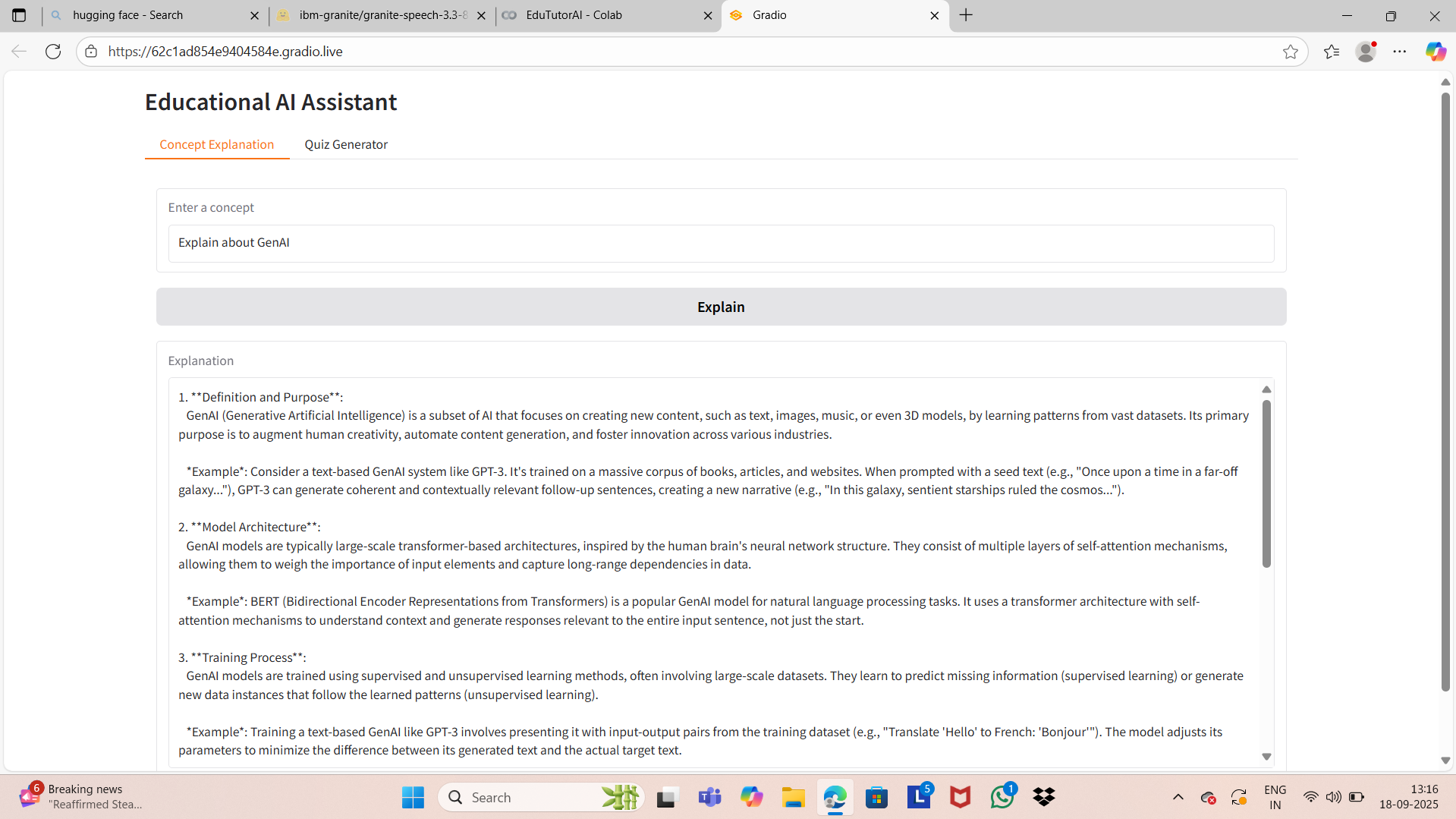
Integrate with educational platforms like Moodle or Google Classroom.

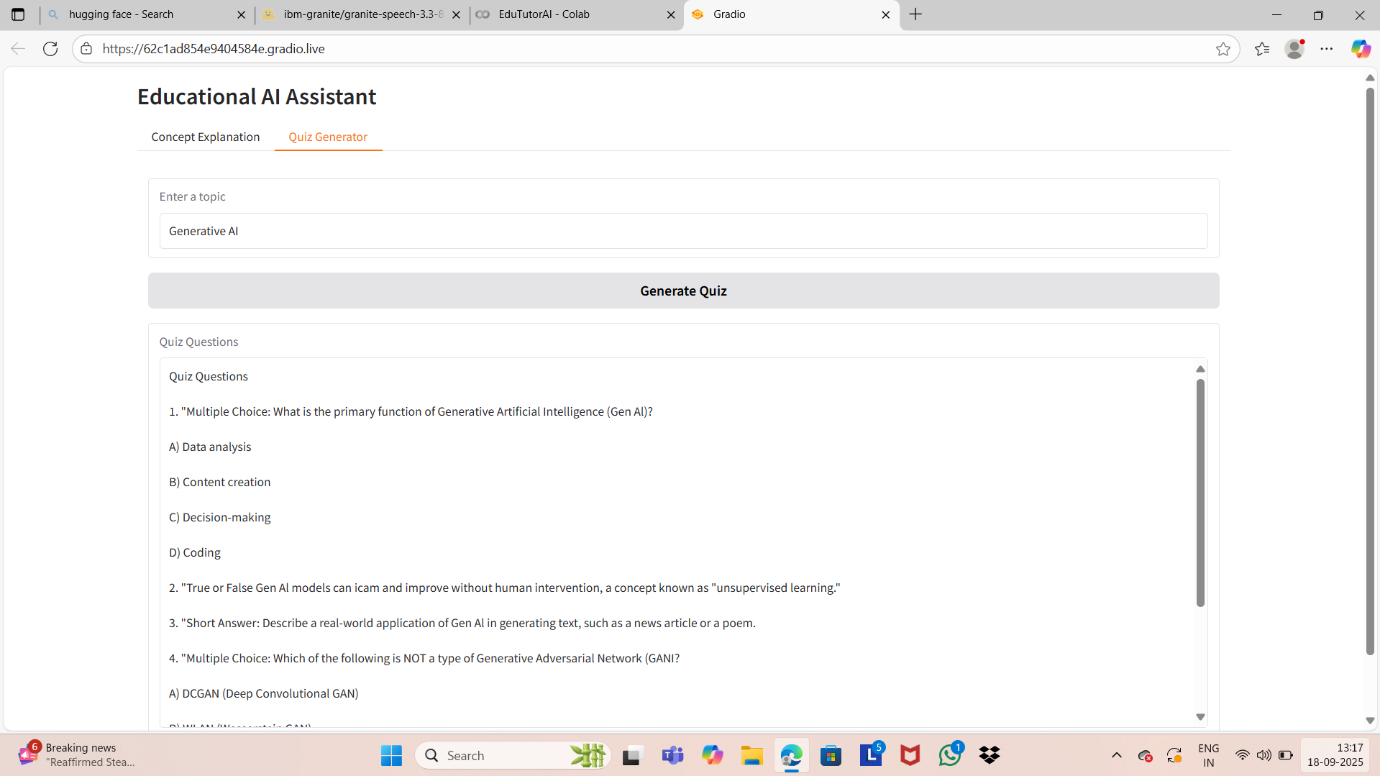
Implement authentication for personalized learning paths.

**11. Screenshots:**









**12. Known Issues:**

Model latency when using large prompts.

Limited understanding in highly technical or ambiguous queries.

Dependency on internet access in Google Colab environment.

Occasional token truncation when input length exceeds maximum allowed size.