

# EDU TUTOR AI WITH IBM

## Project Documentation

### 1. Introduction

- Project Title: EDUTUTOR AI with IBM
- Team Leader: SARANYA S
- Team Member: KAVISELVI S
- Team Member: LAVANYA S
- Team Member: RAJASRI P

### 2. Project Overview

- Purpose:

EduTutorAI is an AI-powered virtual tutor that helps students learn by explaining concepts clearly and generating practice quizzes.

- Features:

Concept explanation with examples.

Quiz generator with multiple question types and answers.

Simple and interactive Gradio interface.

AI-driven, accurate, and student-friendly responses.

### 3. Architecture

#### 1. Frontend (User Interface – Gradio)

Provides two tabs: Concept Explanation and Quiz Generator.

Accepts user input (concept/topic).

Displays AI-generated explanations or quizzes.

## 2. Backend (Model + Processing)

Tokenizer (Hugging Face AutoTokenizer): Converts text input into tokens.

Model (IBM Granite Instruct LLM): Processes tokens and generates responses.

Response Generator: Formats output by decoding tokens and cleaning text.

## 3. Functions

`concept_explanation(concept)`: Creates a detailed explanation with examples.

`quiz_generator(concept)`: Generates quiz questions + answers section.

`generate_response(prompt)`: Core function handling prompt → model → output.

#### 4. Execution Flow

User enters a topic → Gradio sends input → Tokenizer encodes → Granite LLM generates output → Response is decoded and displayed in UI.

#### 5. Deployment

Runs locally or on cloud.

`app.launch(share=True)` allows sharing via public link.

#### 4. Setup Instructions

Prerequisites:

- Python Programming Knowledge
- Gradio Framework
- IBM Granite Model Access (via Hugging Face)
- Google Colab with T4 GPU
- GitHub Account Steps:
  1. Access the Naan Mudhalvan Smart Internz portal.
  2. Choose an IBM Granite model from Hugging Face.
  3. Run the application in Google Colab with required libraries.
  4. Upload final project files to GitHub.

## 5.Folder Structure

- app/ – Backend logic and integration.
- ui/ – Gradio app interface files.
- edututor\_ai.py – Main application file.
- model\_loader.py – Handles IBM Granite model integration.
- dashboard.py – Visualization of edu tutor feedback.

## 6.Running the Application

1. Open Google Colab and load the project notebook.
2. Install dependencies and configure runtime with GPU.
3. Run the notebook cells to start the Gradio app.
4. Access the provided link to interact with Citizen AI.

## 7.API Documentation

1. generate\_response(prompt, max\_length=512)

Purpose: Core function to interact with the model.

Input: prompt (string), max\_length (int, default 512).

Output: AI-generated response (string).

2. concept\_explanation(concept)

Purpose: Explains a concept in detail with examples.

Input: concept (string).

Output: Explanation text.

3. quiz\_generator(concept)

Purpose: Generates 5 quiz questions + answers.

Input: concept (string).

Output: Quiz questions and an ANSWERS section.

4. Interface (Gradio)

Tab 1: Concept Explanation → Input: concept → Output: explanation.

Tab 2: Quiz Generator → Input: topic → Output: quiz + answers.

## 8. Authentication

The current project runs locally with Hugging Face models, so no external authentication (API keys) is required.

If deployed on Hugging Face Hub or a cloud server, you may need a Hugging Face Access Token to load private models.

## 9. User Interface

EduTutorAI has a simple Gradio web interface with two tabs: Concept Explanation (input a concept → get detailed explanation) and Quiz Generator (input a topic → get 5 quiz questions with answers), accessible on any device via browser.

## 10. Testing

Concept explanations are clear and relevant.

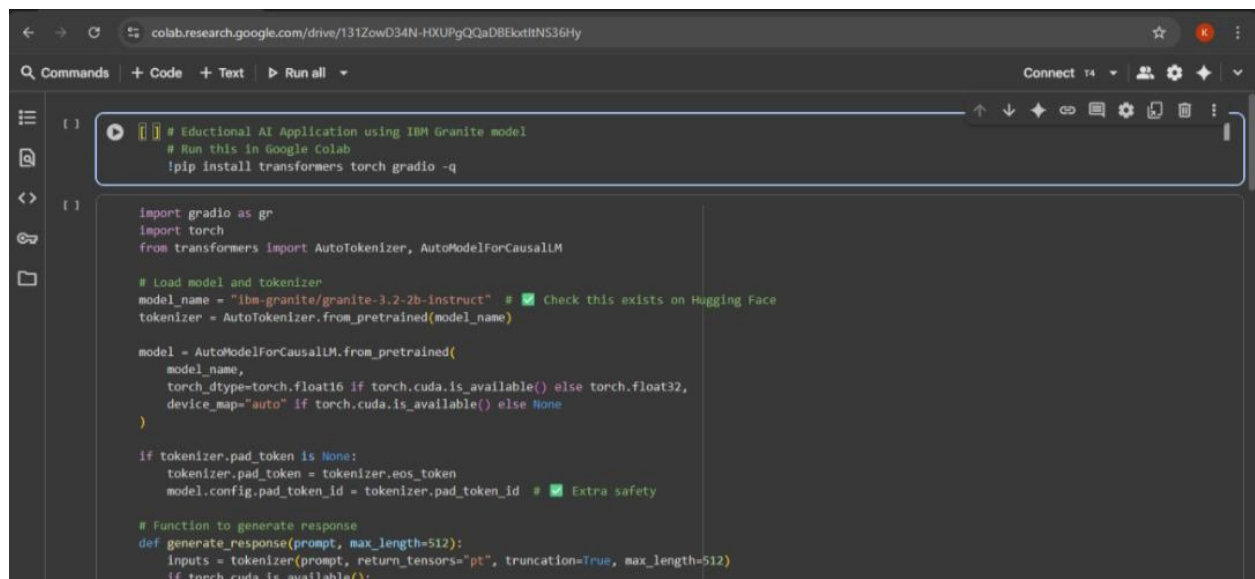
Quiz generator produces 5 varied questions with answers.

Gradio UI buttons, tabs, and outputs work properly.

Works on both CPU and GPU without errors.

Responses are accurate and user-friendly.

## 11. Screenshots



The screenshot shows a Google Colab notebook interface. The top bar includes navigation icons, a URL bar with the Colab link, and a 'Connect' button. Below the top bar, there are tabs for 'Commands', 'Code', 'Text', and 'Run all'. The main area displays a Jupyter notebook with the following code:

```
[ ] # EduTutorAI 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" # Check this exists on Hugging Face
tokenizer = AutoTokenizer.from_pretrained(model_name)

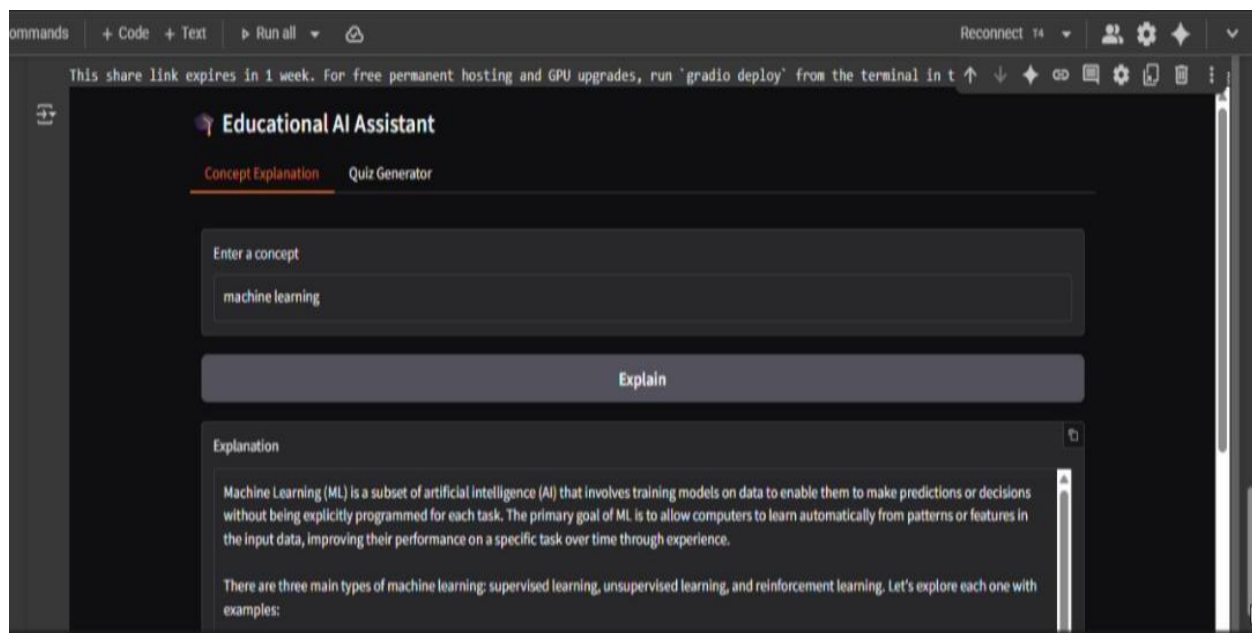
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto" if torch.cuda.is_available() else None
)

if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token
    model.config.pad_token_id = tokenizer.pad_token_id # Extra safety

# Function to generate response
def generate_response(prompt, max_length=512):
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)
    if torch.cuda.is_available():
```

```
tokenizer_config.json: 8.88k/? [00:00<00:00, 358kB/s]
vocab.json: 777k/? [00:00<00:00, 6.93MB/s]
merges.txt: 442k/? [00:00<00:00, 11.8MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 38.0MB/s]
added_tokens.json: 100% [87.0/87.0] [00:00<00:00, 2.13kB/s]
special_tokens_map.json: 100% [701/701] [00:00<00:00, 13.5kB/s]
config.json: 100% [786/786] [00:00<00:00, 34.6kB/s]
model.safetensors.index.json: 29.8k/? [00:00<00:00, 1.71MB/s]
Fetching 2 files: 100% [2/2] [01:24<00:00, 84.95s/t]
model-00002-of-00002.safetensors: 100% [67.1M/67.1M] [00:01<00:00, 40.4MB/s]
model-00001-of-00002.safetensors: 100% [5.00G/5.00G] [01:24<00:00, 127MB/s]
Loading checkpoint shards: 100% [2/2] [00:22<00:00, 9.23s/t]
generation_config.json: 100% [137/137] [00:00<00:00, 12.5kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://f265e09bfa5e273c7e.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 Hugg
```



## 12. Known Issues

Responses may sometimes be too long or repetitive.

Quiz questions may vary in difficulty and not always be balanced.

Explanations depend on model knowledge (may miss very recent topics).

Requires good internet and hardware for smooth performance.

### 13.Future Enhancements

Add support for voice input/output for accessibility.

Include multilingual support for regional languages.

Provide difficulty levels for quiz generation (easy, medium, hard).

Add progress tracking for learners.

Deploy as a mobile app for easier student access.