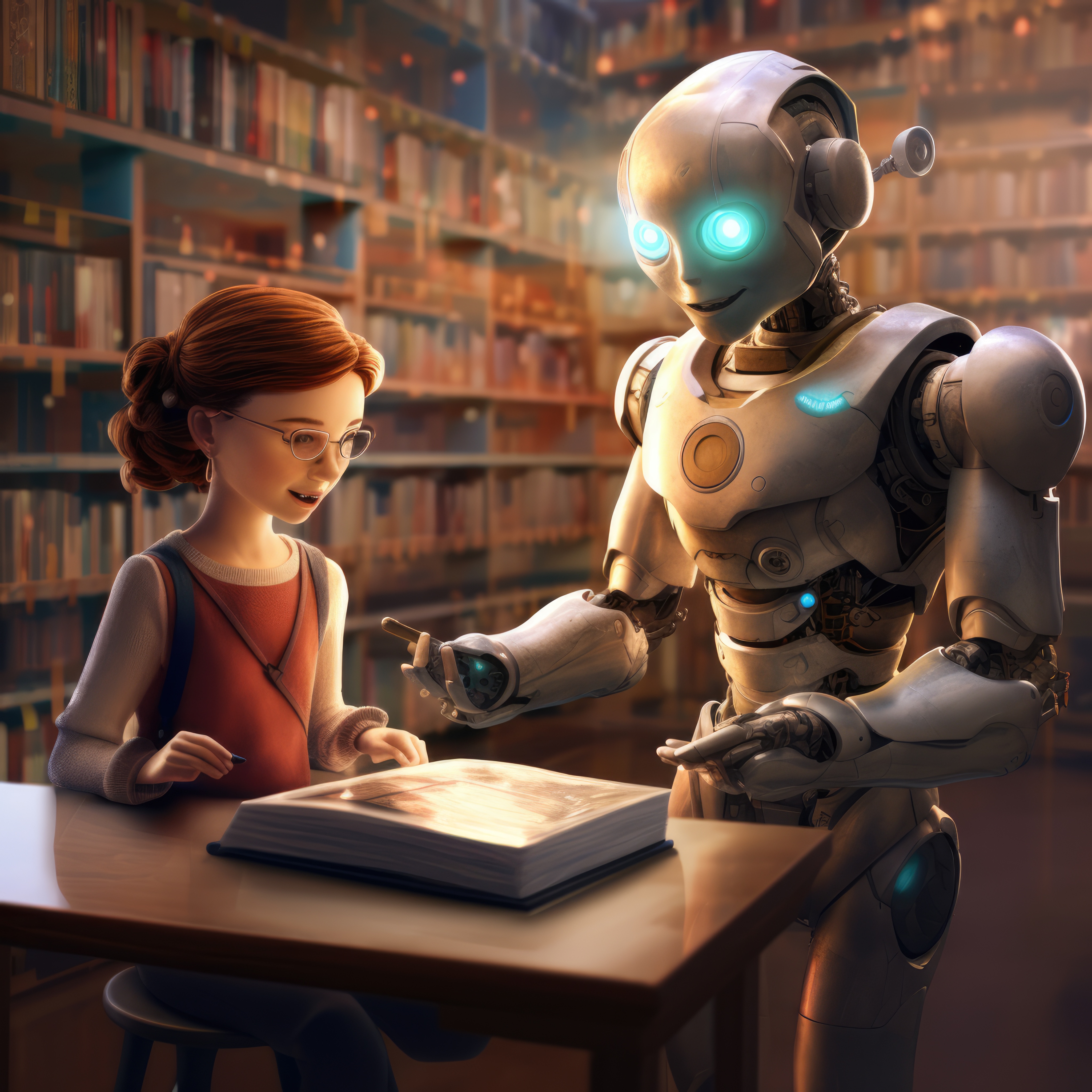
**Edu Tutor AI: Personalized**

**Learning**

GenerateAIwithIBM

# 



**Project Documentation**

## 1. Introduction

* Project Title: EduTutor AI
* Team Member:

1. Sujitha T

2. Thilageswari S

3. Vinothini V

4. Yuvetha J

## 2. Project Overview

Purpose:

EduTutor AI is designed to act as an intelligent educational assistant for learners. It leverages Large Language Models (LLMs) to explain academic concepts in simple language and generate quizzes dynamically. The assistant enhances self-learning by offering both explanations and assessments in an interactive way.

Features:

* Concept Explanation

Key Point: AI-driven explanations

Functionality: Provides detailed explanations of academic concepts with real-life examples.

* Quiz Generator

Key Point: Assessment support

Functionality: Generates 5 quiz questions on a given topic with multiple formats (MCQs, True/False, Short Answer) and provides an ANSWERS section.

* Conversational Interface

Key Point: Natural language interaction

Functionality: Learners can ask for explanations or quizzes in plain text.

* User-Friendly Gradio Interface

Key Point: Simple and interactive UI

Functionality: Web-based tabs for concept explanation and quiz generation.

## 3. Architecture

* Frontend (Gradio): Provides a clean and interactive web UI with tabs for “Concept

Explanation” and “Quiz Generator.”

* Backend (Transformers + PyTorch): Uses IBM Granite LLM for text generation. Handles tokenization, model inference, and GPU acceleration.
* LLM Integration (Hugging Face Transformers): Tokenizer ensures input formatting. Model generates structured responses.

## 4. Setup Instructions

Prerequisites:

* Python 3.9+
* pip and virtual environment tools
* Gradio
* Hugging Face Transformers
* Torch with CUDA support (optional)

Installation:

1. Clone the repository
2. Install dependencies: pip install -r requirements.txt
3. Run the app: python app.py
4. Access the Gradio link to interact with EduTutor AI

## 5. Folder Structure

EduTutor-AI/

│── app.py # Main application script

│── requirements.txt # Dependencies

│── docs/ # Documentation files

│── models/ # Pretrained model references

│── utils/ # Helper functions (if extended later)

## 6. Running the Application

* Launch the Gradio interface by running app.py
* Navigate between Concept Explanation and Quiz Generator tabs
* Input the desired topic and view the AI-generated output in real time

## 7. API Documentation

(Optional for future extension)

* POST /explain – Returns AI-generated explanation of a concept
* POST /quiz – Generates quiz questions with answers

## 8. Authentication

Currently open environment. Future enhancements:

* Token-based authentication
* User session tracking for personalized learning

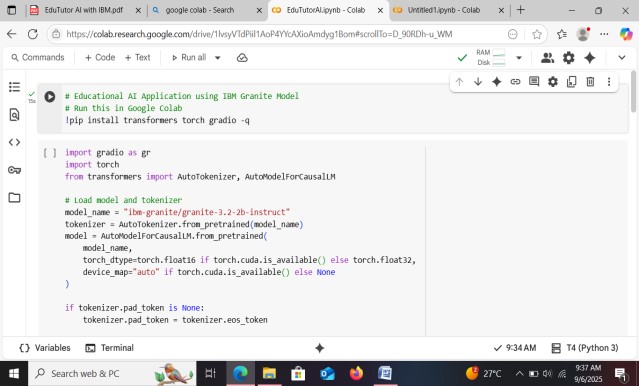
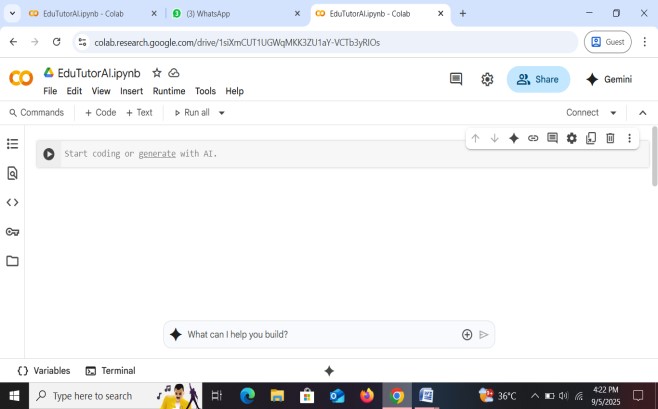
## 9. User Interface

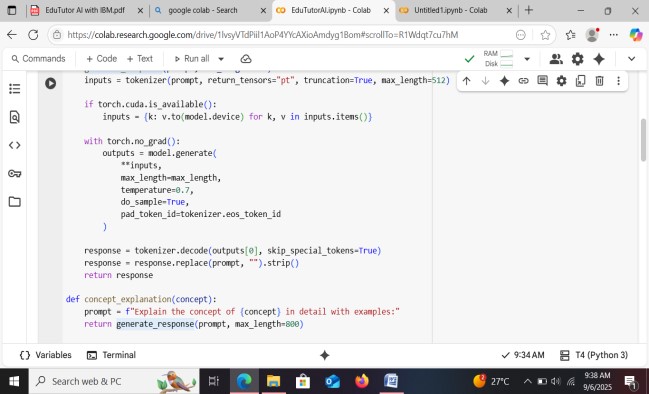
* Tabs for concept explanation and quiz generation
* Textbox inputs for user queries
* Textbox outputs for displaying explanations and quizzes
* Minimalist design for accessibility

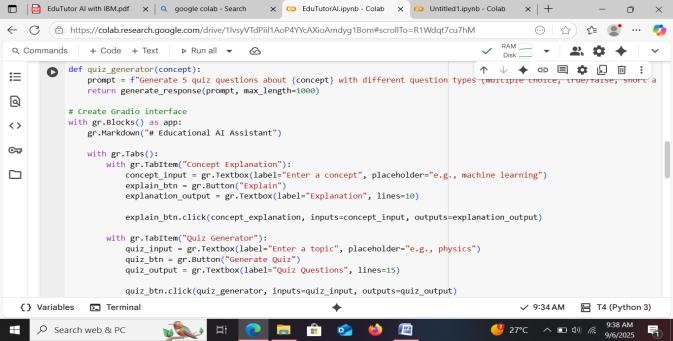
## 10. Testing

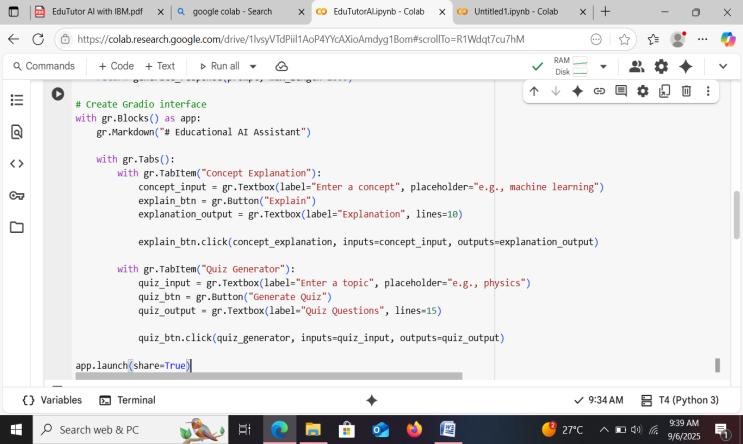
* Unit Testing: Prompt generation and tokenizer handling
* Manual Testing: Interaction through Gradio UI
* Edge Cases: Empty inputs, large prompts, GPU/CPU compatibility

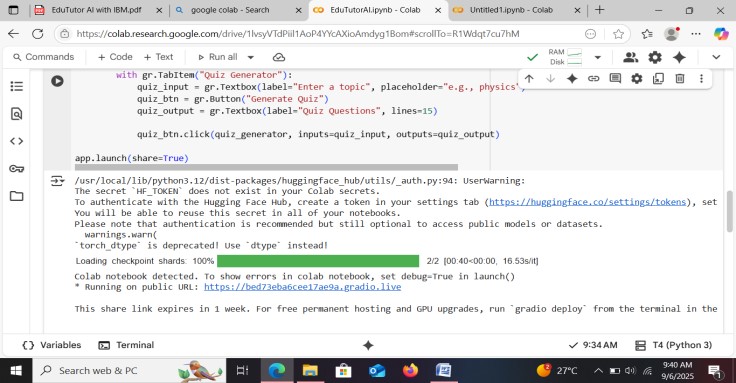
## 11. Screenshots

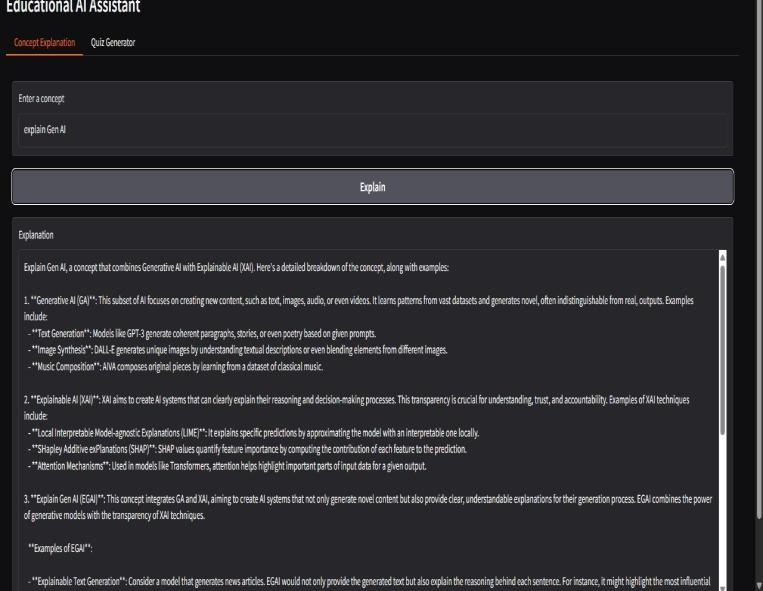


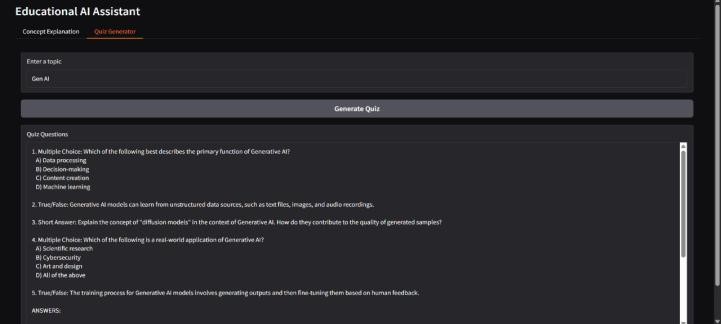












## 12. Known Issues

* Long explanations may exceed output box size
* Limited to topics supported by the pretrained Granite model

## 13. Future Enhancements

* Add voice-based queries
* Extend to subject-specific tutors (Math, Science, etc.)
* Track user progress with learning analytics dashboards
* Integrate FastAPI backend for API-based usage