# EDU TUTOR AI PROJECT DOCUMENTATION

|  |  |
| --- | --- |
| **DATE** | **10-09-2025** |
| **TEAM ID** | **NM2025TMID00612** |
| **PROJECT NAME** | **EDU TUTOR AI** |
| **Maximum Marks** |  |

# ABSTRACT

This project introduces an AI-powered Educational Tutor designed to support students and educators by simplifying complex topics and promoting active learning. The system is built using the IBM Granite 3.2 2B Instruct language model, integrated through the Hugging Face Transformers library and deployed with a user-friendly Gradio interface. It offers two key features: Concept Explanation, which provides detailed and easy-to-understand descriptions of any academic topic, and Quiz Generation, which creates diverse quiz questions (multiple choice, true/false, and short answer) along with correct answers. By leveraging the power of natural language processing (NLP), this tool enables interactive, personalized, and on-demand learning, making education more accessible and engaging.

# INTRODUCTION

Learning new concepts can sometimes be challenging, especially without personalized help or practice. This project creates an AI-powered Educational Tutor that explains topics clearly and generates quizzes to test understanding. Using the IBM Granite language model and an easy Gradio interface, the tool offers interactive support for students to learn at their own pace. It aims to make studying simpler, more engaging, and accessible anytime, helping learners improve their knowledge effectively.

# USE CASE SCENARIOS

**Scenario 1**: Personalized Learning Experience  
Students sync courses via Google Classroom, receive AI-generated quizzes, and get instant feedback.  
  
**Scenario 2:** Educator Dashboard & Performance Insights  
Educators view quiz history, scores, last topics attempted, and insights via Pinecone vector database.  
  
**Scenario 3:** Diagnostic Testing and Adaptive Quizzing  
Students take a diagnostic test generated by IBM Watsonx; difficulty adapts accordingly.  
  
**Scenario 4:** Google Classroom Integration  
Seamless syncing of student data, subjects, and classes for auto quiz generation.

# ARCHITECTURE

* **Modular & Scalable:** Designed for flexibility and easy updates.
* **Frontend:** React or Streamlit for interactive UI, dashboards, and quizzes.
* **Backend:** FastAPI provides Restfully APIs for quizzes, feedback, and authentication.
* **LLM Integration:** IBM Watsonx and Granite generate quizzes and adaptive feedback.
* **Vector Database:** Pinecone stores quiz data and user performance for personalization.

# MILESTONE 1: REQUIREMENTS SPECIFICATION

This milestone outlines the essential libraries and tools required for setting up the project environment. The dependencies include frameworks for building the API, managing authentication, handling vector databases, and creating the user interface.

**Core Libraries:**

* **FastAPI:** A modern, fast (high-performance) web framework for building APIs with Python.
* **Uvicorn:** A lightning-fast ASGI server implementation, used to run FastAPI applications.
* **Langchain IBM:** Integrates IBM’s language models to support natural language processing tasks.
* **Pinecone:** A vector database service used to store and retrieve embeddings for personalized recommendations and performance tracking.
* **Streamlit:** A Python library for building interactive web apps, mainly for frontend UI.
* **Google Auth OAuthlib & Google API Python Client:** Used for handling Google authentication and accessing Google APIs.
* **Python Dotenv:** Manages environment variables securely from .env files.

**Installation Instructions:**

To install all required packages, run the following command in your project directory:

pip install -r requirements.txt

This will ensure that all dependencies are installed with the correct versions for smooth functioning of the project.

# MILESTONE 2: INITIALIZATION OF ENVIRONMENT VARIABLES

To securely manage sensitive information such as API keys and project IDs, create a .env file in your project root directory. This file will store all necessary environment variables for connecting to IBM Watsonx and Pinecone services.

**Create a .env file with the following entries:**

WATSONX\_MODEL\_ID=granite-13b-instruct-v2

WATSONX\_API\_KEY=your\_ibm\_watsonx\_api\_key

WATSONX\_ENDPOINT=https://us-south.ml.cloud.ibm.com

WATSONX\_PROJECT\_ID=your\_project\_id

PINECONE\_API\_KEY=your\_pinecone\_api\_key

PINECONE\_INDEX\_NAME=edututor

**Notes:**

* Replace your\_ibm\_watsonx\_api\_key, your\_project\_id, and your\_pinecone\_api\_key with your actual credentials.
* The .env file should **not** be committed to version control to keep your keys secure.
* The project uses the python-dotenv package to load these variables at runtime.

# MILESTONE 3: AI INTEGRATION WITH IBM WATSONX

This milestone focuses on integrating the IBM Watsonx language model to enable AI-driven quiz generation and response parsing.

* **Model Setup:**  
  The Granite model is loaded using langchain\_ibm.WatsonxLLM, which connects to IBM Watsonx services and allows seamless interaction with the large language model.
* **Prompt Template:**  
  Quiz questions are dynamically generated using LangChain’s PromptTemplate. This enables customizable prompts to produce diverse and context-relevant quiz content.
* **Quiz Parsing:**  
  The raw output from Watsonx is processed and parsed into structured JSON format. This facilitates easier display of quiz questions and automatic evaluation of user responses.

# MILESTONE 4: GOOGLE CLASSROOM SYNC

This milestone integrates Google Classroom features to sync user data and enhance the educational experience.

* **Authentication:**  
  Uses google-auth-oauthlib to enable secure user login via Google accounts.
* **API Integration:**  
  Leverages google-api-python-client to interact with Google Classroom APIs, allowing access to courses, assignments, and user information.
* **User Login Flow:**  
  Users authenticate through their Google accounts, enabling seamless synchronization of classroom data with the AI tutor.

# MILESTONE 5: PINECONE VECTOR DB INTEGRATION

This milestone focuses on integrating Pinecone as the vector database to store and manage user and quiz data efficiently.

* **User Profiles:**  
  Stores each user’s profile along with embeddings that represent their learning progress and behavior.
* **Quiz Metadata:**  
  Updates metadata such as quiz scores, topics, and submission dates after each quiz attempt to track performance over time.
* **Data Access for Educators:**  
  Enables educators to retrieve student data from Pinecone for detailed analysis of learning progress and to provide personalized support.

# MILESTONE 6: STREAMLIT FRONTEND UI

This milestone involves developing the frontend interface using Streamlit, offering separate panels for students and educators.

* **Student Panel:**
  + **Login:** Supports manual login or Google OAuth authentication.
  + **Dashboard:** Displays overview of learning progress and upcoming quizzes.
  + **Take Quiz:** Allows students to attempt quizzes generated by the AI tutor.
  + **Quiz History:** Shows past quiz attempts and scores for self-assessment.
* **Educator Panel:**
  + **Dashboard:** Provides analytics and insights on all students’ performance to help educators monitor progress and tailor support.

# MILESTONE 7: FUNCTIONAL VERIFICATION

This milestone verifies the functionality and integration of the modular AI-powered architecture.

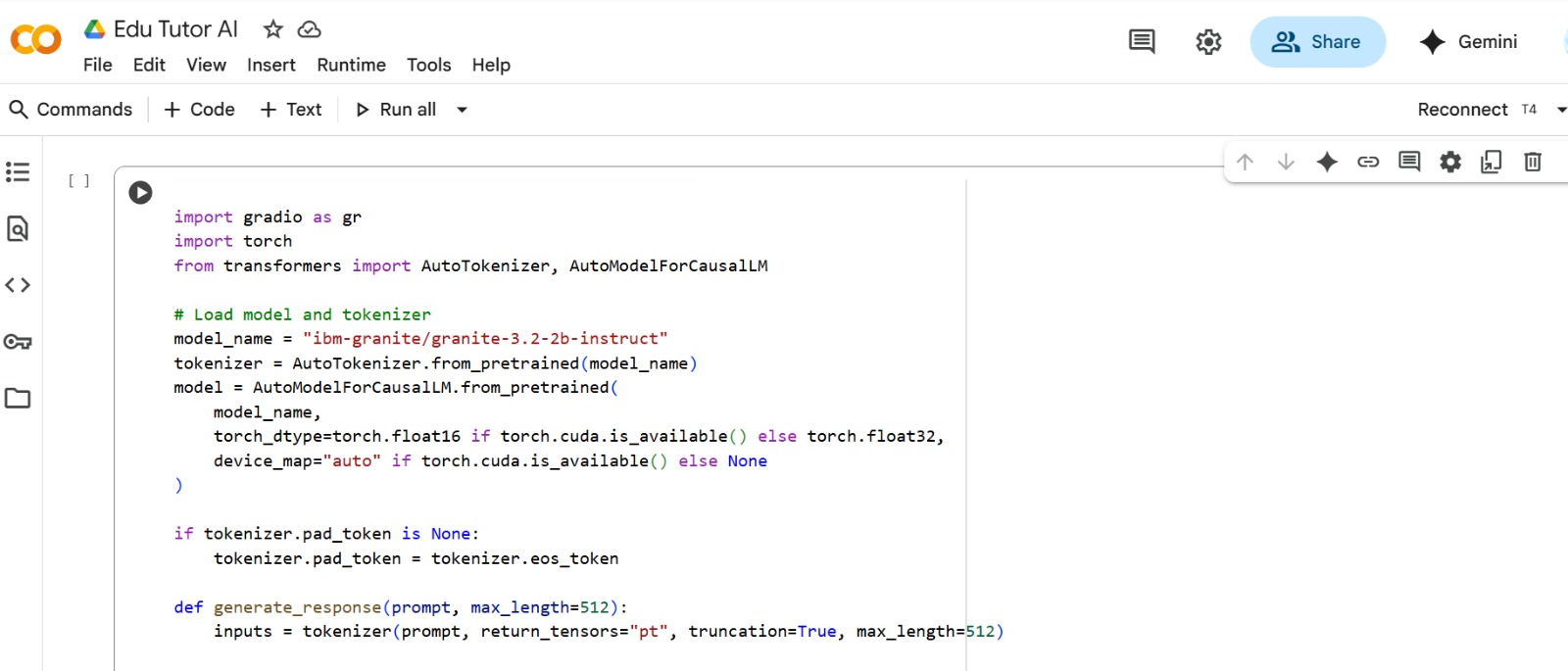
* **FastAPI Backend:**  
  Handles critical operations including student and educator login, quiz generation, answer evaluation, Google Classroom sync, and updating metadata.
* **Watsonx + Granite Models:**  
  Dynamically generate multiple-choice questions (MCQs) based on user-selected topics, using instruction-tuned inference through LangChain’s Watsonx integration.
* **Pinecone Vector Database:**  
  Stores user profile embeddings, quiz history metadata, and supports similarity search to enable adaptive learning experiences.
* **Streamlit Frontend:**  
  Provides role-based dashboards for students and educators, quiz submission forms, and OAuth-based login.  
  **Note:** Some errors were encountered preventing complete frontend build and integration.

# PROJECT FLOW

1. **User Input:**  
   Students log in via manual credentials or Google Classroom and request quizzes by selecting topic and difficulty level.
2. **AI Quiz Generation:**  
   Watsonx + Granite models generate MCQs that are temporarily stored on the frontend without answers; the backend stores quizzes with answers.
3. **User Quiz Submission:**  
   Students submit answers through the UI. The backend evaluates responses, calculates scores, and updates records in Pinecone.
4. **Feedback Loop:**  
   Educators access student performance data via their dashboard for monitoring and personalized support.

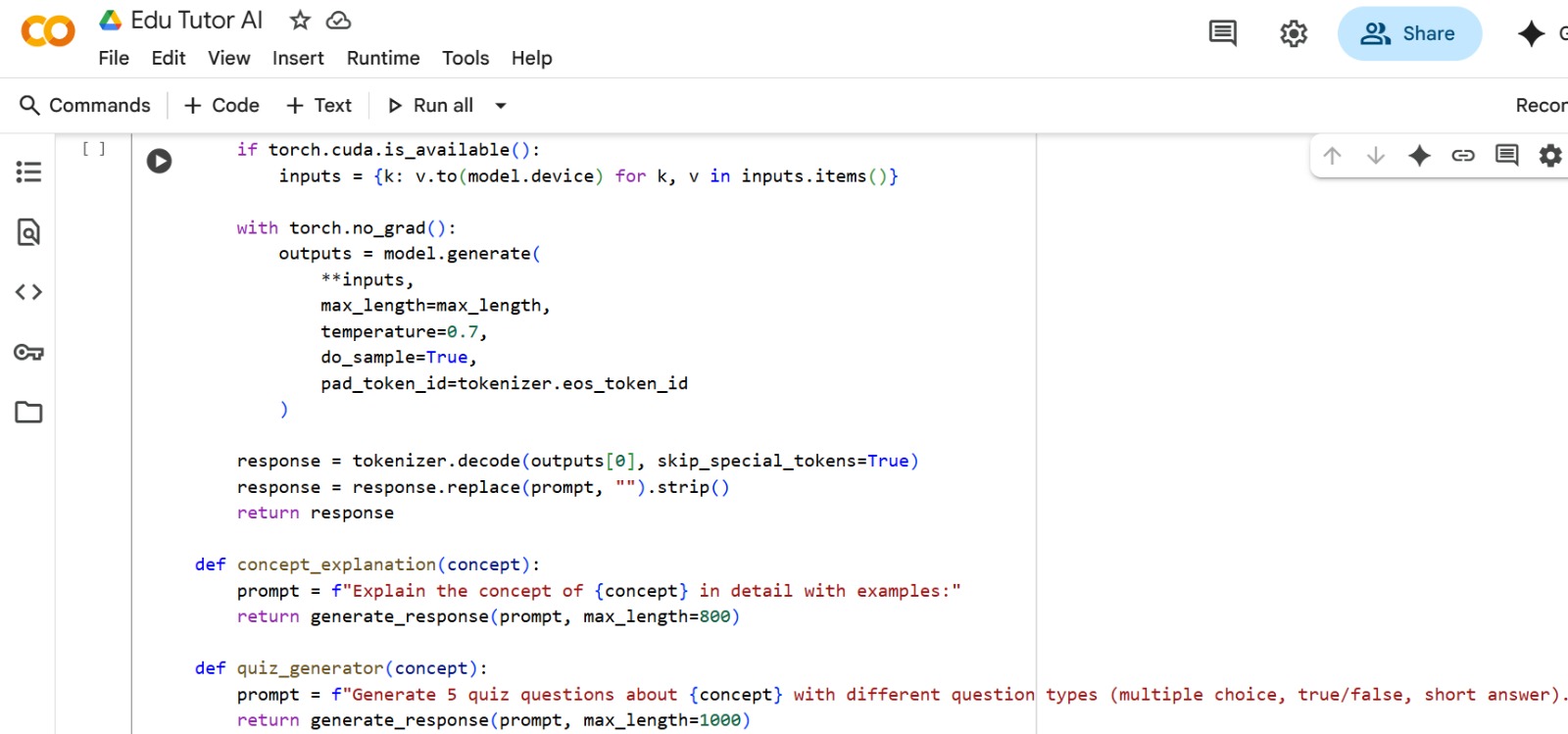
# WORKING OF EDU TUTOR AI

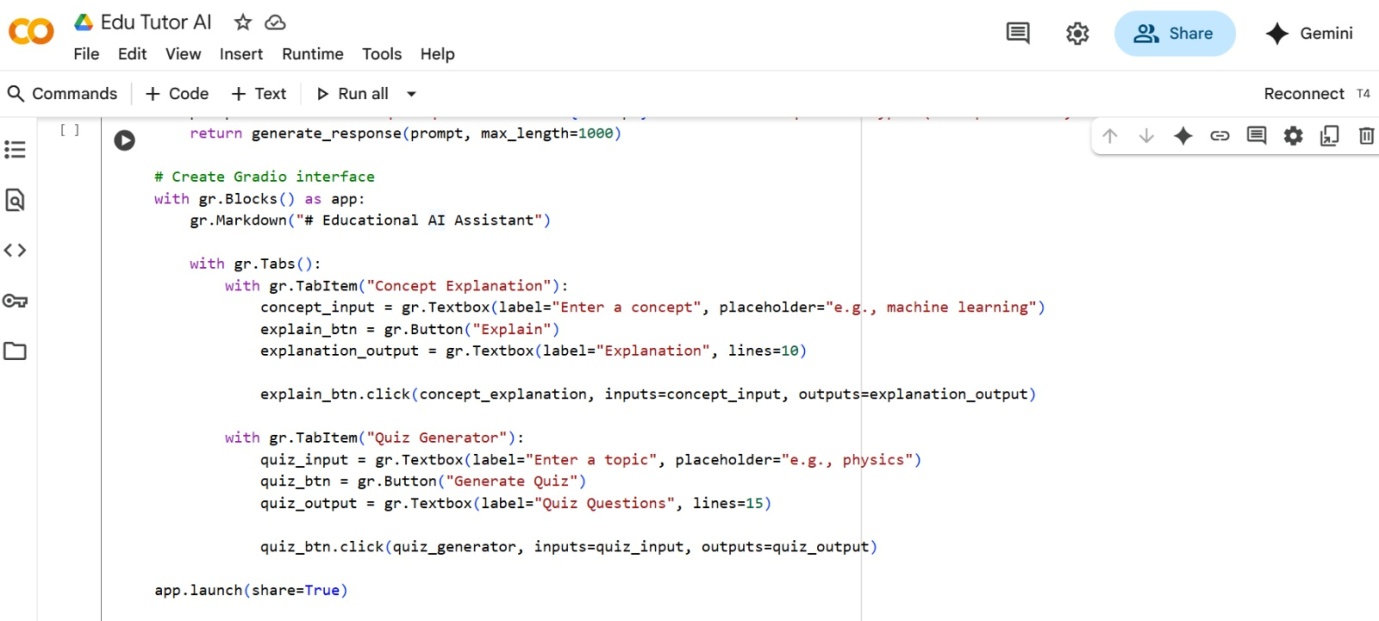
**Code Snippet 1: Model Loading and Tokenizer Initialization**

****

This screenshot shows the code where the Granite Instruct model and tokenizer are loaded. The tokenizer converts text into tokens, and the model processes them to generate responses. A check for GPU availability is also included for faster execution.

**Code Snippet 2: Function Definitions for Explanation and Quiz**

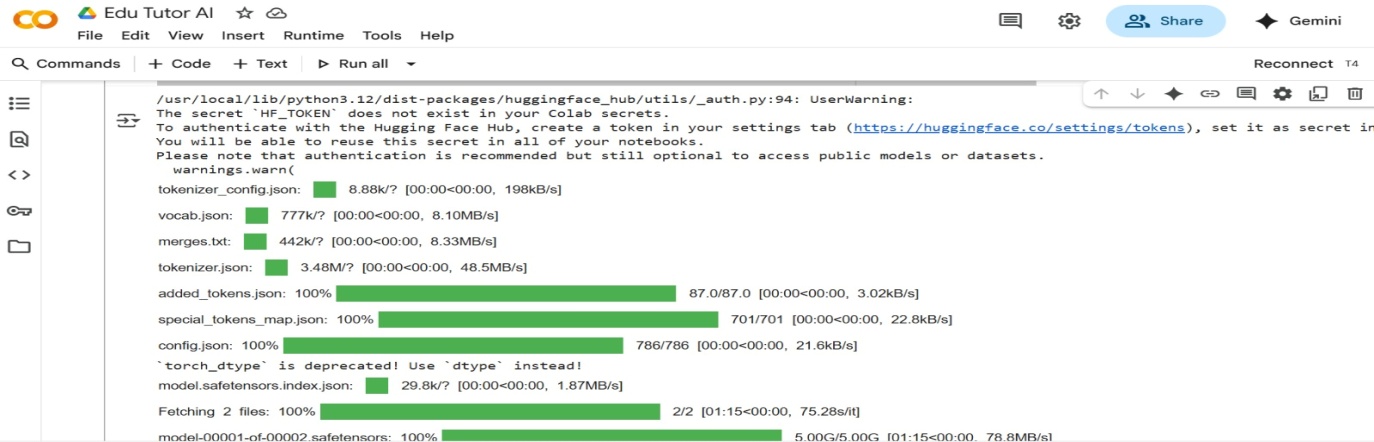
****

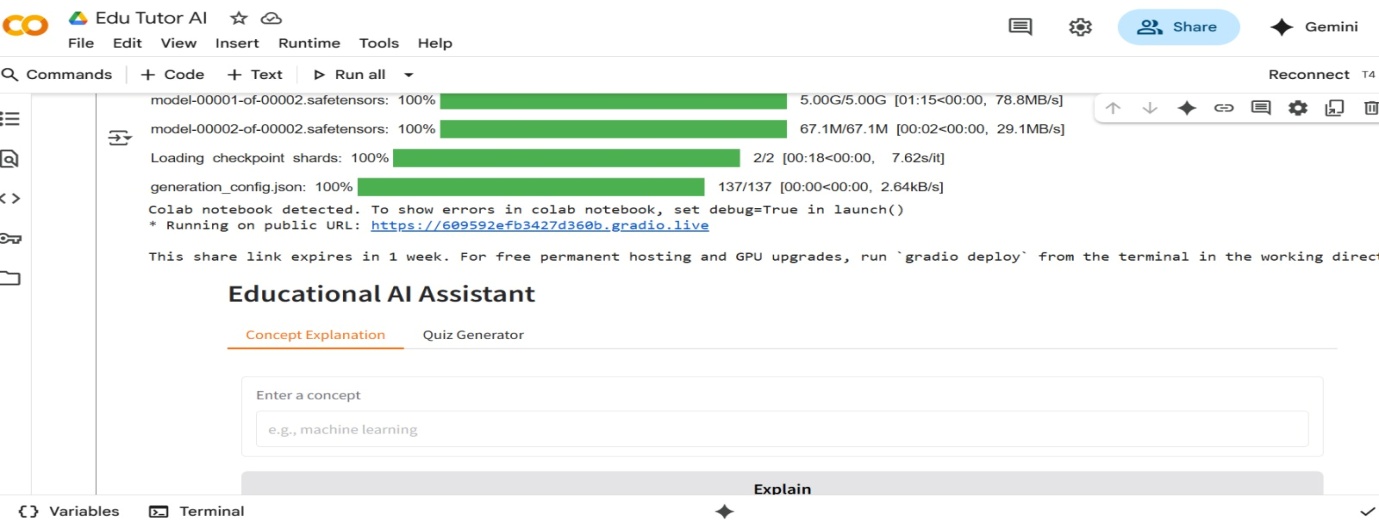
****

**This part of the code defines two main functions:**

* **concept\_explanation()** – generates simple explanations for user-entered topics.
* **quiz\_generator()** – creates quiz questions and answers for the chosen subject.These functions make the Edu TutorAI interactive by providing both learning content and practice quizzes.

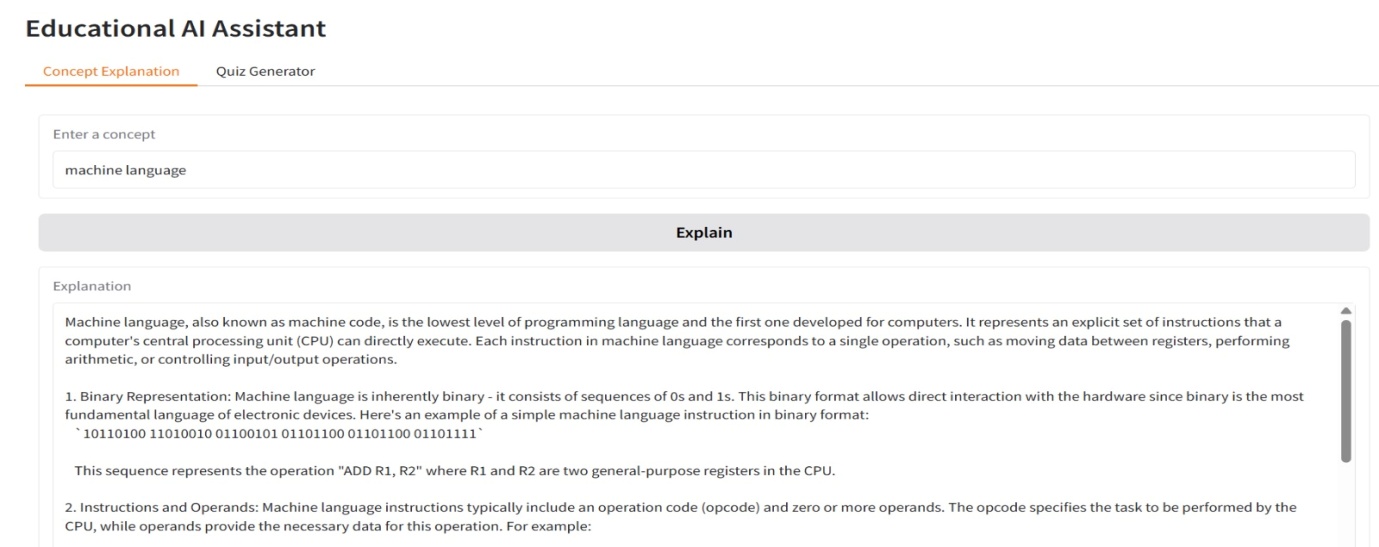
### ****PROGRAM EXECUTION IN GOOGLE COLAB-****





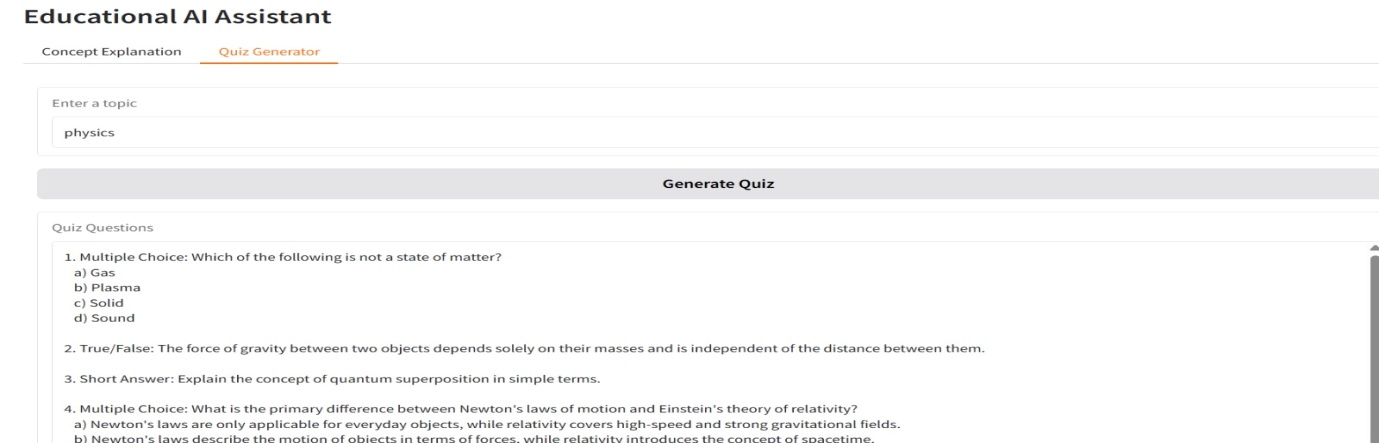
The Edu TutorAI code was executed in **Google Colab**, which provides a cloud-based Python environment. The following screenshot shows the code being run successfully.

Output for Concept Explanation



This screenshot shows the result when a topic, like Machine Learning, is entered. The AI generates a clear and detailed explanation for the concept.

Output for Quiz Generator



The Quiz Generator allows users to enter a topic, and the AI automatically creates **five quiz questions** in different formats, such as multiple choice, true/false, and short answer. At the end, it provides all the correct answers so students can check their performance instantly. This feature helps learners practice and reinforce their understanding of any subject.

### ****CONCLUSION AND FUTURE SCOPE****

* Edu TutorAI demonstrates how AI can be used to make learning interactive and accessible.
* It provides detailed concept explanations and generates quizzes for self-assessment.
* The project was successfully implemented in Python using Hugging Face Transformers, PyTorch, and Gradio.
* In the future, it can be extended with features like:
  + Voice-based tutoring
  + Multilingual support
  + Mobile app integration
  + Adaptive learning based on student performance