

# **EduTutor AI:Personalized Learning with Generative AI And LMS Integration**

## **Project Documentation**

### **1.Introduction**

- **Project Title:**EduTutor AI:Personalized Learning with Generative AI And LMS Integration
- **Team member:**Pravin Kumar S      **NM ID:**498452C2A9DFF748E90408538B4C5F8E
- **Team member:**Sarah Lavina D      **NM ID:**60057127F99C030B3514B5377161DC5D
- **Team member:**Sanjay C      **NM ID:**A8A158EEEC02DBCAABC47A2CB55BBC4E
- **Team member:**Nithyan J      **NM ID :** 58819C30717BF02B1164122DAF957860

### **2.Project Overview**

#### **Project description:**

EduTutor AI is an AI-powered personalized education platform that revolutionizes the way students learn and educators assess progress. It provides dynamic quiz generation, student evaluation, Google Classroom integration, and real-time feedback—all powered by IBM Watsonx and Granite foundation models. Designed with modular architecture, this platform streamlines personalized education and enhances learning outcomes for students across academic levels.

#### **Scenario 1: Personalized Learning Experience**

A student logs into EduTutor AI and synchronizes their courses using their Google Classroom credentials. The platform analyzes course data, generates quizzes on key topics using the Granite LLM, and assesses responses for instant feedback—creating a highly personalized and engaging learning journey.

#### **Scenario 2: Educator Dashboard & Performance Insights**

Educators can log in to view real-time quiz performance of all students. The dashboard highlights quiz history, scores, last topics attempted, and insights fetched from the Pinecone vector database. This empowers teachers to monitor learning progress and personalize their instruction based on data.

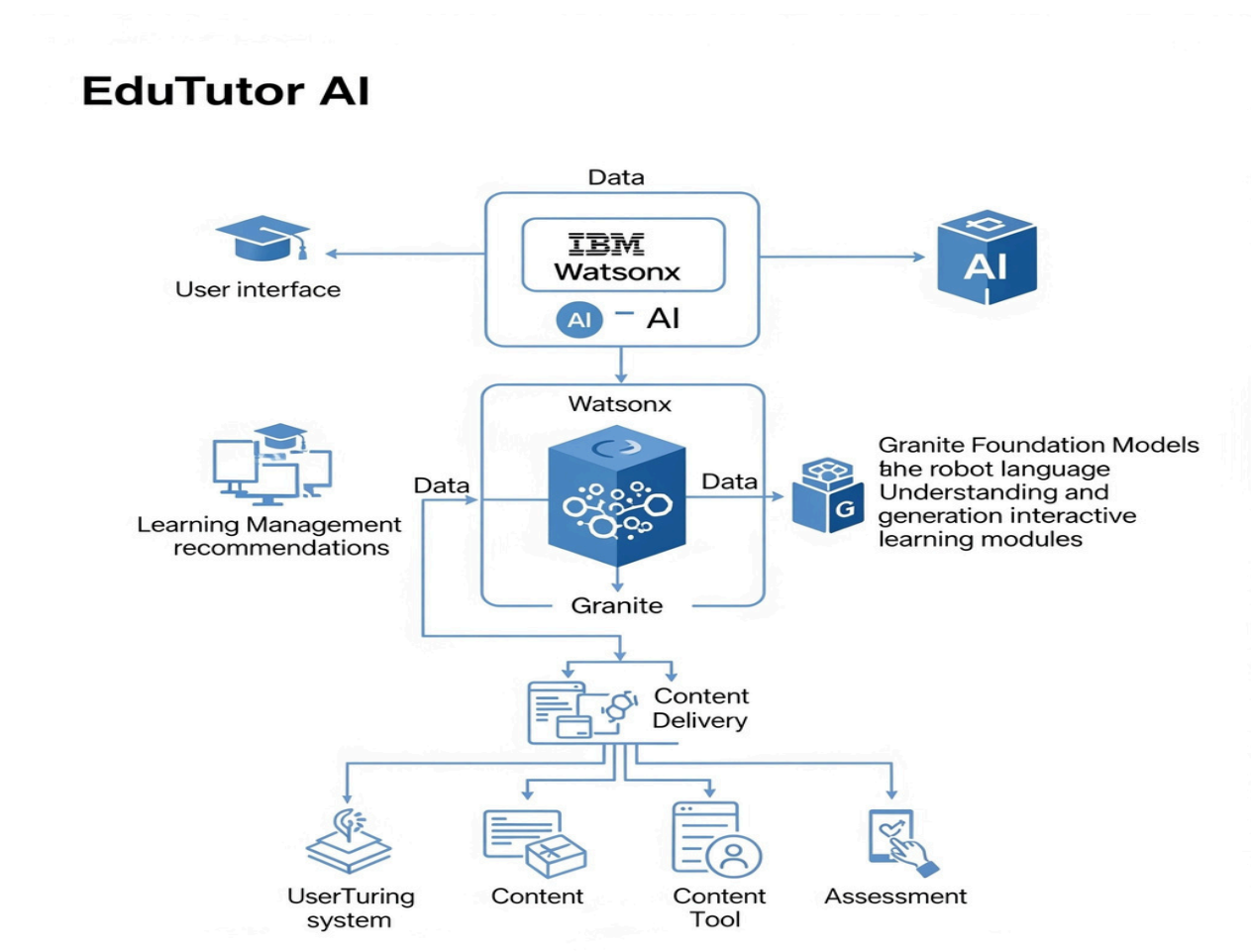
### Scenario 3: Diagnostic Testing and Adaptive Quizzing

Upon registration, students undergo a diagnostic test generated by IBM Watsonx models. Based on the results, the platform adapts quiz difficulty and topic relevance, ensuring students are challenged at the right level.

### Scenario 4: Google Classroom Integration

EduTutor AI syncs courses directly from Google Classroom, allowing seamless access to student data, class names, and subjects. This enables automatic quiz topic generation and helps maintain consistent alignment with the academic curriculum.

## 3. Architecture



## 4. Project flow

### 1. User Input:

Students log in using credentials or Google Classroom and request a quiz by selecting topic and difficulty.

### 2. AI Quiz Generation:

Watsonx+Granite models generate MCQs, stored temporarily without answers in the frontend and with answers in the backend

### 3. User Quiz Submission:

Students submit answers via UI. The backend evaluates the answers, scores the quiz, and stores it in Pinecone DB. But Some errors couldnt built the frontend part of it

### 4. FeedbackLoop:

- a. Educators access this data in the dashboard.

## 5. Requirements Specification

- fastapi
- unicorn
- langchain\_ibm
- pinecone
- streamlit
- google-auth-oauthlib
- google-api-python-client
- python-dotenv

Installation:

```
pip install -r requirements.txt
```

## 6. Initialization of Environment Variables

Create a .env file with :

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

## 7.AI Integration with IBM Watsonx

- Model Setup: Granite model is loaded via `langchain_ibm.WatsonxLLM`.
- Prompt Template: Dynamically generates quiz questions using LangChain's PromptTemplate.
- Quiz Parsing: Watsonx output is parsed into structured JSON for display and evaluation.

## 8.Google Classroom Sync

- Uses `google-auth-oauthlib` and `google-api-python-client`.
- User logs in via Google.

## 9.Pinecone Vector DB Integration

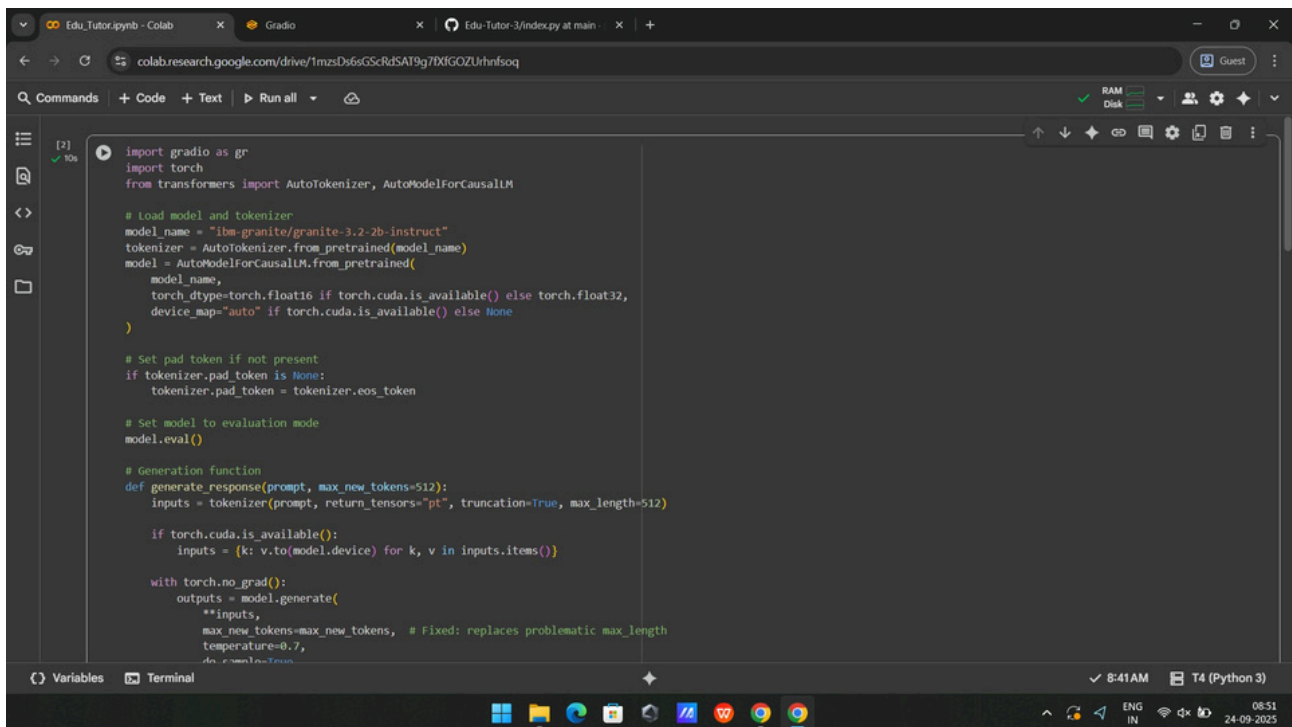
- Stores each user's profile with embeddings.
- Quiz metadata (score, topic, date) is updated after submission.
- Allows educators to fetch student data and analyze progress.

## 10.Streamlit Frontend UI

- Student Panel :
  - Login(Manuel or Google)
  - Dashboard
  - Take Quiz
  - Quiz History
- Educator Panel :
  - Dashboard(all student analytics)

## 11. Screenshots

- Coding



```
[2] ✓ 10s
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# Load model and tokenizer
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
)

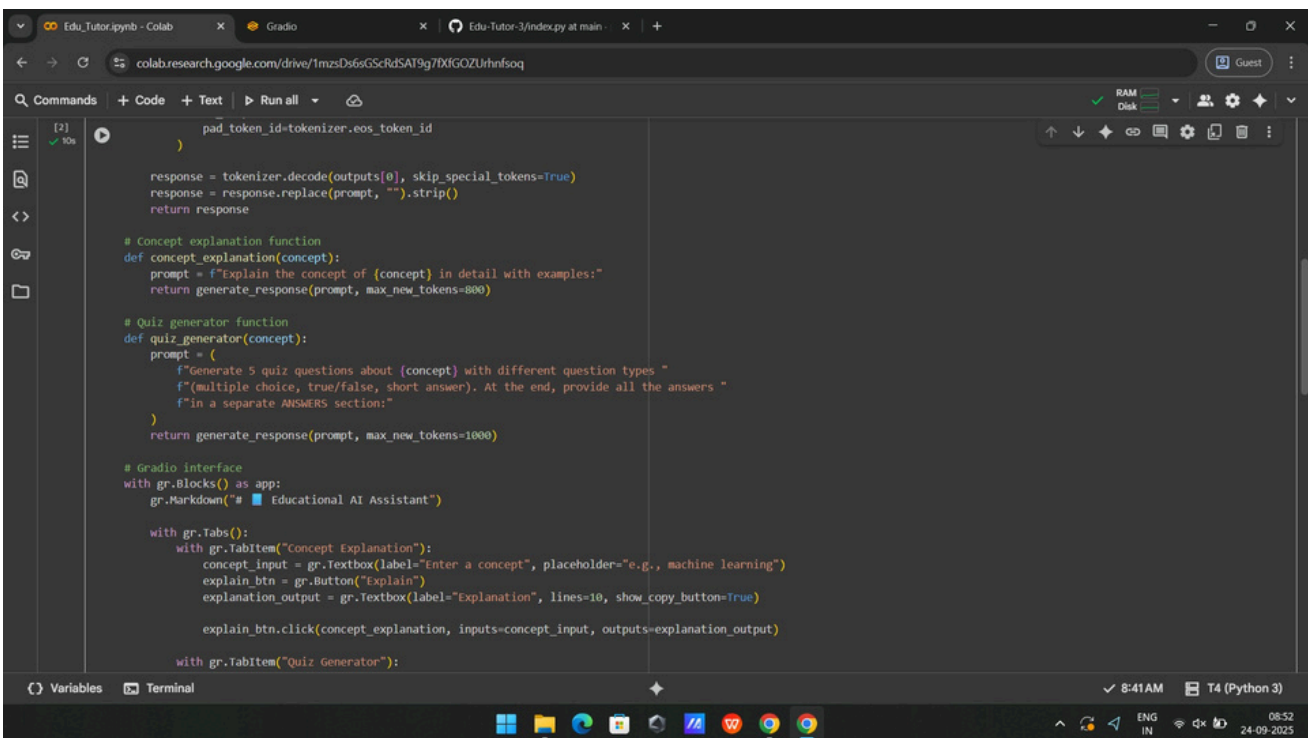
# Set pad token if not present
if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

# Set model to evaluation mode
model.eval()

# Generation function
def generate_response(prompt, max_new_tokens=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_new_tokens=max_new_tokens, # Fixed: replaces problematic 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

# Concept explanation function
def concept_explanation(concept):
    prompt = f"Explain the concept of {concept} in detail with examples:"
    return generate_response(prompt, max_new_tokens=800)

# Quiz generator function
def quiz_generator(concept):
    prompt = (
        f"Generate 5 quiz questions about {concept} with different question types "
        f"(multiple choice, true/false, short answer). At the end, provide all the answers "
        f"in a separate ANSWERS section:"
    )
    return generate_response(prompt, max_new_tokens=1000)

# Gradio interface
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, show_copy_button=True)
            explain_btn.click(concept_explanation, inputs=concept_input, outputs=explanation_output)

        with gr.TabItem("Quiz Generator"):
```

- Output

