

# EduTutor AI: Personalized Learning with Generative AI and LMS Integration

---

Leader: SIVANESAN.M

Team Member: SIVARANJAN.S

Team Member: SRIRAM.P

Team Member: SRIHARISH.K

## 1. Introduction

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.

## 2. Project Overview

**Purpose:** The purpose of EduTutor AI is to create a personalized and adaptive learning environment that helps students improve their skills through tailored quizzes, smart feedback, and progress tracking. It empowers educators with AI-driven insights to evaluate performance and recommend learning paths.

**Features:**

- Dynamic Quiz Generation
- Student Performance Evaluation
- Google Classroom Integration
- AI-Powered Feedback
- Real-time Progress Tracking
- Multi-modal Input Support
- User-Friendly Dashboard Interface

### 3. Architecture

Frontend: Built with React/Streamlit for an interactive and user-friendly UI.

Backend: Implemented with FastAPI to handle quiz generation, evaluation, and feedback APIs.

LLM Integration: IBM Watsonx Granite models for text generation, evaluation, and recommendations.

Database: MySQL for storing user details, results, and learning data.

Integration: Google Classroom API for seamless learning management system connectivity.

### 4. Setup Instructions

Prerequisites:

- Python 3.9 or later
- FastAPI and required libraries
- IBM Watsonx API key
- MySQL server setup

Installation Steps:

1. Clone the repository
2. Install dependencies from requirements.txt
3. Configure database in .env file
4. Run FastAPI backend server
5. Launch frontend dashboard
6. Start interacting with EduTutor AI modules

### 5. Folder Structure

app/ – Backend APIs and business logic

ui/ – Frontend code and dashboards

models/ – Machine learning and LLM integration

database/ – MySQL configurations

docs/ – Documentation and reports

main.py – Entry point to run the backend server

### 6. Running the Application

- Start FastAPI server
- Launch React/Streamlit dashboard
- Navigate through the quiz, performance, and feedback pages
- Upload student data, generate quizzes, and review AI feedback

## 7. API Documentation

Available APIs:

- POST /quiz/generate – Generates quizzes dynamically
- POST /evaluate – Evaluates student answers
- GET /progress – Retrieves student progress reports
- POST /feedback – Provides AI-generated learning feedback
- GET /classroom/sync – Integrates with Google Classroom

## 8. Authentication

EduTutor AI supports:

- JWT-based authentication
- Role-based access control (Admin, Educator, Student)
- Secure API key for third-party integrations

## 9. User Interface

The UI is minimal and interactive, providing:

- Sidebar navigation for modules
- Quiz interface
- Student progress charts
- AI-generated feedback panel
- Report download option

## 10. Testing

Testing Phases:

- Unit Testing (Quiz generation, evaluation logic)
- API Testing (via Swagger, Postman)
- Manual Testing (student flows, quiz attempts)
- Edge Cases (incorrect inputs, invalid logins)
- Integration Testing (Google Classroom sync)

## 12. Known Issues

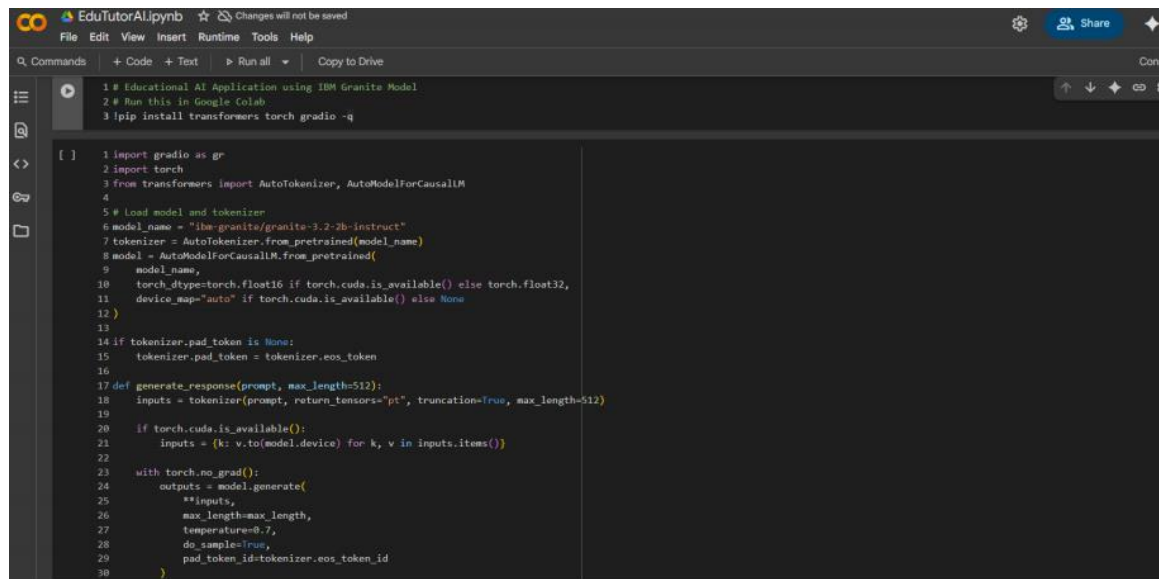
- Limited offline support
- Dependency on Watsonx cloud availability
- Basic UI theme (can be enhanced)

## 13. Future Enhancements

- Add voice-based quiz interaction
- Mobile app version

- Multi-language support
- Advanced analytics dashboards
- Offline learning mode

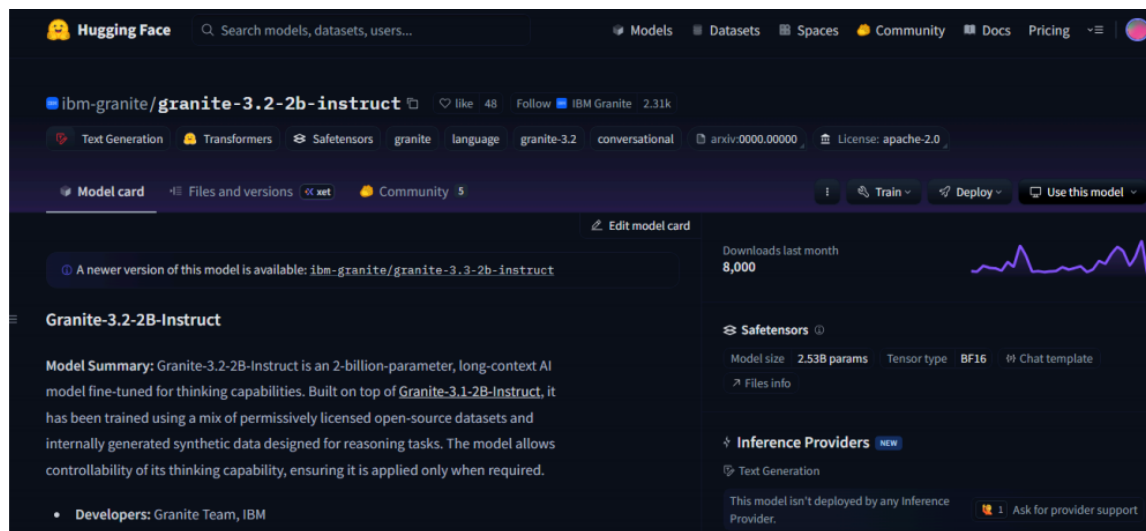
## 14. Screenshots & Outputs



```
1 # Educational AI Application using IBM Granite Model
2 # Run this in Google Colab
3 !pip install transformers torch gradio -q

[ ] 1 import gradio as gr
    2 import torch
    3 from transformers import AutoTokenizer, AutoModelForCausalLM
    4
    5 # Load model and tokenizer
    6 model_name = "ibm-granite/granite-3.2-2b-instruct"
    7 tokenizer = AutoTokenizer.from_pretrained(model_name)
    8 model = AutoModelForCausalLM.from_pretrained(
    9     model_name,
   10     torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
   11     device_map="auto" if torch.cuda.is_available() else None
   12 )
   13
   14 if tokenizer.pad_token is None:
   15     tokenizer.pad_token = tokenizer.eos_token
   16
   17 def generate_response(prompt, max_length=512):
   18     inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)
   19
   20     if torch.cuda.is_available():
   21         inputs = {k: v.to(model.device) for k, v in inputs.items()}
   22
   23     with torch.no_grad():
   24         outputs = model.generate(
   25             **inputs,
   26             max_length=max_length,
   27             temperature=0.7,
   28             do_sample=True,
   29             pad_token_id=tokenizer.eos_token_id
   30         )
```

Figure 1: Google Colab - Model Import and Setup



**Hugging Face** Search models, datasets, users... Models Datasets Spaces Community Docs Pricing

**ibm-granite/granite-3.2-2b-instruct** like 48 Follow IBM Granite 2.31k

Text Generation Transformers Safetensors granite language granite-3.2 conversational arxiv:0000.00000 License: apache-2.0

Model card Files and versions xet Community

**Granite-3.2-2B-Instruct**

**Model Summary:** Granite-3.2-2B-Instruct is an 2-billion-parameter, long-context AI model fine-tuned for thinking capabilities. Built on top of **Granite-3.1-2B-Instruct**, it has been trained using a mix of permissively licensed open-source datasets and internally generated synthetic data designed for reasoning tasks. The model allows controllability of its thinking capability, ensuring it is applied only when required.

- Developers: Granite Team, IBM

Downloads last month: 8,000

**Safetensors**

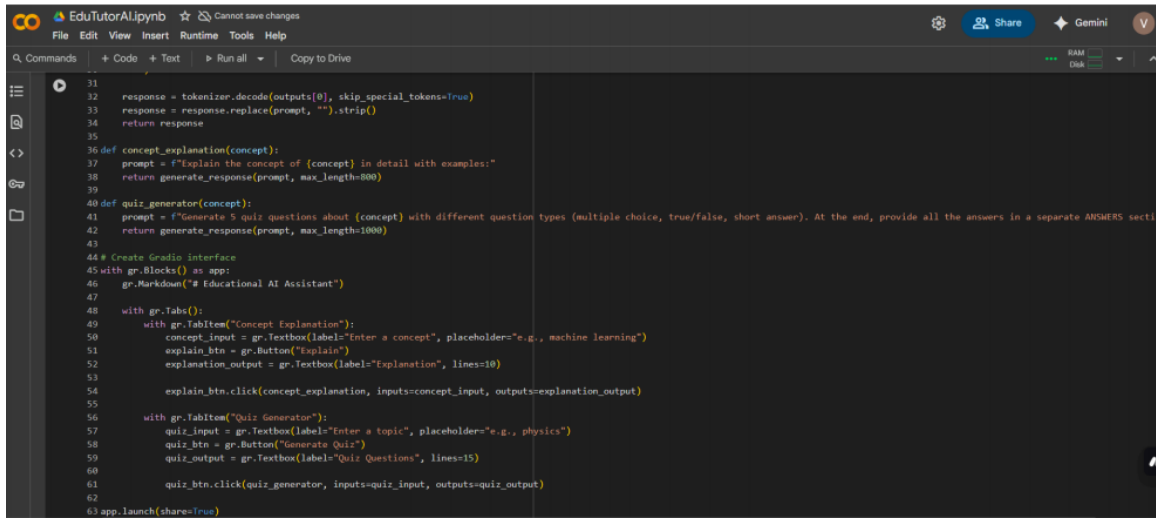
Model size: 2.53B params Tensor type: BF16 Chat template Files info

**Inference Providers**

Text Generation

This model isn't deployed by any Inference Provider. Ask for provider support

Figure 2: Hugging Face - IBM Granite Model Page



```
31
32 response = tokenizer.decode(outputs[0], skip_special_tokens=True)
33 response = response.replace(prompt, "").strip()
34 return response
35
36 def concept_explanation(concept):
37     prompt = f"Explain the concept of {concept} in detail with examples:"
38     return generate_response(prompt, max_length=800)
39
40 def quiz_generator(concept):
41     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."
42     return generate_response(prompt, max_length=1000)
43
44 # Create Gradio interface
45 with gr.Blocks() as app:
46     gr.Markdown("# Educational AI Assistant")
47
48     with gr.Tabs():
49         with gr.TabItem("Concept Explanation"):
50             concept_input = gr.Textbox(label="Enter a concept", placeholder="e.g., machine learning")
51             explain_btn = gr.Button("Explain")
52             explanation_output = gr.Textbox(label="Explanation", lines=10)
53
54             explain_btn.click(concept_explanation, inputs=concept_input, outputs=explanation_output)
55
56         with gr.TabItem("Quiz Generator"):
57             quiz_input = gr.Textbox(label="Enter a topic", placeholder="e.g., physics")
58             quiz_btn = gr.Button("Generate Quiz")
59             quiz_output = gr.Textbox(label="Quiz Questions", lines=15)
60
61             quiz_btn.click(quiz_generator, inputs=quiz_input, outputs=quiz_output)
62
63 app.launch(share=True)
```

Figure 3: Google Colab - Application Code with Gradio UI