
Project Documentation

1. Introduction

Project title: Edu Tutor AI: Personalized Learning With Generative Ai and LMS integration

Team members:

1. PALLAVI.M (Team lead)
2. MENAGA .M
3. MAHALAKSHMI.K
4. PRABHAVATHI.K
5. SUJITHA.V

2. Project Overview

Purpose:

The purpose of the Edututor AI project is to build a personalized intelligent learning assistant that adapts to each learner. Unlike traditional e-learning, it uses Generative AI with LMS integration to provide custom study material, interactive quizzes, real-time doubt solving, and progress tracking for effective learning.

Features:

1. Personalized Learning – Tailors content based on learner's progress and preferences.
2. AI Tutor Chatbot – Provides real-time doubt clarification using Generative AI.
3. Customized Study Material – Generates summaries, notes, and flashcards for easy learning
4. Interactive Quizzes – Auto-generated assessments to test knowledge.
5. Progress Tracking – Monitors learner performance and suggests improvements.
6. LMS Integration – Connects with Learning Management Systems for content and data management.
7. Reports & Analytics – Provides insights to both students and teachers.
8. User-Friendly Interface – Simple, engaging, and accessible platform.

3. Architecture

1. User Layer

Students and teachers access the system via a web or mobile interface.

Provides login, personalized dashboard, and interaction with the AI tutor.

2. Application Layer

Frontend: Built using HTML, CSS, JavaScript (React/Angular optional).

Backend: Handles business logic using Python (Flask/Django).

AI Tutor Module: Powered by IBM Watson / Generative AI for Q&A, summaries, and recommendations.

3. Database Layer

Stores user profiles, course content, quiz results, and performance data.

Supports analytics for personalized feedback.

4. Integration Layer

Connects with LMS (e.g., Moodle or custom LMS).

Syncs course materials, progress reports, and assessments.

5. Cloud Deployment

Hosted on IBM Cloud for scalability and security.

API integration with AI and LMS services.

4. Setup Instructions

Prerequisites: -

- Python installed on your system
- IBM Watsonx API key
- Pinecone API key
- Google OAuth credentials
- FastAPI and Streamlit installed

Installation:

1. Clone the repository: git clone <https://github.com/yourusername/edututor-ai.git>

2. Install dependencies: pip install -r requirements.txt

3. Configure environment variables in a .env file:

- WATSONX_API_KEY
- WATSONX_ENDPOINT
- WATSONX_PROJECT_ID
- PINECONE_API_KEY
- PINECONE_INDEX_NAME

5. Folder Structure

edututor/

```
|— backend/
| |— app/
| | |— __init__.py
| | |— models/
| | |— routes/
| | |— services/
| | |— utils/
| | |— config/
| |
| |— manage.py
| |— requirements.txt
| |— tests/
|
|— frontend/
| |— public/
| |— src/
```

```
| | |— components/
| | |— pages/
| | |— hooks/
| | |— services/
| | |— styles/
| | └─ utils/
| └─ package.json
|   └─ tsconfig.json
|
|— integrations/
| |— moodle/
| |— canvas/
| |— blackboard/
|   └─ custom/
|
|— docker/
| |— Dockerfile.backend
| |— Dockerfile.frontend
|   └─ docker-compose.yml
|
|— docs/
| |— setup.md
| |— api_reference.md
|   └─ user_guide.md
|
|— .env.example      environment variables
|— .gitignore
```

6. Running the Application

- Activate the virtual environment using the appropriate command-for-your OS.
- Start the backend server with the runserver command.
- Start the frontend by running npm start inside the frontend folder.
- Alternatively, use Docker with docker-compose up --build to launch everything together.
- Open your browser to access the backend API and the frontend dashboard

7. Authentication

Endpoints to log in, log out, and manage user tokens.

8. User Interface

- Dashboard – Central hub showing enrolled courses, recent activity, notifications, and progress overview.
- Course Module – Displays course content, lessons, quizzes, and assignments pulled from the LMS.
- AI Tutor Panel – Interactive chat or assistant space where learners can ask questions and receive personalized explanations or generated study materials.
- Analytics Section – Visual reports for students (progress, strengths, weak areas) and teachers (class performance, engagement levels).
- Admin & Settings – Management area for user roles, AI configuration, LMS integration, and system preferences.

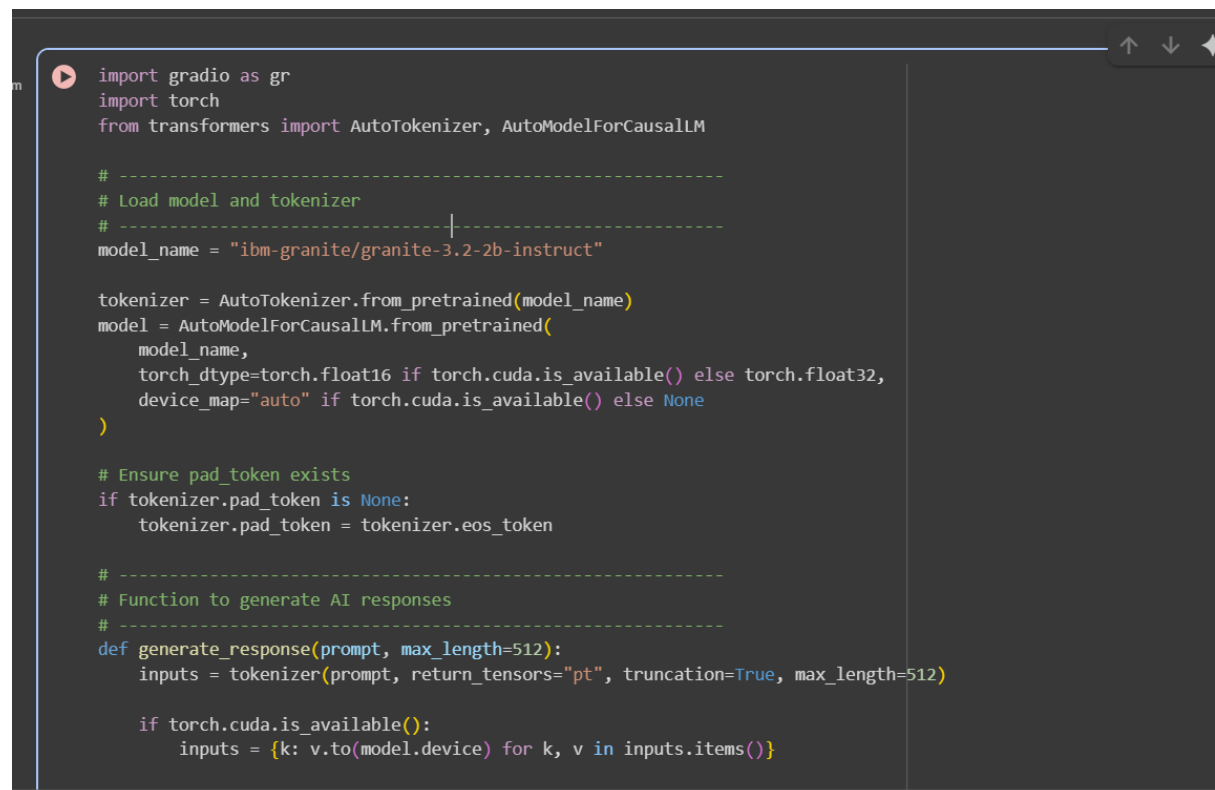
9. Testing

- Unit Testing – Verify individual components like authentication, AI services, and LMS connectors.
- Integration Testing – Test interactions between backend, frontend, and LMS APIs.

- Functional Testing – Ensure features like login, course enrollment, AI tutor responses, and analytics work correctly.
- User Acceptance Testing (UAT) – Validate the system with teachers, students, and admins for real-world workflows.

Performance & Security Testing – Check system load handling, response times, data protection, and API security.

10. Screenshots



```
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
)

# Ensure pad_token exists
if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

# -----
# Function to generate AI responses
# -----
def generate_response(prompt, max_length=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_length=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
# -----
def concept_explanation(concept):
    prompt = f"Explain the concept of {concept} in detail with examples:"
    return generate_response(prompt, max_length=800)

# -----
# Quiz Generator
# -----
def quiz_generator(concept):
    prompt = f"Generate 5 quiz questions about {concept} with different question types (multiple choice, true/false, short answer). At the"
    return generate_response(prompt, max_length=1000)

# -----
# Gradio Interface
# -----

```

```

with gr.Blocks() as app:
    gr.Markdown("### 🧠 Educational AI Assistant (IBM Granite)")

    with gr.Tabs():
        # Tab 1: Concept Explanation
        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)
            explain_btn.click(concept_explanation, inputs=concept_input, outputs=explanation_output)

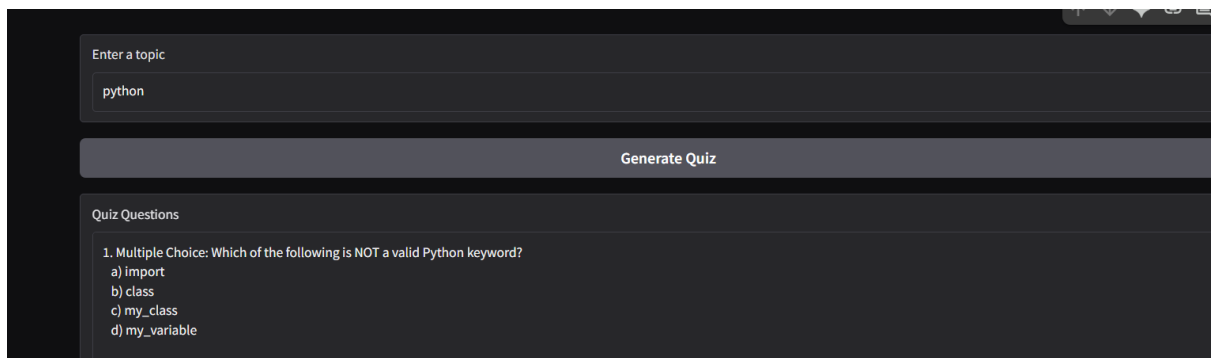
        # Tab 2: Quiz Generator
        with gr.TabItem("Quiz Generator"):
            quiz_input = gr.Textbox(label="Enter a concept", placeholder="e.g., photosynthesis, gravity, physics")
            quiz_btn = gr.Button("Generate Quiz")
            quiz_output = gr.Textbox(label="Quiz Questions", lines=10)
            quiz_btn.click(quiz_generator, inputs=quiz_input, outputs=quiz_output)

    # -----
    # Launch the app
    # -----
    if __name__ == "__main__":
        app.launch()

```

/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning: The secret 'HF_TOKEN' does not exist in your Colab secrets. To authenticate with the Hugging Face Hub, create a token in your settings tab (<https://huggingface.co/settings/tokens>), set it as secret in your environment. You will be able to reuse this secret in all of your notebooks. Please note that authentication is recommended but still optional to access public models or datasets.

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret 'HF_TOKEN' does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
warnings.warn(
tokenizer_config.json: 8.88k/? [00:00<00:00, 239kB/s]
vocab.json: 777k/? [00:00<00:00, 6.31MB/s]
merges.txt: 442k/? [00:00<00:00, 11.0MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 43.1MB/s]
added_tokens.json: 100% [87.0/87.0] [00:00<00:00, 2.75kB/s]
special_tokens_map.json: 100% [701/701] [00:00<00:00, 22.1kB/s]
config.json: 100% [786/786] [00:00<00:00, 58.8kB/s]
"torch_dtype" is deprecated! Use "dtype" instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 2.16MB/s]
Fetching 2 files: 100% [2/2] [01:57<00:00, 117.00s/it]
model-00002-of-00002.safetensors: 100% [67.1M/67.1M] [00:03<00:00, 19.5MB/s]
model-00001-of-00002.safetensors: 100% [5.00G/5.00G] [01:56<00:00, 43.7MB/s]
Loading checkpoint shards: 100% [2/2] [00:19<00:00, 8.21s/it]
generation_config.json: 100% [What can I help you build?]
colab notebook detected. To show errors in colab notebook, set debug=True in launch()
```



12. Known Issues

- LMS API Limitations – Some LMS platforms (like Moodle, Canvas, or Blackboard) may restrict API calls, causing delays or incomplete data sync.
- AI Response Variability – Generative AI may sometimes give overly complex, irrelevant, or inconsistent answers.
- Performance Bottlenecks – High traffic or simultaneous AI requests can slow down response times if caching and scaling aren't optimized.
- Integration Conflicts – Version mismatches between backend, frontend, or third-party services (AI SDKs, LMS APIs) may cause errors.

- User Experience Gaps – Students may find navigation or AI tutor interaction confusing if UI personalization isn't tuned properly.

13. Future Enhancements

- Adaptive Learning Paths – Dynamically adjust course flow based on student performance, behavior, and AI recommendations.
- Multimodal AI Tutor – Support text, voice, and even image-based queries so learners can interact in different formats.
- Gamification Features – Add badges, leaderboards, and rewards to increase engagement and motivation.
- Smart Assessment Generator – Use AI to auto-create quizzes, assignments, and practice tests tailored to individual learners.
- Predictive Analytics – Implement AI-driven insights to forecast student dropouts, weak areas, and recommend interventions early