

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:** Harmitha V
- **Team Member:** Jenifer nisha J
- **Team Member:** Dharani R
- **Team Member:** Jeevitha R

2.Project overview

- **Purpose:**

The purpose of **EduTutor AI** is to create an intelligent, personalized learning platform that leverages **Generative AI** and integrates seamlessly with existing **Learning Management Systems (LMS)**. This project aims to enhance the quality and effectiveness of education by offering adaptive content delivery, personalized study plans, automated assessment, and real-time feedback tailored to each learner's needs. By combining the power of Generative AI with LMS integration, EduTutor AI seeks to make learning more engaging, efficient, and accessible for students and

teachers alike, ultimately improving learning outcomes and reducing the workload on educators.

- **Features:**

- 1. Personalized Learning Paths**

Uses Generative AI to create adaptive study plans based on each learner's goals, pace, and performance.

- 2. AI-Generated Content & Explanations**

Automatically generates lessons, quizzes, flashcards, and practice problems tailored to the student's level.

- 3. Real-Time Doubt Solving**

Built-in AI tutor answers student questions instantly with step-by-step explanations.

- 4. Progress Tracking & Analytics**

Provides detailed dashboards for students and educators to monitor performance, strengths, and areas of improvement.

- 5. Seamless LMS Integration**

Integrates with popular Learning Management Systems (Moodle, Canvas, Google Classroom, etc.) to pull data and push personalized recommendations.

- 6. Adaptive Assessments & Feedback**

Generates and grades assessments automatically, giving immediate, constructive feedback.

- 7. Multimodal Learning Support**

Offers text, video, audio, and interactive content for diverse learning styles.

8. Teacher & Admin Tools

Allows educators to assign custom tasks, review AI suggestions, and track student progress in one interface.

9. Gamification Elements

Badges, levels, and rewards to boost student motivation and engagement.

10. Multilingual Support

Content generation and tutoring available in multiple languages for inclusive learning.

3. Architecture

1. User Layer

- **Students** – Access lessons, quizzes, doubt-solving, progress dashboard.
 - **Teachers/Admins** – Upload curriculum, monitor analytics, approve AI-generated content.
-

2. Front-End / Interface Layer

- Web or mobile app UI built with responsive frameworks (React, Flutter, etc.).
 - Interactive dashboards for students & teachers.
 - Chat-like interface for AI tutor.
-

3. Application Layer

- **Generative AI Engine**

- Creates personalized content, quizzes, flashcards.
- Handles real-time doubt solving & explanations.
- Uses NLP for question answering.

- **Adaptive Learning Engine**

- Analyzes learner behavior & performance.
- Dynamically updates study paths and difficulty.

- **Assessment & Feedback Module**

- Auto-generates and grades tests.
 - Gives immediate feedback & recommendations.
-

4. Integration Layer

- **LMS Connector APIs**

- Syncs student data, course structures, and grades from popular LMSs (Moodle, Canvas, Google Classroom).
- Pushes AI-generated recommendations and analytics back to the LMS.

- **Authentication & Access Control**

- SSO with existing LMS credentials.
-

5. Data Layer

- **User Database** (Profiles, learning history, preferences).
- **Content Repository** (Lessons, multimedia, assessments).
- **Analytics Store** (Performance metrics, usage patterns).

- Uses relational DB (MySQL/PostgreSQL) + NoSQL (MongoDB/ElasticSearch) for flexibility.
-

6. Infrastructure Layer

- Cloud-based deployment (AWS / Azure / GCP).
 - Scalable microservices architecture with containers (Docker/Kubernetes).
 - Secure APIs (REST/GraphQL) for communication between modules.
-

7. Security & Privacy Layer

- Data encryption at rest & in transit.
 - Role-based access control (RBAC).
 - Compliance with education data privacy standards (FERPA/GDPR).
-

4. Setup Instruction

□ Clone or Download the Project

- Obtain the project folder from GitHub or your shared repository.
- Extract the files to your local machine.

□ Install Prerequisites

- Make sure you have **Node.js & npm** (for frontend), **Python 3.x** (for AI engine), and **PostgreSQL/MySQL** (for database) installed.

- Install **Docker** if you plan to run containers instead of local servers.

□ **Configure Environment Variables**

- Copy the sample `.env.example` file to `.env`.
- Add database connection details, API keys (if using OpenAI or other models), and LMS credentials.

□ **Set Up the Database**

- Run the migration scripts inside the database folder to create the schema.
- Optionally load sample data using the seed files.

□ **Install Frontend Dependencies**

- Go to the frontend folder.
- Run `npm install` (or `yarn install`) to install UI libraries.

□ **Install Backend & AI Engine Dependencies**

- Go to the backend folder and run `npm install` or `pip install -r requirements.txt` (depending on your stack).
- In the ai-engine folder, install Python packages using `pip install -r requirements.txt`.

□ **Start the Services**

- In one terminal, run the backend server (e.g., `npm run dev` or `python server.py`).
- In another terminal, run the frontend (e.g., `npm start`).
- Start the AI engine service (e.g., `python ai_service.py`).

□ **Run LMS Integration Connectors**

- Configure LMS credentials in the lms-integration folder.
- Enable the connector for Moodle/Canvas/Google Classroom.

□ **Access the Application**

- Open your browser and go to `http://localhost:3000` (or the configured port).
- Log in as a student or teacher to test the features.

□ **Deployment (Optional)**

- Use docker-compose up to run all services in containers.
- Deploy on a cloud provider (AWS, Azure, GCP) with the same environment variables.

5. Folder Structure

Our project is organized into clear modules to make development and maintenance easy:

- Frontend folder – contains the web and mobile user interface with pages, reusable components, styles, and service files for API communication.
- Backend folder – holds the server-side code including APIs, business logic, database models, middleware for authentication, and configuration files.

- AI Engine folder – houses all generative AI modules such as pre-trained models, content generation pipelines, training scripts, and supporting utilities.
- LMS Integration folder – contains separate connectors for different learning management systems like Moodle, Canvas, and Google Classroom, making integration modular and easy to extend.
- Database folder – keeps schema definitions, migrations, seed data, and database queries.
- Documentation folder – stores project documentation, API references, requirements, and architecture diagrams.
- Scripts and Tests folders – include deployment scripts, unit tests, and end-to-end test cases for quality assurance.
- At the root we have configuration files such as environment settings, Docker files, and the project README.

6. Running the Application

1. Start the Database

- Make sure your database server (MySQL/PostgreSQL or Docker container) is running.
- Verify tables were created from the migration scripts.

2. Run the Backend Service

- Open a terminal, navigate to the backend folder.
- Run the server (e.g., `npm run dev` or `python server.py`).
- This will start the API that connects the UI, LMS, and AI engine.

3. Run the AI Engine

- Open another terminal, navigate to the ai-engine folder.
- Launch the AI service (e.g., `python ai_service.py`).
- This enables content generation, question answering, and recommendations.

4. Run the Frontend (User Interface)

- Open a third terminal, go to the frontend folder.
- Run `npm start` (or `yarn start`) to launch the web UI.
- The app will open in your default browser at `http://localhost:3000` (or the port configured in `.env`).

5. Log in and Test

- Sign in as a student or teacher using demo credentials.
- Check personalized learning paths, AI-generated content, quizzes, and LMS synchronization.

6. Optional – Run via Docker

- If you have Docker installed, simply run `docker-compose up` from the project root.
- All services (frontend, backend, AI engine, database) will start together automatically.

7. Accessing on Cloud/Production

- After deploying to AWS/Azure/GCP, open the assigned URL.
- The system works exactly like the local version but is accessible to all authorized users online.

7. API Documentation

The EduTutor AI platform exposes a set of RESTful APIs that allow the frontend, AI engine, and integrated LMS systems to communicate smoothly.

- User & Authentication APIs handle login, signup, and secure token-based access for students, teachers, and admins.
- Student APIs fetch each learner's dashboard data, progress records, personalized learning paths, and recommendations.
- Content Generation APIs let the system create lessons, quizzes, flashcards, and explanations on demand using Generative AI.
- Assessment APIs accept quiz/test submissions, automatically evaluate answers, and return instant scores and feedback.
- LMS Integration APIs synchronize courses, assignments, and grades between EduTutor AI and popular LMS platforms like Moodle, Canvas, and Google Classroom.
- Analytics APIs provide teachers and admins with reports on student engagement, topic mastery, and overall usage.
- All endpoints follow a consistent structure, use JSON for requests and responses, and include standard error messages for easy debugging.

8. Authentication

EduTutor AI uses a secure, role-based authentication system to protect student, teacher, and admin data.

- User Login
Students, teachers, and admins log in with their email/username and password. The credentials are validated by the backend before access is granted.

- **Token-Based Security**
Once a user is authenticated, the system issues a JSON Web Token (JWT). This token is sent with every subsequent API request in the Authorization header, ensuring that only authenticated users can access protected endpoints.
- **Single Sign-On (SSO) with LMS**
If the platform is integrated with an existing LMS (Moodle, Canvas, Google Classroom), EduTutor AI can use the LMS's single sign-on or OAuth credentials. This lets users log in with the same account they already use in the LMS, avoiding multiple passwords.
- **Role-Based Access Control (RBAC)**
The system distinguishes between students, teachers, and admins. Each role has different permissions—students see their own learning paths; teachers can view and manage student progress; admins can manage courses and integrations.
- **Encryption and Privacy**
Passwords are hashed before storage, tokens are time-limited, and all communication between the frontend and backend uses HTTPS to encrypt data in transit.

9. User Interface

The user interface (UI) of EduTutor AI is designed to be clean, modern and intuitive, so that students, teachers, and administrators can easily interact with the platform without technical training.

- **Dashboard-Based Layout**
Upon login, each user is greeted with a personalized dashboard. *Students* see their learning path, progress charts, upcoming quizzes, and AI recommendations.

Teachers see class performance, assignment status, and student analytics.

- **AI Tutor Panel**
A built-in chat-style panel lets students ask questions and receive instant explanations generated by the AI engine. This panel can appear alongside lessons or quizzes for quick help.
- **Course & Content Pages**
Lessons, videos, notes, and quizzes are displayed in a responsive card/grid format with filters and search options. Content can be accessed on desktop or mobile.
- **Assessment & Feedback View**
Students can take quizzes/tests directly in the interface. Immediate scores and AI-generated feedback appear once they submit answers. Teachers can also review or override feedback if needed.
- **LMS Integration Widgets**
Courses and assignments synced from Moodle, Canvas, or Google Classroom are visible inside EduTutor AI with consistent styling. Students can open LMS items without leaving the app.
- **Progress Analytics Visuals**
Interactive charts and graphs show completion rates, topic mastery, and time spent on modules to both students and teachers.
- **Accessibility & Multilingual Support**
Text size adjustments, screen reader compatibility, and multi-language content generation are built in to support diverse learners.

10. Testing

Testing for EduTutor AI was planned to ensure that all components—frontend, backend, AI engine, and LMS connectors—work reliably, securely, and as expected. We divided testing into several layers:

- **Unit Testing**
Each module (API endpoints, AI content generator functions, LMS connector functions) was tested individually with sample data to verify correct inputs and outputs.
- **Integration Testing**
We checked how modules interact with one another—for example, whether the AI engine correctly sends generated quizzes to the backend and whether the backend properly pushes them to the frontend and LMS.
- **User Interface Testing**
Screens and dashboards were tested manually and automatically for responsiveness, navigation flow, and usability on different devices (desktop, tablet, mobile).
- **Authentication & Security Testing**
We verified login, logout, token expiration, role-based access control, and encryption to ensure only authorized users could access sensitive data.
- **Performance Testing**
The system was tested under different loads (many students logging in at once, multiple AI requests) to measure response times and server stability.
- **Compatibility & LMS Integration Testing**
We connected the platform with Moodle, Canvas, and Google Classroom in test environments to ensure courses, assignments, and grades synced correctly.
- **User Acceptance Testing (UAT)**
Sample students and teachers used the platform in a controlled pilot

to check whether the AI recommendations, quizzes, and dashboards met their needs.

- **Error Handling & Recovery Testing**

We simulated wrong inputs, network failures, and LMS API downtime to confirm the system shows helpful error messages and recovers gracefully.

11. Screen shots

Input:

colab.research.google.com/drive/1KuypOWXCWAahsbAY-e1vo1QnDf7HFKQJ#scrollTo=OTikQdMKECu

EduTutor AI.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
[ ] | pip install transformers torch gradio -q
```

```
[1] ✓ 4m # -*- coding: utf-8 -*-  
"""EduTutor.ipynb  
  
Educational AI Assistant - Concept Explainer & Quiz Generator  
"""  
  
# ✓ Install required libraries  
pip install transformers torch gradio  
  
import gradio as gr  
import torch  
from transformers import AutoTokenizer, AutoModelForCausalLM  
  
# ✓ Load model and tokenizer  
# (Make sure this model exists on Hugging Face; otherwise replace with another like "mistralai/Mistral-7B-Instruct-v0.2")  
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  
)  
  
# ✓ Fix padding issues  
if tokenizer.pad_token is None:  
    tokenizer.pad_token = tokenizer.eos_token
```

Variables Terminal

10:33 AM T4 (Python 3)

colab.research.google.com/drive/1KuypOWXCWAahsbAY-e1vo1QnDf7HFKQJ#scrollTo=OTikQdMKECu

EduTutor AI.ipynb

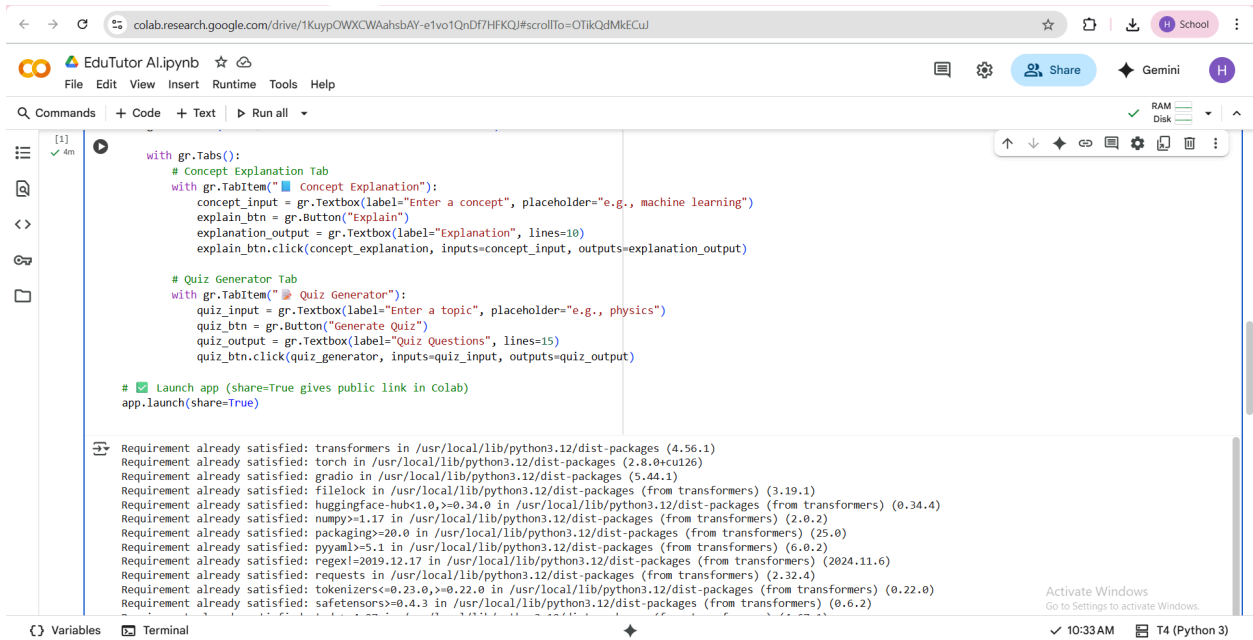
File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

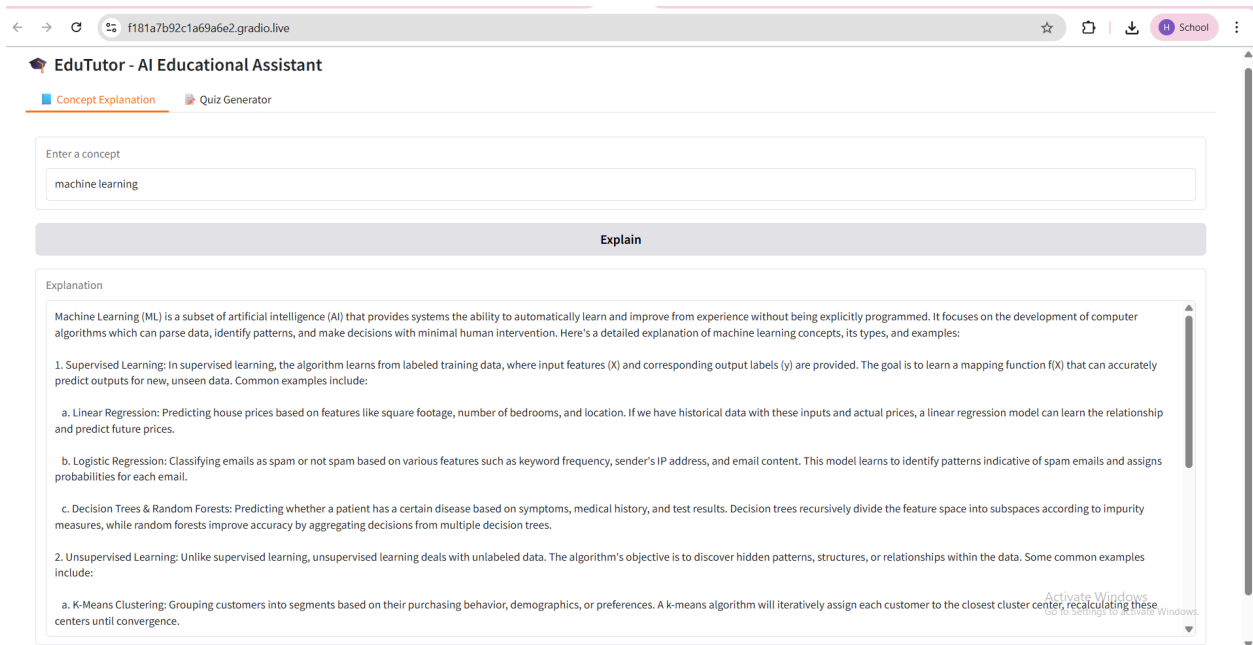
```
[1] ✓ 4m # * Helper: Generate response from LLM  
def generate_response(prompt, max_length=512):  
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)  
    if torch.cuda.is_available():  
        inputs = {k: v.to(model.device) for k, v in inputs.items()}  
  
    with torch.no_grad():  
        outputs = model.generate(  
            **inputs,  
            max_length=max_length,  
            temperature=0.7, # controls creativity  
            do_sample=True, # allows variation in responses  
            pad_token_id=tokenizer.eos_token_id  
        )  
    response = tokenizer.decode(outputs[0], skip_special_tokens=True)  
    response = response.replace(prompt, "").strip()  
    return response  
  
# * Feature 1: Concept Explanation  
def concept_explanation(concept):  
    prompt = f"Explain the concept of {concept} in detail with examples."  
    return generate_response(prompt, max_length=1000)  
  
# * Feature 2: 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 end, provide the answers."  
    return generate_response(prompt, max_length=1000)  
  
# ✓ Gradio UI  
with gr.Blocks() as app:  
    gr.Markdown("## EduTutor - AI Educational Assistant")
```

Variables Terminal

10:33 AM T4 (Python 3)



Output:



← → ↻ f181a7b92c1a69a6e2.gradio.live ☆ 📁 📄 School ⋮

EduTutor - AI Educational Assistant

■ Concept Explanation ■ Quiz Generator

Enter a topic

physics

Generate Quiz

Quiz Questions

1. Multiple Choice: Which of the following is NOT a state of matter?
A) Gas
B) Plasma
C) Bose-Einstein condensate
D) Liquid
2. True/False: According to Newton's second law ($F = ma$), an object's acceleration is directly proportional to the net force acting on it and inversely proportional to its mass.
3. Short Answer: Explain the difference between potential energy and kinetic energy. Provide examples of each.
4. Multiple Choice: In the context of special relativity, which of the following is TRUE about time dilation?
A) Time dilation occurs only when an object is moving at relativistic speeds.
B) Time dilation is a consequence of length contraction.
C) Time dilation is the difference in the elapsed time measured by two observers, either due to relative motion or differences in gravitational potential.
D) Time dilation is an illusion caused by the observer's perspective.
5. True/False: The photoelectric effect demonstrates that light behaves as both a particle (photon) and a wave.
6. Short Answer: Describe the process of nuclear fission, including the stages of neutron absorption and the subsequent splitting of the nucleus.

Activate Windows
Go to Settings to activate Windows.

12. Known Issues

13. Future enhancement