

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

---

(Project Documentation)

## Introduction :

- Project Title: EduTutor AI Personalized Learning with Generative AI and LMS Integration
- Team ID : NM2025TMID02705
- Team Leader : POOJASHREE.M (Voice over & Creator)
- Team Member : PAULRAJ.B(Documentation)
- Team Member : YUVARAJ.P(Video Analyser)
- Team Member : VARUN PRASAD.P (Uploader)
- Team Member : YUVAPRATHAP.N (Editor)

## Project Overview :

EduTutor AI: Personalized Learning with Generative AI and LMS Integration is an intelligent tutoring system designed to transform the way students learn and teachers teach. The project leverages Generative AI models and integrates them with Learning Management Systems (LMS) to create a personalized, adaptive, and efficient digital learning environment.

### Purpose

The purpose of EduTutor AI is to bridge the gap between traditional learning and modern technology by providing a platform that can:

- Personalize the learning journey for each student.
- Reduce teacher workload through automation.
- Enhance engagement, accessibility, and learning outcomes.

## Key Features

- Personalized Learning Pathways – AI tailors content and quizzes based on student performance.
- Content Generation – Automated creation of study notes, practice questions, and summaries.
- Automated Assessments – Instant quiz generation and grading.
- Student Progress Tracking – Realtime dashboards for performance monitoring.
- LMS Integration – Compatible with platforms like Moodle and Google Classroom.
- Teacher Assistance – AI helps design lesson plans and provide teaching recommendations.

## Benefits

- For Students: Adaptive, engaging, and supportive learning.
- For Teachers: Reduced workload, realtime analytics, and improved teaching efficiency.
- For Institutions: Scalable, datadriven, and futureready education system.

## Future Scope

EduTutor AI will evolve to include:

- Multilingual support for diverse learners.
- AI-driven career guidance tools.
- Integration with AR/VR for immersive classrooms.
- Smart analytics powered by student behavior tracking.

## System Architecture :

The architecture of **EduTutor AI: Personalized Learning with Generative AI and LMS Integration** is designed to ensure scalability, personalization, and seamless integration with existing educational platforms. It follows a **modular, layered approach** consisting of five main components:

### 1. Frontend Layer (User Interface)

- **Technology:** Streamlit / React / Gradio
- **Users:** Students, Teachers, and Administrators
- **Functions:**

- Student dashboard (personalized content, quizzes, progress tracking)
- Teacher dashboard (lesson plans, grading, analytics)
- Chat-based tutoring system
- File uploads (assignments, notes)
- Real-time visualization of learning progress

## 2.Backend Layer (Application Logic)

- **Technology:** FastAPI / Node.js
- **Functions:**
  - Handles all API requests between frontend and AI modules
  - Manages authentication and user sessions
  - Processes assessments, grading, and data storage
  - Controls LMS synchronization through APIs

## 3.AI Integration Layer (Generative AI Engine)

- **Technology:** IBM Granite LLMs (via Hugging Face Transformers)
- **Functions:**
  - Generates explanations, quizzes, and study materials
  - Summarizes uploaded documents into easy notes
  - Provides adaptive recommendations for students
  - Enhances teacher support with automated content creation

## 4.Database & Storage Layer

- **Technology:** SQL / NoSQL (MongoDB / PostgreSQL)
- **Functions:**
  - Stores user profiles, learning progress, and grades
  - Maintains generated quizzes, notes, and reports
  - Tracks analytics for students and teachers
  - Ensures scalability and secure data storage

## 5.LMS Integration Layer

- **Technology:** Moodle / Google Classroom APIs / LTI Standard
- **Functions:**
  - Syncs student grades and assignments with LMS
  - Imports course content from existing platforms
  - Provides a two-way connection between EduTutor AI and institutional systems

## Optional Components

- **Vector Database (Pinecone/FAISS):** For semantic search over documents.
- **ML Models (Scikit-learn):** For forecasting and anomaly detection (student performance trends).

- **Authentication & Security:** Token-based authentication (JWT/OAuth2) and role-based access.

## Data Flow

1. **Student/Teacher Interaction** – User sends a query or request via the frontend.
2. **Backend Processing** – Request goes through backend APIs for routing.
3. **AI Response Generation** – AI model generates answers, quizzes, or recommendations.
4. **Database Storage** – Results and user progress are stored securely.
5. **LMS Sync** – Updated grades and assignments sync with LMS platforms.
6. **Feedback Loop** – Student and teacher feedback helps improve system responses.

## Setup Instructions :

### Prerequisites

Before setting up EduTutor AI, ensure the following software and accounts are available:

- **Python 3.9 or later** – For backend and AI modules
- **Node.js & npm (optional)** – If React frontend is used
- **Google Colab or local GPU (T4/V100 preferred)** – For running AI models
- **Hugging Face account** – To access IBM Granite models
- **Git & GitHub account** – For version control and project submission
- **Virtual environment tool (venv or conda)** – For dependency isolation
- **API Keys / Access Tokens**
  - Hugging Face token for Granite models
  - LMS API access (Moodle/Google Classroom if integration is needed)
- **Internet connection** – Required for model downloads and cloud APIs

### Installation Process

Step 1: Clone the Repository

Step 2: Install dependencies from requirements.txt

Step 3: Create a Environment .env file and configure credentials

Step 4: Run the Backend Server using fast API

Step 5: Launch the Frontend via Stream lit

Step 6: Access the Application

## Folder Structure :

- **app/** – Contains backend logic, APIs, and AI modules.
- **ui/** – Manages user interface (student/teacher dashboards, chat, reports).
- **data/** – Stores input datasets, student activity logs, and quiz results.
- **reports/** – Saves AI-generated reports and progress tracking files.
- **notebooks/** – Research and prototype notebooks (Google Colab/Jupyter).
- **docs/** – Technical documentation, design specifications, and diagrams.
- **screenshots/** – Demo captures for project documentation.
- **tests/** – Contains test cases for reliability and validation.

## Running the Application :

- Start the **backend server** to handle API requests.
- Launch the **frontend interface** (Gradio / Streamlit / React).
- Open the **application link** in your browser.
- Log in as **Student, Teacher, or Admin**.
- Students can **ask questions, take quizzes, and track progress**.
- Teachers can **upload materials, generate lesson plans, and view analytics**.
- Admins can **manage users and LMS integration**.
- View **AI-generated responses, reports, and dashboards** in real time.

## API Documentation :

The **EduTutor AI** system provides RESTful APIs to connect the frontend with backend services, AI modules, and LMS platforms.

- **User Interaction APIs** – Handle chat queries, quiz generation, and assessments.
- **Content Management APIs** – Allow uploading of documents, summarization, and report generation.
- **LMS Integration APIs** – Sync assignments, grades, and courses with external LMS platforms.
- **Feedback & Analytics APIs** – Collect feedback, generate insights, and detect performance anomalies.
- **Security & Authentication APIs** – Manage login, role-based access, and token verification.

## Key Characteristics:

- RESTful architecture
- JSON data format
- Token-based authentication
- Role-based permissions
- Standardized error handling

## Authentication :

- EduTutor AI uses **role-based authentication** for Students, Teachers, and Administrators.
- Users log in with credentials and receive a **secure token** (e.g., JWT) for session validation.
- Access is granted based on **user roles** – students access learning, teachers manage content, admins handle system controls.
- All data transfers are **encrypted** for security.
- Tokens are time-limited to ensure **safe session management**.
- Supports **OAuth2 and LMS logins** for seamless integration.

## User Interface :

The **EduTutor AI user interface** is designed to be simple, intuitive, and accessible for all users, including students, teachers, and administrators.

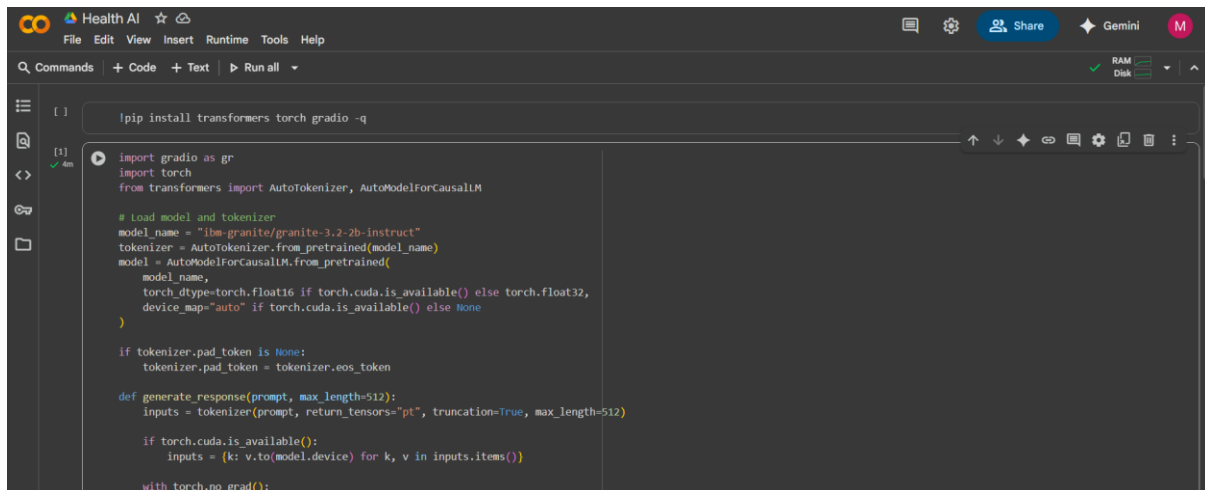
- **Dashboard:** Central hub displaying personalized content, quizzes, and progress tracking.
- **Navigation Sidebar:** Easy access to features such as Chat Tutor, Assignments, Reports, and Settings.
- **Chat Interface:** Allows students to interact with the AI tutor in natural language.
- **Teacher Panel:** Provides lesson plan generation, automated grading tools, and student performance analytics.
- **Admin Panel:** Manages users, system settings, and LMS integration.
- **Reports Section:** Displays AI-generated summaries, progress charts, and performance insights.
- **Design Focus:** Minimalist layout, responsive design, and user-friendly interactions.

## Testing :

The EduTutor AI system was tested in multiple phases to ensure reliability, accuracy, and smooth user experience.

- **Unit Testing** – Verified individual modules such as quiz generation, content summarization, and report creation.
- **Integration Testing** – Ensured proper communication between frontend, backend, AI model, and database.
- **System Testing** – Validated complete workflow, including user login, content upload, AI responses, and LMS synchronization.
- **User Acceptance Testing (UAT)** – Conducted trials with students and teachers to check usability and performance.
- **Performance Testing** – Measured system response time and scalability under multiple users.
- **Error Handling Tests** – Checked how the system responds to invalid inputs, missing files, or network issues.

## Screen Shots of the Coding :



```

[1] | pip install transformers torch gradio -q

[1] | 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
    | )
    |
    | if tokenizer.pad_token is None:
    |     tokenizer.pad_token = tokenizer.eos_token
    |
    | 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():

```



```

[1] | 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
    |
    | def concept_explanation(concept):
    |     prompt = f"Explain the concept of {concept} in detail with examples:"
    |     return generate_response(prompt, max_length=800)
    |
    | 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 all the answers in a separate section."
    |     return generate_response(prompt, max_length=1000)
    |
    | # Create Gradio interface

```



The screenshot shows a Google Colab notebook with the following Python code:

```
# Create 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)

            explain_btn.click(concept_input, inputs=concept_input, outputs=explanation_output)

        with gr.TabItem("Quiz Generator"):
            quiz_input = gr.Textbox(label="Enter a topic", placeholder="e.g., physics")
            quiz_btn = gr.Button("Generate Quiz")
            quiz_output = gr.Textbox(label="Quiz Questions", lines=15)

            quiz_btn.click(quiz_generator, inputs=quiz_input, outputs=quiz_output)

app.launch(share=True)
```

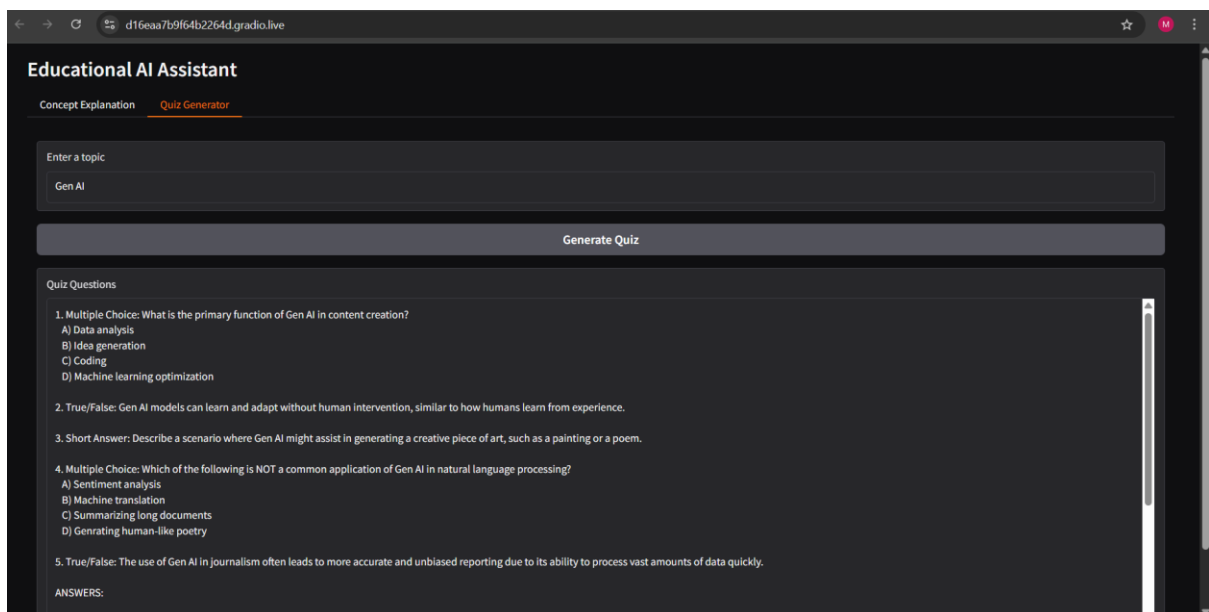
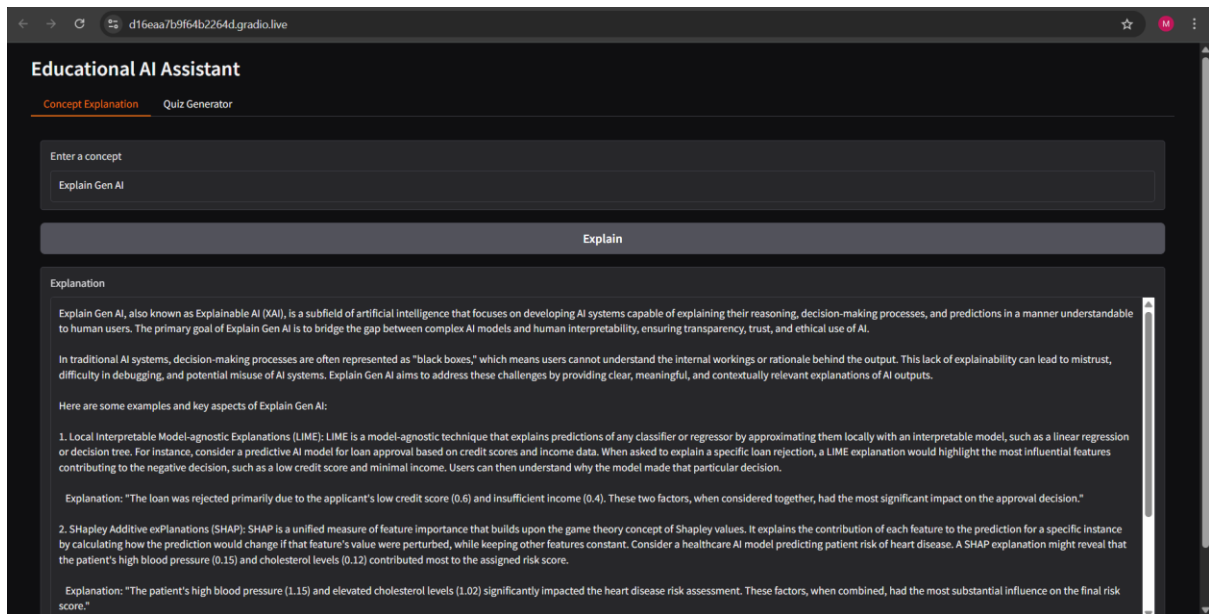
## Screen Shots of the Output :



The screenshot shows the terminal output of a Gradio application launch. It includes a warning about the 'HF\_TOKEN' secret, a list of files being downloaded, and a share link for the application.

```
/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, 375kB/s]
vocab.json: 777k/? [00:00<00:00, 15.3MB/s]
merges.txt: 442k/? [00:00<00:00, 12.7MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 44.1MB/s]
added_tokens.json: 100% [00:00<00:00, 2.39kB/s]
special_tokens_map.json: 100% [00:00<00:00, 25.3kB/s]
config.json: 100% [00:00<00:00, 38.0kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 2.24MB/s]
Fetching 2 files: 100% [03:30<00:00, 210.77s/it]
model-00001-of-00002.safetensors: 100% [03:30<00:00, 25.1MB/s]
model-00002-of-00002.safetensors: 100% [00:01<00:00, 18.4MB/s]
Loading checkpoint shards: 100% [00:20<00:00, 8.41s/it]
generation_config.json: 100% [00:00<00:00, 9.62kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://d16eaa7b2f64b2264d.gradio.live
This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (https://huggingface.co/spaces)
```





## Known Issues :

Although EduTutor AI functions effectively as a prototype, certain limitations and issues are still present:

- **High Resource Usage** – Large AI models require GPUs or cloud resources, making local execution difficult.

- **Dependency on Internet** – Stable internet is required for accessing models, APIs, and LMS integration.
- **Model Accuracy Variations** – AI-generated content may sometimes be too generic, inaccurate, or not fully aligned with syllabus requirements.
- **Limited LMS Integration** – Current LMS connectors work with basic features; advanced functions may need customization.
- **Scalability Challenges** – Handling large numbers of simultaneous users may affect response time.
- **Session Expiry** – Token expiration or session timeouts may interrupt user experience.
- **Data Privacy Concerns** – Storing student performance data securely requires stronger encryption and compliance with data protection standards.
- **Incomplete Offline Mode** – System cannot be used effectively without internet connectivity.

## Future enhancement :

To make EduTutor AI more powerful and widely usable, several improvements are planned for future versions:

- **Multilingual Support** – Extend AI tutoring to multiple regional and international languages.
  - **AR/VR Integration** – Provide immersive learning experiences through virtual classrooms and simulations.
  - **AI-Driven Career Guidance** – Recommend learning paths and career options based on student performance and interests.
  - **Advanced Analytics** – Deeper insights with predictive performance analysis and personalized recommendations.
  - **Voice-Based Interaction** – Enable students to interact with the AI tutor using speech instead of only text.
  - **Mobile App Development** – Provide Android and iOS applications for easy accessibility on smartphones.
  - **Enhanced LMS Integration** – Support full synchronization with Moodle, Google Classroom, Blackboard, and other major LMS platforms.
  - **Offline Mode** – Allow limited access to study materials and quizzes without internet connectivity.
  - **Improved Security** – Strengthen data protection with multi-factor authentication and compliance with data privacy laws.
-