PROJECT TITLE:

EDU TUTOR AI : PERSONALIZED LEARNING WITH   GENERATIVE AI AND LMS INTEGRATION

1. Introduction

Project Title: EDU TUTOR AI : PERSONALIZED LEARNING WITH   GENERATIVE AI AND LMS INTEGRATION

Team Leader: VANDHANA S

Team Members: JHANAVARSHNI S B

Team Members: INDHUMATHI K

Team Members: SARANYA D

Team Members: SHASHIBAALA V

2. Project Overview

- Purpose:

    The AI-Powered Quiz and Concept Learning Assistant is designed to make learning interactive and effective for students while giving educators tools to analyze performance. By integrating IBM Watsonx Granite models, the system generates concept explanations and quizzes in real-time.

Students can log in, select a topic and difficulty, attempt quizzes, and receive instant evaluation. Educators can review quiz data through a feedback loop, helping them track progress and improve teaching methods.

- Features:
-
    a. **Conversational Interface**

Key Point: Natural language interaction

Functionality: Allows students to enter questions or topics and receive clear explanations in plain language.

    b. **Policy Summarization**

Key Point: Simplified learning content

Functionality: Converts complex academic concepts into concise and easy-to-understand explanations with examples.

c. Resource Forecasting (Adapted as Quiz Difficulty Forecasting)

Key Point: Personalized quiz generation

Functionality: Suggests quiz difficulty (easy, medium, hard) based on student input or past performance.

d. Eco-Tip Generator

Key Point: Personalized study advice

Functionality: Provides students with tailored tips to improve understanding and preparation on specific topics.

e. Citizen Feedback Loop

Key Point: Teacher insights

Functionality: Stores quiz results and provides performance data to educators, enabling them to track student progress.

f. KPI Forecasting (Adapted as Student Performance Forecasting)

Key Point: Strategic learning support

Functionality: Predicts student performance trends based on quiz history, helping educators plan interventions.

g. Anomaly Detection

Key Point: Identifying irregular patterns

Functionality: Detects unusual answer patterns (e.g., random guessing or repetitive wrong answers) to flag potential issues.

h. Multimodal Input Support

Key Point: Flexible learning input

Functionality: Accepts different input types such as text based concepts, uploaded documents, or selected topics.

i. Streamlit or Gradio UI

Key Point: User-friendly interface

Functionality: Provides an intuitive interface for students and educators through Gradio (current) or Streamlit (planned).

j. AI Quiz Generation

Key Point: Interactive learning assessment

Functionality: Generates 5 quiz questions (MCQ, True/False, Short Answer). Answers are stored in the backend.

k. Quiz Submission & Evaluation

Key Point: Automated scoring

Functionality: Student answers are checked by the backend; scores and performance data are stored in Pinecone DB.

l. Educator Dashboard (Planned)

Key Point: Performance analytics

Functionality: Educators access aggregated results, view student progress, and identify learning gaps.

m. User Authentication

Key Point: Secure login

Functionality: Students log in with credentials or Google Classroom.

n. Gradio UI

Key Point: Easy interaction

Functionality: Provides tabbed navigation for explanations and quizzes.

3. Architecture

Frontend (Gradio):

The frontend is developed using Gradio, offering an intuitive interface with tab-based navigation for students and educators. It allows students to input concepts for explanation, request quizzes by selecting topic and difficulty, and submit answers. The interface is minimalist and accessible, ensuring smooth interaction. Planned enhancements include expanding the frontend with Streamlit to provide dashboards, performance reports, and educator views, making it scalable for larger classroom use.

Backend (FastAPI – planned, Python functions – current):

The backend is powered by Python functions in the current prototype, with FastAPI integration planned for production deployment. It manages quiz generation, response evaluation, and communication between the model and the database. Once fully integrated, FastAPI will expose modular endpoints for tasks like quiz submission, scoring, feedback collection, and performance retrieval. Its asynchronous capabilities will ensure high efficiency and real-time processing of multiple student requests.

LLM Integration (IBM Watsonx Granite):

The system integrates IBM Watsonx Granite models for natural language understanding and generation. The Granite 3.2-2b-instruct model is used to generate detailed explanations of concepts as well as quizzes with multiple types of questions. Carefully designed prompts guide the model to produce context-relevant outputs that are accurate, student-friendly, and consistent. This integration allows the assistant to handle varied educational queries in real time.

Vector Search (Pinecone):

Quiz results and embeddings are stored in Pinecone, a vector database that enables efficient semantic search and retrieval. Each student's answers and performance data are embedded and indexed for quick querying. Educators can search performance trends or retrieve similar past results using natural language queries, creating a seamless link between stored learning data and actionable insights. This supports both individual student tracking and class-level analysis.

ML Modules (Planned):

Lightweight machine learning modules will be integrated to enhance personalization and monitoring. Forecasting modules will analyze historical quiz results to predict student performance trends, while anomaly detection modules will flag irregular answer patterns, such as random guessing or consistently repeated mistakes. These ML enhancements will provide educators with predictive insights and early warnings, supporting targeted intervention and improved learning outcomes.

4.Setup Instructions

Prerequisites:

- o Python 3.9+
- o pip and virtualenv
- o API Keys for IBM Watsonx & Pinecone
- o Internet access for cloud-based services
- o (Optional) GPU with CUDA for faster execution

Installation Process:

1. Clone the repository.

2. Install dependencies:

- pip install -r requirements.txt

3. Create a .env file and add credentials

4. Launch the backend (FastAPI – planned).

5. Run the Gradio app:

- python app.py

5.Folder Structure

- ➢ app.py – Main application script that runs the Gradio interface for concept explanation and quiz generation.

- ➢ requirements.txt – Contains all dependencies required to run the project (Gradio, Transformers, Torch, etc.).

- ➢ .env – Stores API keys and environment variables for Watsonx and Pinecone.

- ➢ backend/ – Planned FastAPI backend directory for API endpoints, database operations, and model integration.

- ➢ backend/routes/ – Subdirectory for modular API routes (quiz generation, submission, results, feedback).

- ➢ backend/db/ – Handles Pinecone database connection and operations for storing/retrieving quiz results.

- ➢ backend/models/ – Contains model wrappers and integration logic for IBM Granite LLM.

- ➢ ui/ – Planned directory for educator dashboard and additional frontend features (performance reports, analytics).

6. Running the Application

To start the project:

- Step 1: Launch the Gradio application by running the main script app.py.

- Step 2: The system loads the IBM Granite model, tokenizer, and initializes backend logic.

- Step 3: Gradio generates a local URL and, if enabled, a shareable public link for accessing the app.

- Step 4: Navigate to the Concept Explanation tab, enter a concept or topic, and view the AI-generated detailed explanation.

- Step 5: Switch to the Quiz Generator tab, enter a subject/topic and difficulty level, and generate five quiz questions.

- Step 6: Attempt the quiz in the UI. The backend evaluates the answers and calculates the score.

- Step 7: Results are stored in Pinecone DB for performance tracking and later access by educators.

- Step 8: Planned enhancements include launching the backend via FastAPI and expanding the frontend with Streamlit dashboards for analytics and reporting.

7.API Documentation

Backend APIs available include:

POST /quiz/generate

- ➢ Accepts a topic and difficulty level as input.
- ➢ Uses the IBM Granite model to generate five quiz questions of different types.
- ➢ Returns the quiz questions to the frontend while storing correct answers in the backend.

POST /quiz/submit

- ➢ Accepts student answers as input.
- ➢ Evaluates the responses against stored correct answers.
- ➢ Returns the student's score and saves the result in Pinecone DB.

GET /results/student/{id}

- ➢ Retrieves past quiz results for a specific student.
- ➢ Provides detailed performance history across topics and difficulty levels.

GET /results/class/{class_id}

- ➢ Fetches aggregated performance data for an entire class.
- ➢ Helps educators identify learning gaps and monitor class progress.

POST /feedback

- ➢ Allows educators to submit feedback or comments on student performance.
- ➢ Feedback is stored for later review and analytics.

All endpoints will be tested and documented in Swagger UI once FastAPI integration is completed, ensuring developers and educators can inspect and trial the APIs during development.

8. Authentication

This version of the project currently runs in an open environment for demonstration purposes. However, secure deployments will integrate multiple authentication mechanisms to protect student and educator data.

Planned authentication features include:

- Token-Based Authentication (JWT or API Keys):

  Ensures only authorized users and applications can access the backend APIs.

- Google Classroom / OAuth2 Integration:

  Allows students and educators to log in directly using their Google accounts for seamless c   classroom integration.

- Role-Based Access Control:

  Provides different levels of access such as Student, Educator, and Administrator, ensuring   data security and proper usage.

- Session and History Tracking:

  Maintains user sessions, enabling personalized experiences and retrieval of past activity or   quiz history..

9.User Interface

Concept Explanation Tab

- o   Input: Concept name.
- o   Output: Detailed AI-generated explanation with examples.

Quiz Generator Tab

- o   Input: Topic + difficulty.

- o Output: 5 quiz questions (MCQ, True/False, Short Answer).

Quiz Submission

- o Students select answers in UI.
- o Instant feedback and scoring.

Educator Dashboard

- o Graphs of class performance.
- o Student progress tracking.
- o Downloadable reports.

10.Testing

Testing was done in multiple phases to ensure reliability and correctness of the system.

Unit Testing:

Conducted on core functions such as generate_response(), concept_explanation(), and quiz_generator().

Verified that the model consistently generates relevant explanations and quiz questions.

API Testing (Planned):

FastAPI endpoints such as /quiz/generate and /quiz/submit will be tested using Swagger UI, Postman, and automated test scripts.

Manual Testing:

The Gradio interface was tested for input/output flow, quiz generation, and explanation accuracy.

Students and educators simulated real interactions to validate usability.

Edge Case Handling:

Tested with empty inputs, rare or uncommon topics, long prompts, and invalid quiz submissions.

Verified that the system handles errors gracefully without crashing.

Performance Testing (Planned):

Planned tests will measure response times under load, especially when multiple students generate quizzes simultaneously.

11.Known Issues

➢ Frontend Limitations:
The current Gradio interface does not fully support interactive quiz submission with multiple-choice selection.

➢ Educator Dashboard Pending:
Dashboard for educators to view performance trends and reports is not yet implemented.

➢ Model Variability:
The IBM Granite model may sometimes generate questions that do not fully align with the chosen difficulty level.

➢ Performance Constraints:
On CPU-only systems, response generation is slower compared to GPU-enabled environments.

➢ Partial Integration:
FastAPI backend endpoints are planned but not yet fully implemented in the current version.

12. Future Enhancements

- **Interactive Quiz UI:**
  Add support for multiple-choice selection with real-time evaluation and feedback.

- **Educator Dashboard:**
  Develop a Streamlit-based dashboard to display student performance analytics, graphs, and downloadable reports.

- **Google Classroom Integration**:

Enable seamless login and synchronization of student accounts with Google Classroom.

- **Adaptive Learning:**
  Implement ML modules to adjust quiz difficulty automatically based on student performance trends.

- **Multimodal Input:**
  Allow uploading of documents (PDFs, Word files) for automatic quiz generation and summarization.

- **Performance Optimization:**
  Improve system speed and scalability by enhancing backend architecture and leveraging GPU acceleration.

- **Report Generation:**
  Provide AI-generated study guides and personalized performance reports for both students and educators.