EduTutor AI: Personalized Learning

Project Documentation

1.Introduction

- Team Member 1: Yashika
- Team Member 2: Lakshana
- Team Member 3: Sowmiya
- Team Member 4: Jemima

2.project overview

- Purpose:

  The purpose of EduTutor AI: Personalized Learning is to make learning more student-centered, interactive, and accessible by using Artificial Intelligence. It provides learners with clear explanations of complex concepts and automatically generates quizzes to test understanding. By combining explanations with interactive self-assessments, EduTutor AI enables personalized learning experiences that adapt to the learner's needs.

- **Features**

Conversational Interface

Key Point: Natural language interaction

Functionality: Allows learners to ask questions and receive clear explanations in plain language, making complex topics easier to understand.

Concept Explanation

Key Point: Example-based learning support

Functionality: Provides detailed explanations of any topic along with examples, helping students grasp difficult concepts effectively.

Quiz Generator

Key Point: Interactive self-assessment

Functionality: Generates five quiz questions in multiple formats (MCQ, true/false, short answer) with an answer section for self-evaluation.

AI-powered Responses

Key Point: Dynamic and adaptive explanations

Functionality: Uses advanced AI models to generate personalized responses that adapt to the learner's input and knowledge level.

User-Friendly Interface

Key Point: Easy navigation for all users

Functionality: Built with Gradio, it provides a simple interface with separate tabs for concept explanation and quiz generation.

3. Architecture

Frontend (Gradio):

The frontend is built with Gradio, providing an interactive web UI with multiple tabs including concept explanation and quiz generation. Each tab is modularized to handle a specific task, making the interface simple, scalable, and easy for learners to navigate without technical knowledge.

Backend (Python):

Python serves as the backend framework that powers the application logic for quiz generation and concept explanation. It manages prompt creation, model inference, and output formatting, ensuring smooth communication between user inputs and AI responses. The backend also handles tokenization and response post-processing.

LLM Integration (IBM Granite):

Granite 3.2 Instruct models from IBM Watsonx are used for natural language understanding and generation. Prompts are carefully designed to provide detailed explanations of concepts and generate quizzes with multiple formats, ensuring high-quality learning support.

Model Framework (PyTorch and Transformers):

The system relies on PyTorch for efficient model execution and the Hugging Face Transformers library for tokenization, decoding, and text generation. Together, they ensure accurate handling of prompts, optimized performance on CPU/GPU, and user-friendly responses.

Interface Logic (Gradio Blocks):

The application is organized using Gradio Blocks, which link UI elements such as buttons, textboxes, and output areas to backend functions. When a learner submits input, the corresponding function is triggered, and results are displayed instantly in the relevant section of the interface.

4. Setup Instructions

Prerequisites:

- Python 3.9 or later
- pip and virtual environment tool
- Internet access to download models and dependencies

Installation Process:

- Clone the repository
- Create and activate a virtual environment
- Install dependencies using pip install transformers torch gradio
- Run the Python script using python edtutor_ai.py
- Open the provided Gradio link in a browser to access the interface

5.Folder Structure

app/ – Contains the main script and helper modules for running the Educational AI Assistant.

app/models/ – Subdirectory for model loading and tokenization logic using Hugging Face Transformers.

app/utils/ – Contains utility functions like response generation and prompt formatting.

ui/ – Contains Gradio interface components including layout definitions, input forms, and output displays.

edtutor_ai.py – Entry script for launching the Gradio app with concept explanation and quiz generation functionalities.

generate_response.py – Handles tokenization, text generation, and formatting of AI responses.

concept_explanation.py – Defines the prompt and logic for explaining concepts in detail.

quiz_generator.py – Defines the prompt and logic for creating quiz questions and answers.

6. Running the Application

To start the project:

➢ Launch the Python script to initialize the Gradio application.

➢ The Gradio server automatically creates a web interface with multiple tabs.

➢ Navigate through tabs to access features such as concept explanation and quiz generation.

➢ Enter a topic or concept in the input box and interact with the AI assistant in real time.

➢ All interactions are processed instantly, with the backend handling AI responses and dynamically updating the frontend.

Frontend (gradio):

The frontend is built with Gradio, offering an interactive web UI with multiple tabs including concept explanation and quiz generation. The interface is simple and lightweight, making it easy for learners to type queries, receive explanations, and attempt quizzes. Each tab is modularized to ensure scalability and smooth navigation.

Backend (python):

Python powers the backend logic that connects user input with the AI model. It manages prompt creation, model inference, and response formatting. The backend ensures that queries entered in the frontend are processed by the model and the generated outputs are displayed instantly.

7. API Documentation

Backend functions available include:

entered concept and returns a detailed explanation with examples generated by the AI model.

POST /generate-quiz – Takes a topic as input and produces 5 quiz questions in multiple formats (MCQ, true/false, short answer), along with an answer section.

GET /health-check – Returns the status of the backend to ensure the model and tokenizer are loaded correctly.

POST /process-response – Handles user prompts, tokenizes input, runs inference on the Granite model, and returns a clean AI-generated output.

All functions are tested directly in the Gradio interface, which acts as both the UI and testing platform. Swagger UI is not included, but the API logic can be integrated into FastAPI or Flask if needed.

8. Authentication

This version of the project runs in an open environment for demonstration. However, secure deployments can integrate:

- Token-based authentication using JWT or API keys
- OAuth2 integration with IBM Cloud or other identity providers
- Role-based access control for different types of users (e.g., student, teacher, admin)
- Planned enhancements include user sessions, login history, and personalized learning progress tracking

9. User Interface

The interface is simple and student-friendly, focusing on accessibility for non-technical users. It includes:

- Sidebar for navigation
- Tabbed layouts for Concept Explanation and Quiz Generator
- Real-time form inputs and outputs
- Clean text display for explanations and quiz questions
- Minimal design for clarity and quick understanding

---

10. Testing

Testing was carried out in multiple phases:

- Unit Testing – for response generation and prompt functions
- Interface Testing – within the Gradio UI for real-time input/output
- Manual Testing – checking concept explanations, quiz flow, and answer accuracy
- Edge Case Handling – empty inputs, invalid topics, long queries

Each feature was validated to ensure consistent performance.

---

11. Screenshots

(Insert key screenshots of the Gradio interface:

- Home page
- Concept Explanation tab
- Quiz Generator tab
- Example quiz output

12. Known Issues

- Sometimes responses are lengthy or repetitive
- Quiz answers may need manual checking for accuracy
- Limited customization of quiz format in current version
- No authentication in demo mode

---

13. Future Enhancements

- Add user authentication (JWT or OAuth2)
- Store user progress and history for personalized learning
- Expand quiz types (MCQs with options, scoring system, leaderboards)
- Export results in PDF/Excel formats
- Add speech-to-text and text-to-speech for accessibility