**EDU TUTOR AI- PERSONALIZATION LEARNING**

**PROJECT DOCUMENTATION**

**1.Introduction**

* Project title :Edu Tutor Ai- Personalization Learning
* Team member : Udayasri.N
* Team member : Vani.M
* Team member:Varsha.M

## 2. Project Overview

* **Purpose:**  
  The Edu Tutor AI is designed to enhance personalized education through AI-powered explanations and quizzes. It helps learners quickly understand complex concepts and practice them with automatically generated quizzes. Educators can use it to provide interactive lessons and self-paced learning.
* **Features:**

**Concept Explanation**

Key Point: Detailed, example-rich explanations.

Functionality: Users enter a topic, and the AI explains it in depth.

**Quiz Generator**

Key Point: Interactive learning with varied question types.

Functionality: Generates multiple-choice, true/false, and short-answer questions with answers.

**Personalized Learning**

Key Point: Adaptive content generation.

Functionality: Provides detailed responses tailored to the learner’s input.

**Gradio-based User Interface**

Key Point: Simple and accessible platform.

Functionality: Enables easy interaction via tabs for concept explanation and quiz generation.

### ****Architecture****

**1. Frontend (Gradio UI)**

* Built using **Gradio Blocks** for modular and interactive design.
* Implements **tab-based navigation**:
* **Concept Explanation Tab:** Accepts user input, processes AI-generated explanations.
* **Quiz Generator Tab:** Creates quizzes with answers dynamically.
* Minimalistic and accessible interface designed for both students and educators.
* Supports **real-time updates without page reloads**.

1. **Backend (Python)**

* Written in **Python**, managing request-response flow between frontend and model.
* Includes **prompt formatting, input validation, and output post-processing**.
* Can be extended with APIs (e.g., FastAPI or Flask) for multi-user deployment.

1. **AI/LLM Integration**

* Uses **IBM Granite 3.2 2B Instruct model** via Hugging Face Transformers.
* Incorporates **temperature-controlled sampling (temperature=0.7)** for balanced creativity.
* Utilizes **tokenizer with padding handling** for robust prompt processing.

1. **Response Generation Pipeline**

* User input → Tokenization → Model inference → Decoding → Post-processing → Output display.
* Handles **truncation (max length 512)** to prevent overflow.
* Filters special tokens and cleans up the response before displaying.

1. **Scalability & Deployment**

* Can run locally or be deployed on **cloud platforms (AWS, Azure, IBM Cloud)**.
* Supports GPU acceleration if available (torch.cuda.is\_available()).
* Future-ready for **multi-user scaling with containerization (Docker/Kubernetes)**.

1. **Data Flow**

* Inputs: User concepts/topics.
* Processing: AI model generates explanations or quizzes.
* Outputs: Explanatory text or structured quiz questions with answers.
* All processing is **stateless (no data stored permanently)**.

1. **Security & Privacy (Planned)**

* Current version: Open mode (no authentication).
* Planned:
* API key-based access control.
* Role-based permissions for students/teachers.
* Optional encryption for user data.

1. **Extensibility**

* Can integrate:
* Document-based learning (PDF, CSV uploads).
* Voice interaction via Speech-to-Text and Text-to-Speech.
* Progress tracking and adaptive learning paths.

## 4. Setup Instructions

### Prerequisites

* Python 3.9 or later
* pip package manager
* Libraries: gradio, torch, transformers

### Installation

1. Clone the repository.
2. Install dependencies using:

pip install -r requirements.txt

1. Run the application:

python app.py

**5. Folder Structure**

* app.py – Launches the full Gradio application with tabs.
* models/ – Handles AI model and tokenizer loading.
* utils/ – Modular helper scripts (validation, response formatting, etc.).
* interface/ – Defines each Gradio tab and the overall app layout.
* data/ – Reserved for future enhancements like saving user data or preloaded concepts.
* requirements.txt – Ensures reproducibility of dependencies.
* .env – Stores API keys or model paths for secure configurations.

### ****6.Running the Application****

1. **Local Setup**

* Clone the repository.
* Create and activate a virtual environment.
* Install required dependencies using pip install -r requirements.txt.
* Run the application: python app.py.
* Open the default link: [http://127.0.0.1:7860](http://127.0.0.1:7860" \t "_new).

1. **GPU Acceleration (Optional)**

* The script automatically detects GPU using torch.cuda.is\_available().
* Uses **torch.float16** for faster performance on GPU.

1. **Cloud Deployment**

* Can be deployed on **AWS, Azure, GCP, or IBM Cloud**.
* Use **Uvicorn or Gunicorn** for production hosting.
* Environment variables stored in .env for security.
* Supports containerization with Docker for scalability.

1. **Public Sharing**

* Built-in app.launch(share=True) generates a temporary public link.
* For permanent hosting, use **Gradio Spaces (Hugging Face)** or a custom domain.

1. **Troubleshooting**

* **Model not loading?** Check internet connection and verify model name: ibm-granite/granite-3.2-2b-instruct.
* **Port already in use?** Change port: app.launch(server\_port=7861).
* **Slow response?** Lower max\_length or run on GPU.

1. **Usage Verification**

* Test **Concept Explanation**: Enter a concept (e.g., "machine learning").
* Test **Quiz Generator**: Enter a topic (e.g., "data science").
* Ensure the quiz includes both **questions and answers section**.

### ****API Documentation****

1. **generate\_response(prompt, max\_length=512)**

* **Method:** Internal function
* **Purpose:** Generates AI-based responses for both explanations and quizzes.
* **Inputs:**
* prompt (string) – The user query or concept to process.
* max\_length (int, optional) – Maximum token length for the generated response.
* **Outputs:** Cleaned, human-readable text without special tokens.

1. **concept\_explanation(concept)**

* **Method:** Gradio function (triggered by Explain button).
* **Purpose:** Provides a detailed explanation of the given concept.
* **Inputs:**
* concept (string) – The topic or concept to explain.
* **Outputs:** Paragraph text with explanations and examples.

1. **quiz\_generator(concept)**

* **Method:** Gradio function (triggered by Generate Quiz button).
* **Purpose:** Creates quizzes with different question types (MCQs, True/False, Short Answer).
* **Inputs:**
* concept (string) – The topic for quiz generation.
* **Outputs:**
* A formatted list of 5 questions.
* An **ANSWER section** at the end with correct answers.

1. **Model Handling API**

* Loads **IBM Granite 3.2 2B Instruct model** from Hugging Face.
* Automatically assigns GPU (device\_map="auto") if available.
* Sets pad\_token to eos\_token when missing for compatibility.

1. **Error Handling**

* Returns user-friendly error messages for:
* Empty inputs.
* Invalid topics.
* Model loading failures.

1. **Planned API Endpoints (Future Enhancements)**

* **POST /explain** – Accepts a concept and returns a detailed explanation.
* **POST /quiz** – Accepts a topic and returns generated quiz with answers.
* **GET /health-check** – Checks if the app is running properly.
* **POST /upload-doc** – (Planned) Allows users to upload a document for contextual learning.

### ****8. Authentication****

**1. Current Version (Open Access)**

* The present version runs in **open mode** for demonstration purposes.
* No authentication or user role restrictions applied.
* Suitable for **testing, development, and educational use**.

1. **Planned Authentication Mechanisms**

* **API Key Authentication:**
* Each user or organization will be provided a unique API key.
* Keys will be validated for every request to restrict unauthorized access.
* **JWT (JSON Web Token) Authentication:**
* Tokens will be issued upon login and verified for each API call.
* Supports token expiry and renewal for secure sessions.
* **OAuth2 Integration:**
* Integration with platforms like **Google, IBM Cloud, or Microsoft Azure** for seamless sign-in.
* Allows single sign-on (SSO) in educational institutions.

1. **Role-Based Access Control (RBAC)**

* **Student Role:** Limited access to learning modules, explanations, and quizzes.
* **Teacher Role:** Can monitor students’ progress, create custom quizzes, and manage content.
* **Admin Role:** Full control, including user management and analytics.

1. **User Session Management (Planned)**

* Session tokens to maintain user state across multiple interactions.
* Auto-logout after inactivity to prevent misuse.
* History tracking for personalized learning recommendations.

1. **Data Security Measures**

* **Encryption:** Sensitive data stored or transmitted using HTTPS/TLS.
* **Environment Variables:** API keys stored securely in .env files.
* **Access Logging:** Monitors unauthorized attempts and provides audit trails.

1. **Future Enhancements**

* Multi-factor authentication (MFA) for sensitive accounts.
* Parental/teacher control features for minor users.
* Integration with **institutional LDAP directories**.

### ****9. User Interface****

**1. Interface Framework**

* Built using **Gradio Blocks** for an intuitive, modular design.
* Provides a **web-based interface**, accessible via browser (desktop and mobile).
* No installation required for end-users apart from accessing the shared link.

1. **Main Components**

* **Header Section:** Displays the project title and description.
* **Tabs Navigation:**
* Concept Explanation Tab – For entering topics and receiving detailed explanations.
* Quiz Generator Tab – For generating quizzes and answers dynamically.

1. **Input Elements**

* **Textbox for Concepts/Topics:**
* Accepts free-form user input (e.g., “machine learning”, “photosynthesis”).
* Includes placeholder hints for guidance.
* **Buttons:**
* Explain – Triggers concept explanation.
* Generate Quiz – Triggers quiz creation.

1. **Output Elements**

* **Explanation Textbox:** Displays generated concept explanations.
* **Quiz Output Box:** Shows generated quiz questions and an answer key.
* **Multi-line formatting** ensures better readability of responses.

1. **Design and Usability**

* **Minimalist design** to reduce distractions and focus on content.
* **Adaptive layout:** Works well on different screen sizes (desktop, tablet, mobile).
* **Real-time interaction:** No page refresh required for updates.

1. **Accessibility Features**

* Supports **plain language outputs** for easy comprehension.
* Future enhancements may include
* **Voice interaction (TTS/STT).**
* **Font size adjustments.**
* **Dark mode for low-light usage.**

1. **User Guidance**

* Placeholder text and labels help new users navigate easily.
* Error handling ensures informative messages when inputs are invalid or missing.

1. **Planned Enhancements**

* Add **progress tracking dashboard** for learners.
* **Downloadable reports (PDF/Docx)** for generated quizzes and explanations.
* Multi-language interface support (English, Hindi, Tamil, etc.).

### ****10. Testing****

**1. Unit Testing**

* Tested individual functions such as:
* generate\_response() – Validates AI output formatting.
* concept\_explanation() – Ensures correct prompt handling.
* quiz\_generator() – Verifies quiz question generation and answer formatting.
* Used **pytest/unittest** framework for isolated testing.

1. **Integration Testing**

* Verified seamless interaction between:
* Gradio interface and backend functions.
* Tokenizer and model loading pipeline.
* Input validation and output rendering.

1. **User Interface Testing**

* Checked tab navigation, button triggers, and response display.
* Tested responsiveness on **desktop, tablet, and mobile browsers**.
* Ensured **error messages appear correctly** for empty or invalid inputs.

1. **Performance Testing**

* Measured **response generation time** with/without GPU acceleration.
* Stress-tested with multiple concurrent inputs to assess load capacity.
* Optimized max\_length to balance speed and content richness.

1. **Edge Case Handling**

* Tested with:
* Empty concepts.
* Very long prompts (near token limit).
* Nonsensical inputs (e.g., random characters).
* Ensured graceful handling without application crashes.

1. **API Testing**

* Simulated API calls for internal functions (generate\_response, concept\_explanation, quiz\_generator).
* Used tools like **Postman/Swagger (planned)** for endpoint validation.

1. **Manual Testing**

* Conducted user-level testing to verify usability.
* Collected feedback from students/teachers for improvement.

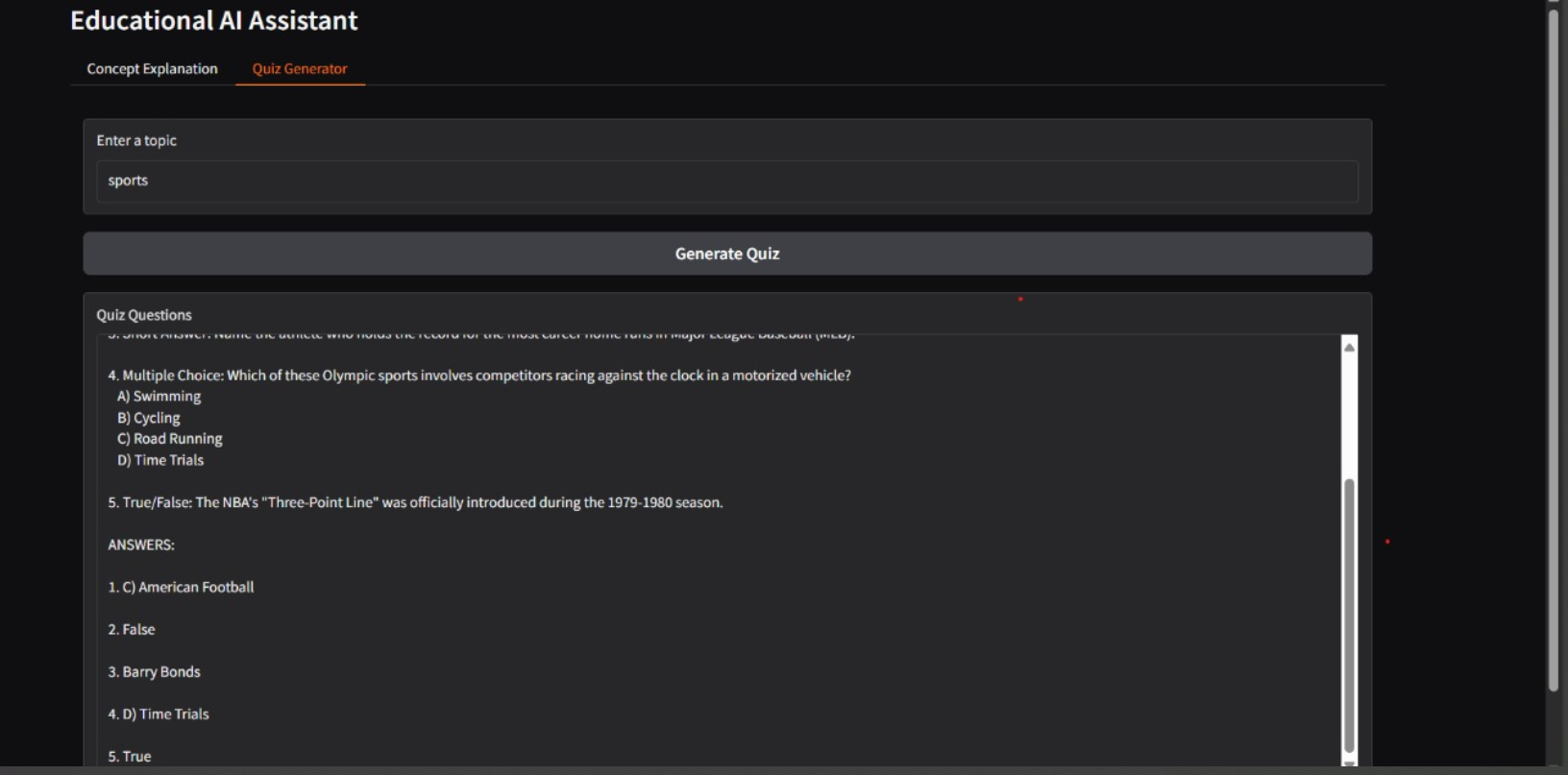
1. **Regression Testing**

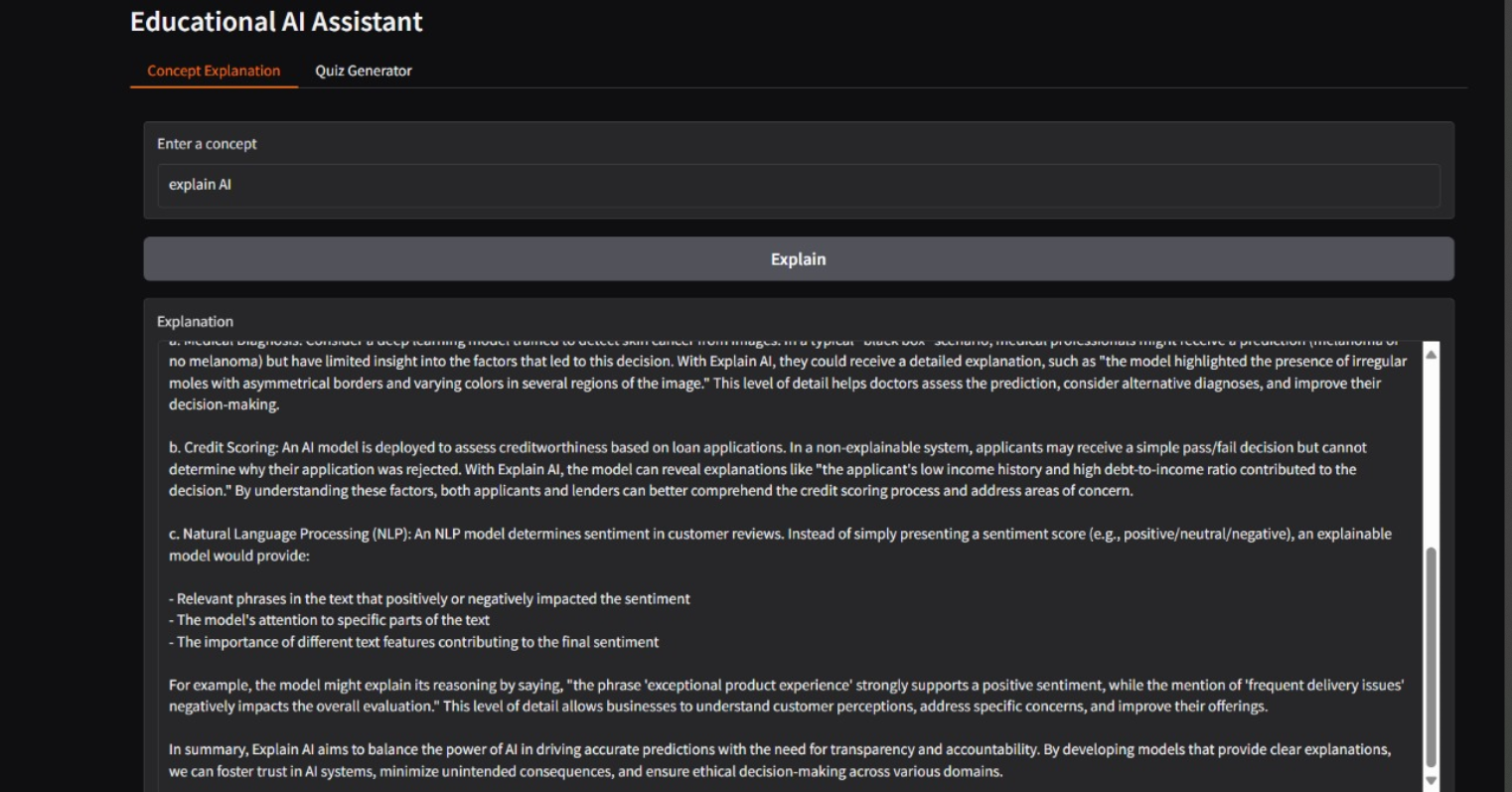
* After updates to the model or interface, re-tested core functions to ensure no previous features broke.

1. **Planned Future Testing**

* **Automated continuous testing** via CI/CD pipelines.
* **A/B testing** to evaluate different response generation settings.
* **Security testing** after authentication integration.

**11. Screen Shots**





### ****12. Known Issues****

**1. Model Dependency on Internet**

* The AI model (ibm-granite/granite-3.2-2b-instruct) requires an initial download from Hugging Face, causing delays or failures if internet is unstable.

1. **Limited Context Memory**

* The system does not retain conversation history between sessions.
* Each query is processed independently, which may affect contextual continuity.

1. **Generic or Repetitive Outputs**

* On vague or ambiguous prompts, the model may generate generic or repetitive content.

1. **Long Response Delay on CPU**

* Inference time is higher on non-GPU systems, especially with longer prompts (max\_length > 700).

1. **Lack of Persistent User Data**

* Current version does not store user interactions, quiz history, or learning progress.

1. **No Authentication in Demo Mode**

* Open access mode may expose the application to unauthorized usage in shared environments.

1. **Limited Language Support**

* Only supports English in the current version. Other languages planned for future updates.

1. **Quiz Format Variability**

* Sometimes quiz questions may not fully align with user expectations or may lack proper difficulty levels.

1. **Mobile Layout Limitations**

* While responsive, certain elements may appear cramped on small screens.

1. **Planned Fixes**

* Caching the model locally to reduce load times.
* Adding session-based history tracking.
* Implementing role-based access and authentication.
* Enhancing language model prompt engineering for more accurate outputs.

### ****13. Future Enhancements****

In future versions, Edu Tutor AI – Personalized Learning will evolve into a more robust and intelligent educational platform. The project aims to introduce secure authentication mechanisms such as API keys, JWT-based login systems, and role-based access control to provide differentiated access for students, teachers, and administrators. Personalized learning pathways will be developed to track individual progress, identify weaker areas, and recommend customized topics and quizzes to improve learning efficiency. Multi-language support will be integrated to make the system accessible to a global audience, including regional languages such as Hindi and Tamil.

The application will also feature offline mode capabilities, allowing users to cache the AI model locally for faster response times and reduced dependency on internet connectivity. Quiz generation will be enhanced with multiple difficulty levels, time-based challenges, and automatic grading, along with options to export quizzes as downloadable PDF or Word files. To improve accessibility, voice-based interaction will be introduced through text-to-speech and speech-to-text features, alongside a dark mode and customizable font sizes.

Cloud-based deployment and containerized scaling using platforms such as AWS, Azure, and IBM Cloud are planned to handle a larger user base efficiently. Teachers and administrators will have access to a comprehensive analytics dashboard to monitor learning engagement, topic popularity, and quiz statistics. Additionally, the system will allow uploading of documents and files, enabling contextual learning from PDFs, CSVs, and other materials. Integration with existing Learning Management Systems (LMS) such as Moodle and Google Classroom will further streamline its adoption in educational institutions. Enhanced security measures like multi-factor authentication, encrypted communication, and session tracking will ensure safe usage, while gamification elements such as badges, leaderboards, and reward points will make learning more interactive and engaging.