EDU TUTOR AI:PERSONOLIZED LEARNING

PROJECT DOCUMENTATION

1. Introduction

- **PROJECT TITLE: Edu Tutor Ai: Personalized learning**
- **TEAM MEMBER: Mohammed Yakoob Z**
- **TEAM MEMBER: BalaMurgan E**
- **TEAM MEMBER: Bharathi M**
- **TEAM MEMBER: Bhuvanesh K**

2. Project overview

- Purpose :

  The purpose of Edu Tutor AI is to revolutionize the learning experience by providing **personalized, AI-driven education** that adapts to each student's strengths, weaknesses, and pace of learning. By integrating with Learning Management Systems (LMS), Edu Tutor AI delivers tailored study plans, instant feedback, and smart content summarization. For teachers and institutions, it serves as a decision-support tool—offering performance analytics, automated assessments, and actionable insights into student progress. Ultimately, Edu Tutor AI bridges AI technology with pedagogy to create a more **engaging, inclusive, and effective learning ecosystem**.

Features:

- **Conversational Tutor**

  - *Key Point:* Natural language learning assistant
  - *Functionality:* Allows students to ask academic questions, receive explanations, and practice concepts in plain language.

- **Adaptive Learning Paths**

  - *Key Point:* Personalized curriculum planning
  - *Functionality:* Adjusts learning modules and recommendations based on student progress and performance.

- **Policy & Content Summarization**

  - *Key Point:* Simplified content understanding
  - *Functionality:* Converts lengthy academic texts, notes, and research papers into concise study summaries.

- **Assessment Generator**
    - *Key Point:* Automated evaluation
    - *Functionality:* Creates quizzes, practice tests, and assignments tailored to student needs.
- **Performance Analytics Dashboard**
    - *Key Point:* Teacher and student insights
    - *Functionality:* Tracks key learning KPIs such as progress, accuracy, completion rate, and provides predictive insights.
- **Feedback & Recommendation Loop**
    - *Key Point:* Continuous improvement
    - *Functionality:* Collects learner feedback and recommends study strategies or extra resources.
- **Anomaly Detection in Learning**
    - *Key Point:* Identify struggling learners early
    - *Functionality:* Flags unusual patterns like sudden performance drops or inactivity.
- **Multimodal Input Support**
    - *Key Point:* Flexible data handling
    - *Functionality:* Accepts text, PDFs, CSVs, and even lecture transcripts for analysis and summarization.
- **Streamlit or Gradio UI**
    - *Key Point:* User-friendly interface
    - *Functionality:* Provides dashboards for students, teachers, and administrators with easy navigation.

3. Architecture

Frontend (Streamlit/Gradio):
Interactive web UI with dashboards, chat, file uploads, performance reports, quiz interfaces, and feedback forms. Navigation via sidebar and modular page setup.

Backend (FastAPI):
FastAPI provides REST endpoints for tutoring conversations, quiz generation, student analytics, feedback storage, and document summarization. Optimized for async processing with Swagger integration.

LLM Integration (IBM Watsonx Granite):
Granite LLM models handle natural language explanations, summarization of learning materials, quiz generation, and adaptive feedback.

Vector Search (Pinecone):
Course documents and notes are embedded using Sentence Transformers and stored in Pinecone for semantic search, enabling "Ask your notes" functionality.

ML Modules (Forecasting & Anomaly Detection):
Machine learning models predict student outcomes (e.g., likelihood of passing) and detect anomalies such as disengagement or irregular learning behavior.

---

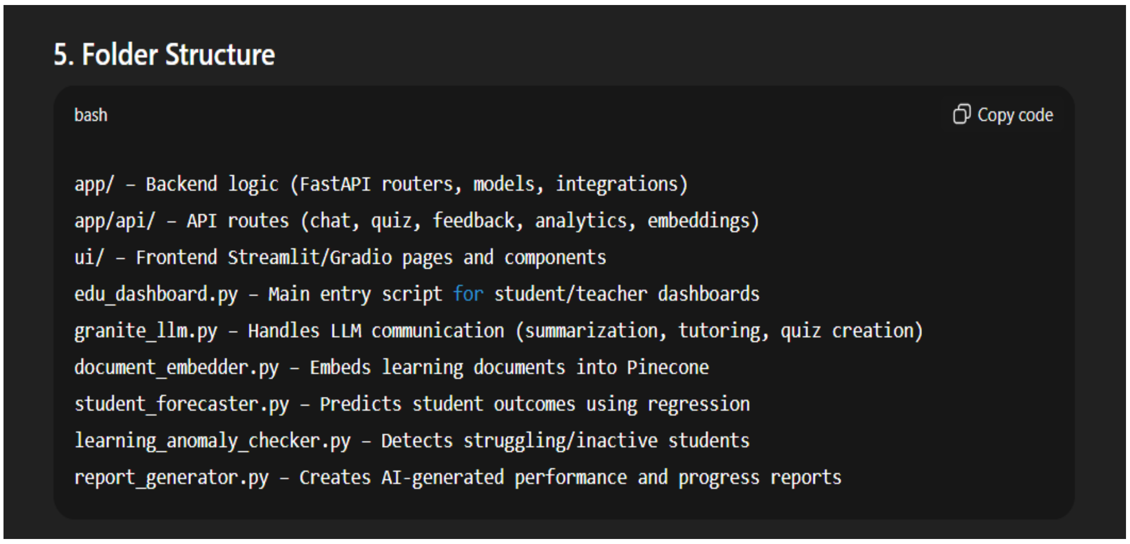
4. Setup Instructions

Prerequisites:

- Python 3.9 or later
- pip and virtual environment tools
- API keys for IBM Watsonx and Pinecone
- LMS integration credentials (Moodle, Canvas, etc.)
- Internet access

Installation Process:

1. Clone the repository
2. Install dependencies from `requirements.txt`
3. Configure `.env` with API keys and LMS credentials
4. Run backend server with FastAPI
5. Launch frontend with Streamlit/Gradio
6. Upload course content, interact with AI tutor, and access dashboards



5. Folder Structure

```bash
app/ – Backend logic (FastAPI routers, models, integrations)
app/api/ – API routes (chat, quiz, feedback, analytics, embeddings)
ui/ – Frontend Streamlit/Gradio pages and components
edu_dashboard.py – Main entry script for student/teacher dashboards
granite_llm.py – Handles LLM communication (summarization, tutoring, quiz creation)
document_embedder.py – Embeds learning documents into Pinecone
student_forecaster.py – Predicts student outcomes using regression
learning_anomaly_checker.py – Detects struggling/inactive students
report_generator.py – Creates AI-generated performance and progress reports
```

6. Running the Application

1. Start the **FastAPI backend** (API endpoints for tutor, analytics, and quiz generation).
2. Launch the **Streamlit/Gradio UI** for students and teachers.
3. Navigate via sidebar to access:
    - Personalized Tutor
    - Content Summarizer
    - Performance Dashboard
    - Quiz Generator
    - Feedback Forms
4. Upload course content (PDFs, notes, CSVs).
5. Interact with the AI tutor in real time and generate insights.

---

7. API Documentation

- **POST /chat/ask** – AI-powered tutoring Q&A
- **POST /upload-doc** – Upload and embed learning content
- **GET /search-docs** – Semantic search across course material
- **POST /generate-quiz** – Creates quizzes from course material
- **GET /get-recommendations** – Suggests personalized study tips
- **POST /submit-feedback** – Collects student feedback for review
- **GET /analytics/student** – Returns performance metrics and predictions

All APIs are available in Swagger UI for quick testing.

---

8. Authentication

This demo runs open for ease of testing. Secure deployment supports:

- **Token-based authentication** (JWT, API keys)
- **OAuth2** with LMS credentials
- **Role-based access** (Student, Teacher, Admin)
- Planned features: user session history and adaptive learning logs

---

9. User Interface

The UI is designed for simplicity and accessibility:

- Sidebar with navigation
- Progress cards and charts for learning KPIs
- Tabs for tutor chat, summarization, and quizzes
- Real-time handling of file uploads and inputs
- Downloadable progress and performance reports
- Clear instructions and guidance for first-time users

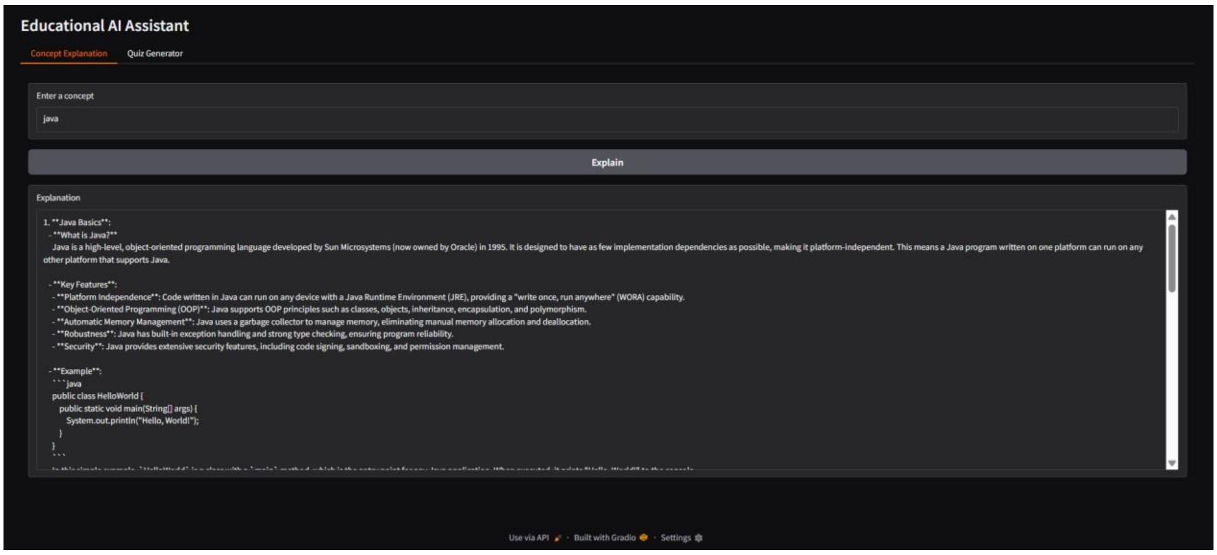
---

10. Testing

Testing carried out in multiple phases:

- **Unit Testing:** For prompt logic, summarization, and quiz generation
- **API Testing:** Via Swagger UI, Postman, and pytest scripts
- **Manual Testing:** For real student Q&A, document uploads, and analytics
- **Edge Case Handling:** Malformed files, incorrect credentials, unexpected inputs
- **Performance Testing:** Checked system response with large class datasets

OUTPUT SCREENSHOTS:



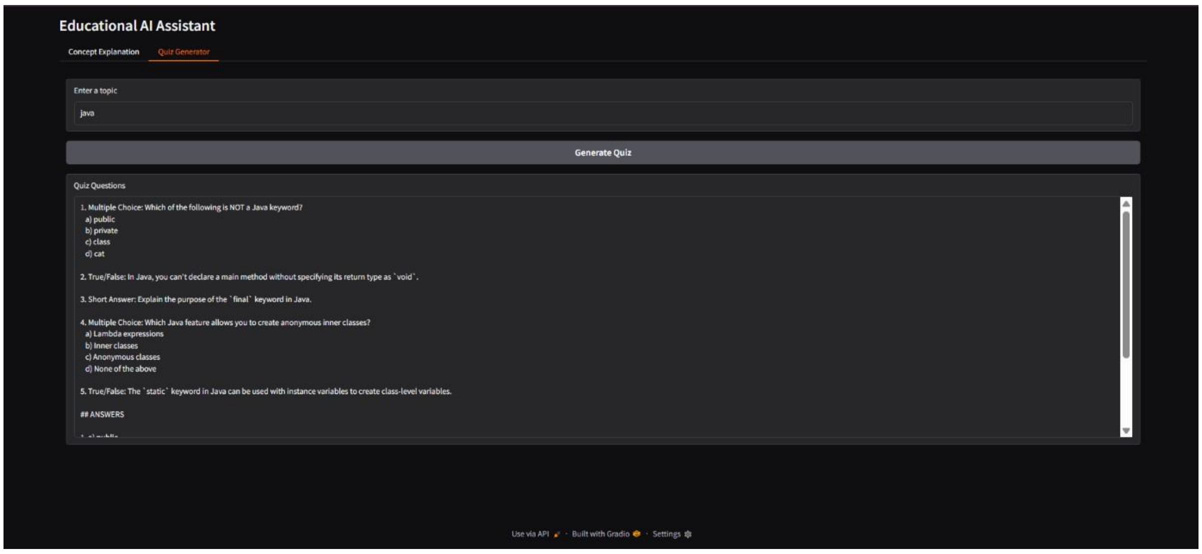
Enter a concept: java

Explanation

~~~
1. **Java Basics**:
  - **What is Java?**
   Java is a high-level, object-oriented programming language developed by Sun Microsystems (now owned by Oracle) in 1995. It is designed to have as few implementation dependencies as possible, making it platform-independent. This means a Java program written on one platform can run on any other platform that supports Java.

  - **Key Features**:
   - **Platform Independence**: Code written in Java can run on any device with a Java Runtime Environment (JRE), providing a "write once, run anywhere" (WORA) capability.
   - **Object-Oriented Programming (OOP)**: Java supports OOP principles such as classes, objects, inheritance, encapsulation, and polymorphism.
   - **Automatic Memory Management**: Java uses a garbage collector to manage memory, eliminating manual memory allocation and deallocation.
   - **Robustness**: Java has built-in exception handling and strong type checking, ensuring program reliability.
   - **Security**: Java provides extensive security features, including code signing, sandboxing, and permission management.

  - **Example**:
   ```java
   public class HelloWorld {
     public static void main(String[] args) {
       System.out.println("Hello, World!");
     }
   }
   ```
~~~



Enter a topic: java

Quiz Questions

```
1. Multiple Choice: Which of the following is NOT a Java keyword?
  a) public
  b) private
  c) class
  d) cat

2. True/False: In Java, you can't declare a main method without specifying its return type as `void`.

3. Short Answer: Explain the purpose of the `final` keyword in Java.

4. Multiple Choice: Which Java feature allows you to create anonymous inner classes?
  a) Lambda expressions
  b) Inner classes
  c) Anonymous classes
  d) None of the above

5. True/False: The `static` keyword in Java can be used with instance variables to create class-level variables.

## ANSWERS
```

11. Known Issues

- **Model Limitations** – The LLM sometimes gives overly generic answers or lacks deep subject expertise in niche domains.
- **Quiz Quality Variations** – Auto-generated quizzes may need manual review to ensure alignment with syllabus and difficulty level.
- **Dependency on Internet** – Since it relies on cloud APIs (Watsonx, Pinecone), the system requires stable internet connectivity.
- **Limited Offline Mode** – Current setup does not support offline usage, restricting access in low-connectivity regions.
- **Scalability Constraints** – Handling very large datasets (multiple classes or institutions) may slow down response times.
- **Feedback Analysis** – Student feedback is stored but advanced sentiment/emotion analysis is still minimal.
- **Integration Challenges** – Full LMS integration (Canvas, Moodle) may face compatibility issues across different versions.

---

12. Future Enhancements

- **Enhanced Personalization** – Deeper adaptive learning with reinforcement learning to refine content suggestions over time.
- **Gamification Features** – Points, badges, and leaderboards to increase student engagement.
- **Voice & Multilingual Support** – Conversational tutoring with speech-to-text and multilingual explanations.
- **Offline Learning Mode** – Local caching of AI models and content for low-internet environments.
- **Advanced Assessment Engine** – AI-driven grading of subjective answers (essays, reports) in addition to quizzes.
- **Collaborative Learning Spaces** – Group chat and peer-to-peer study rooms with AI moderation.
- **Teacher Assistance Tools** – Automated lesson planning, progress report generation, and curriculum alignment suggestions.
- **Emotion & Engagement Tracking** – Use webcam/microphone data (with consent) to analyze student engagement.
- **Blockchain-based Certificates** – Secure storage and verification of student achievements and completion certificates.
- **Integration Marketplace** – Plug-and-play connectors for popular LMS platforms, e-library systems, and ed-tech tools.

Conclusion:

The Edu Tutor project was developed with the vision of making learning more accessible, interactive, and personalized for students. By integrating technology with education, Edu Tutor provides a platform where learners can clarify doubts, strengthen concepts, and explore knowledge beyond traditional classrooms. This project not only demonstrates the practical application of our technical and creative skills but also highlights the potential of digital tools in shaping the future of education. Moving forward, Edu Tutor can be enhanced with more advanced features such as AI-driven recommendations, multilingual support, and interactive assessments, ensuring that it continues to serve as a valuable learning companion for students everywhere.