Project Documentation

1. Introduction

Project Title:
Edu Tutor AI: Personalized Learning Generative AI with IBM

Team Members:
- T. Janani
- G. Kayalvizhi
- A. Janani
- R. Infant Selciya
- S. Afrin Fathima

---

2. Project Overview

Purpose:
The purpose of Edu Tutor AI is to empower students with a personalized learning experience tailored to their pace, interests, and style of understanding. By leveraging IBM Watsonx generative AI and real-time learning analytics, the system supports students with adaptive lessons, quizzes, and instant tutoring. For teachers, it acts as a smart teaching assistant—providing insights into student progress, identifying learning gaps, and recommending interventions. Ultimately, Edu Tutor AI bridges advanced AI technology with education to foster effective, inclusive, and engaging learning.

Features:
- Conversational Interface: Interactive tutoring in natural language.
- Lesson Summarization: Converts lengthy material into concise study notes.
- Adaptive Quiz Generator: Creates personalized quizzes.
- Learning Path Recommendation: Suggests topics/resources based on progress.
- Student Feedback Loop: Refines learning experience through feedback.
- KPI Forecasting for Educators: Provides performance insights.
- Anomaly Detection: Identifies early signs of learning struggles.
- Multimodal Input Support: Accepts PDFs, text, and multimedia.
- Streamlit or Gradio UI: User-friendly dashboards.

---

3. Architecture

Frontend (Streamlit):
Dashboards for students and teachers with chat, quizzes, reports.

Backend (FastAPI):
Handles tutoring, quiz generation, and reporting APIs.

LLM Integration (IBM Watsonx Granite):
Provides tutoring, summarization, and adaptive quiz generation.

Vector Search (Pinecone):
Stores and retrieves educational material using semantic search.

ML Modules:
Forecasts student performance and detects anomalies.

---

4. Setup Instructions

Prerequisites:
- Python 3.9+
- pip & virtual environment
- IBM Watsonx API keys
- Pinecone API key
- Internet access

Steps:
1. Clone repository
2. Install dependencies from requirements.txt
3. Configure .env with credentials
4. Run backend (FastAPI)
5. Launch frontend (Streamlit)
6. Upload learning materials & interact with Edu Tutor AI

---

5. Folder Structure

- app/ : Backend logic (chat, quiz, summarization)
- app/api/ : API routes
- ui/ : Streamlit frontend
- edu_llm.py : IBM Watsonx integration
- quiz_generator.py : Adaptive quizzes
- lesson_embedder.py : Semantic embeddings
- student_forecaster.py : Performance prediction
- progress_checker.py : Anomaly detection
- report_generator.py : Learning reports

---

6. Running the Application
1. Start FastAPI backend
2. Run Streamlit dashboard
3. Navigate sidebar (Tutoring, Quizzes, Reports, Feedback)
4. Upload PDFs/notes
5. Chat, take quizzes, and track progress

---

7. API Documentation
- POST /chat/ask : Ask questions
- POST /upload-doc : Upload study material
- GET /search-docs : Find related content
- GET /generate-quiz : Create adaptive quizzes
- POST /submit-feedback : Store feedback
- GET /progress-report : Generate reports

---

8. Authentication
- JWT/API key authentication

- Optional OAuth2 (IBM Cloud)
- Role-based access (Student/Teacher/Admin)

---

9. User Interface
- Sidebar navigation
- Student Dashboard: progress, quizzes, summaries
- Teacher Dashboard: insights, reports
- Real-time tutoring chat
- PDF progress reports

---

10. Testing
- Unit testing: quiz, summarization, embeddings
- API testing: Swagger UI, Postman
- Manual testing: uploads, tutoring
- Edge cases: large files, invalid inputs

---

11. Screenshots
(Placeholder for UI screenshots)

---

12. Known Issues
- Limited support for handwritten notes
- Needs internet for real-time tutoring
- Model accuracy depends on uploaded material

---

13. Future Enhancements
- Mobile app version
- Gamified quizzes
- Multilingual tutoring
- Voice interaction
- LMS integration
