Edu Tutor AI: personalized learning

1. Introduction

Project Tittle:EDU TUTOR AI: PERSONALIZED LEARNING

Team Leader : ELAKKIYA S

Team member : HERLINA R

Team member : DEVADARSHINI M

Team member : GAYATHRI M

Team member : AARTHI S

2. Project Overview

Purpose: Edu TutorAI isdesigned to provide personalized, AI-powered tutoring for students of all levels. The platform adapts to each learner's pace, style, and subject needs, delivering tailored learning materials, quizzes, and feedback. By leveraging AI, the system enhances student engagement, tracks progress, and bridges gaps in traditional learning. Features: - Conversational Learning Assistant: AI-powered tutor for natural language Q&A.; - Personalized Study Plans: Adaptive daily/weekly learning schedules. - Interactive Quizzes & Feedback: Real-time evaluation with instant grading and feedback. - Progress Tracking Dashboard: Visual insights into performance and growth. - Content Summarization & Notes: Simplified study material for easy understanding. - Multi-Modal Input Support: Accepts text, PDFs, and multimedia content. - Gamified Learning: Rewards, badges, and streaks to motivate students.

3. Architecture

- Frontend (Streamlit/Gradio): Interactive student dashboard with quizzes, chat interface, performance analytics.
- Backend (FastAPI): Handles queries, personalization, quiz generation, and reporting.
- LLM Integration (OpenAI/Watsonx): Natural language processing for tutoring, explanations, and summarization.
- Database (PostgreSQL/Firebase): Stores user progress, results, and materials.
- Recommendation Engine: Suggests study plans and exercises using ML models.

4. Setup Instructions

- Prerequisites:Python3.9+,pip, API keys (OpenAI/Watsonx), Internet access.
- Installation: Clone repo, install requirements, configure `.env`, run FastAPI backend, launch Streamlit/Gradio frontend.

5. Folder Structure

- app/ –FastAPI backend logic
- app/api/ – API routes (chat, quiz, notes, feedback, progress tracking)
- ui/ – Streamlit/Gradio frontend components
- tutor_ai.py – Core LLM integration
- quiz_engine.py – Quiz generation and evaluation
- progress_tracker.py – Performance analytics and reports
- study_plan.py – Personalized study paths

6. Running the Application

- Start FastAPI backend server.
- Run Streamlit/Gradio dashboard.
- Navigate through learning modules.
- Upload study material or start tutoring session.
- Take quizzes, get feedback, and view reports.

7. API Documentation

- POST /chat/ask – Ask a question to AI tutor
- POST /upload-doc – Upload study material
- GET /get-study-plan – Get personalized learning schedule
- POST /take-quiz – Generate and attempt quiz
- GET /track-progress – View progress analytics

8. Authentication

- Token-based authentication (JWT/API keys).
- Role-based access (Student, Teacher, Admin).
- Planned: User sessions & history tracking.

9. User Interface

- Sidebar navigation, dashboard analytics, AI tutor chat, quizzes, notes, study plan recommendations.

10. Testing

- Unit Testing: AI functions & recommendation engine.
- API Testing: Swagger & Postman.
- Manual Testing: Chat, quizzes, summaries.
- Edge Case Handling: Large docs, invalid inputs.

11. Future Enhancements

- Voice-based AI tutor.
- Multi-language support.
- LMS integration (Moodle, Google Classroom).
- Parent/Teacher performance reports.
- AR/VR-based interactive learning.