Edututor AI

1. Introduction

- Project title: Edututor AI
- Team member: -
- Team member: -
- Team member: -
- Team member: -

2. Project Overview

- Purpose: The purpose of Edututor AI is to provide students with a personalized, AI-powered tutor that assists in learning, answering queries, and generating practice problems. It helps learners grasp concepts quickly, provides instant feedback, and adapts to the student's pace and style of learning.
- Features:
- • Conversational Interface – Natural language interaction for easy student engagement.
- • Subject Summarization – Summarizes textbooks, articles, and lessons into concise notes.
- • Practice Question Generator – Creates quizzes and practice tests tailored to student needs.
- • Progress Tracking – Monitors student performance and suggests areas of improvement.
- • Interactive Doubt Solving – Provides step-by-step explanations for math, science, and other subjects.
- • Multimodal Support – Accepts text, images, and PDFs for question answering and explanations.
- • Teacher Dashboard – Allows educators to track student progress and assign practice sets.

3. Architecture

- Frontend: A web-based interactive dashboard for students and teachers.
- Backend: Powered by FastAPI for high-performance APIs handling tutoring, quiz generation, and summarization.
- LLM Integration: Uses advanced AI models for natural language understanding and response generation.
- Database: Stores student profiles, progress history, and generated quizzes.
- Analytics: Provides insights into student weaknesses and strengths.

4. Setup Instructions

- Prerequisites:
- • Python 3.9 or later
- • pip and virtual environment tools

- • API keys for AI model services
- Installation Process:
- • Clone the repository
- • Install dependencies from requirements.txt
- • Configure .env file with credentials
- • Run the backend server using FastAPI
- • Launch the frontend dashboard

5. Folder Structure

- app/ – Contains backend logic including routes and models.
- ui/ – Contains frontend components and dashboards.
- quiz_generator.py – Module for generating practice quizzes.
- summarizer.py – Summarizes study materials.
- progress_tracker.py – Tracks and analyzes student performance.

6. Running the Application

- • Start the backend server with FastAPI.
- • Run the frontend dashboard.
- • Students log in to ask questions, take practice tests, and track progress.
- • Teachers access their dashboard to review reports and assign tasks.

7. API Documentation

- POST /ask-question – Accepts a student query and responds with an AI-generated explanation.
- POST /upload-material – Uploads study material for summarization.
- GET /generate-quiz – Creates quizzes based on selected topics.
- GET /track-progress – Retrieves student progress reports.

8. Authentication

- • Token-based authentication (JWT or API keys).
- • Role-based access for students, teachers, and admins.
- • User sessions and history tracking.

9. User Interface

- • Student dashboard with interactive Q&A;, quizzes, and reports.
- • Teacher dashboard for monitoring progress and assigning work.
- • Intuitive navigation with tabs for lessons, practice, and reports.

10. Testing

- Unit Testing: For question-answering and summarization functions.
- API Testing: Using Postman and Swagger UI.
- Manual Testing: For quiz generation, student interaction, and progress tracking.
- Edge Case Handling: Invalid queries, unsupported file formats, missing data.

11. Screenshots

- (To be added in future documentation)

12. Known Issues

- • Requires stable internet for real-time interaction.
- • Limited accuracy for very complex or ambiguous queries.

13. Future Enhancements

- • Voice-enabled tutoring for hands-free learning.
- • Support for regional languages.
- • Gamification features to motivate students.
- • Integration with popular LMS (Learning Management Systems).