Edu Tutor AI Project Documentation

1. Introduction

• Project Title: Edu Tutor AI
• Team Members: [Add Team Member Names Here]

2. Project Overview

• Purpose:
The purpose of Edu Tutor AI is to provide personalized, AI-powered tutoring support for students across different subjects and grade levels. By leveraging AI and natural language understanding, the platform delivers tailored learning experiences, real-time doubt resolution, and adaptive study resources. It empowers learners to progress at their own pace while assisting teachers with performance insights and content generation.

• Features:

Conversational Tutor	Interactive Q&A sessions with natural language support.
Personalized Learning Paths	AI generates tailored study plans based on student performance.
Homework Assistance	Step-by-step guidance on assignments and problem-solving.
Exam Preparation	Practice tests, flashcards, and mock exams for revision.
Performance Analytics	Visual progress reports for students, parents, and teachers.
Content Generation	Automatic generation of quizzes, summaries, and explanations.
Multilingual Support	Tutoring available in multiple languages.
Teacher Dashboard	Tools for monitoring student progress and generating reports.

3. Architecture

Frontend (Streamlit or Gradio): Provides an intuitive web UI with dashboards, chat interface, quizzes, and reports.
Backend (FastAPI/Django): Handles API requests for chat, content generation, assessments, and analytics.
LLM Integration: Utilizes advanced AI models for tutoring, summarization, and question generation.
Database: Stores user data, progress history, and generated content.
ML Modules: Adaptive learning engine, recommendation system, and performance prediction models.

4. Setup Instructions

• Python 3.9 or later
• pip and virtual environment tools
• API keys for AI/LLM provider
• Clone the repository
• Install dependencies from requirements.txt
• Create a .env file with credentials
• Run backend server
• Launch frontend dashboard

• Start interacting with Edu Tutor AI modules

5. Folder Structure

app/ – Backend logic including APIs, models, and integrations
app/api/ – API routes for chat, quizzes, reports, and analytics
ui/ – Frontend components (Streamlit/Gradio)
edu_tutor_llm.py – AI tutor logic and integration
quiz_generator.py – Creates practice quizzes and tests
progress_tracker.py – Monitors student progress
report_generator.py – Builds AI-generated learning reports

6. Running the Application

➢ Start the backend server
➢ Run the frontend dashboard
➢ Navigate through tutoring modules, quizzes, and reports
➢ Interact with the AI tutor for real-time assistance
➢ View analytics and reports

7. API Documentation

POST /chat/ask – Submits a student query and returns AI-generated explanation
POST /generate-quiz – Generates quizzes based on selected subjects
GET /progress-report – Retrieves performance analytics
POST /upload-notes – Accepts study material for summarization
POST /submit-feedback – Stores feedback from students and teachers

8. Authentication

Authentication methods include:
• Token-based authentication (JWT)
• OAuth2 for secure integrations
• Role-based access (student, teacher, admin)

9. User Interface

The interface includes:
• Sidebar navigation
• Chat interface for tutoring
• Quiz and exam modules
• Performance dashboard
• PDF/Excel report downloads

10. Testing

Testing included:
• Unit Testing for core AI functions
• API Testing with Postman/Swagger
• Manual Testing of UI features
• Edge Case Handling for incorrect inputs and large data

- Performance testing for response times