Edututor AI – Project Documentation

1. Introduction

Project Title: Edututor AI – Personalized Learning Assistant

Team Members:

•Sundar D(Lead)
• Tamil Selvan M(Member)
•Saravanan B(Member)
• Sakthi Balan S(Member)

2. Project Overview

Purpose:
Edututor AI is designed to provide students, teachers, and institutions with a personalized and adaptive learning experience. It uses AI to deliver on-demand tutoring, automated content summarization, and predictive academic insights to improve learning outcomes.

Features:

- • Conversational Interface – Natural language interaction, allows students to ask questions and get clear answers.
- • Lesson Summarization – Converts long lessons into concise summaries.
- • Performance Forecasting – Predicts student performance and suggests remedial topics.
- • Study Tips Generator – Gives daily study tips based on learner behavior.
- • Feedback Loop – Collects feedback to improve course content and learning paths.
- • KPI Tracking – Academic progress monitoring.
- • Anomaly Detection – Detects unusual performance dips.
- • Multimodal Input Support – Accepts text, PDFs, and videos for analysis.
- • Streamlit/Gradio UI – User-friendly dashboard for both students and educators.

3. Architecture

Frontend (Streamlit/Gradio): Interactive dashboards, file uploads, chat interface, test reports, and performance graphs.

Backend (FastAPI): Handles tutoring requests, content processing, and analytics asynchronously.

LLM Integration (IBM Watsonx Granite): Generates explanations, study tips, summaries, and feedback responses.

Vector Search (Pinecone): Embeds uploaded study materials and enables semantic search across textbooks.

ML Modules (Forecasting and Anomaly Detection): Predict student scores and detect struggling learners using regression and classification.

4. Setup Instructions

• Python 3.9+
• Install dependencies from requirements.txt
• Configure .env with API keys
• Run backend with FastAPI
• Launch frontend via Streamlit
• Upload study content and interact with the modules

5. Folder Structure

app/ – Backend logic
app/api/ – Routes for chat, feedback, performance, document embedding
ui/ – Frontend pages for students and teachers
edututor_dashboard.py – Main Streamlit dashboard
granite_llm.py – Handles Watsonx model calls
document_embedder.py – Embeds study materials
performance_forecaster.py – Predicts grades and progress
anomaly_checker.py – Flags performance drops
report_generator.py – Creates personalized progress reports

6. Running the Application

Start FastAPI server
Run Streamlit dashboard
Navigate through the sidebar to access tutoring, analytics, and reports

7. API Documentation

POST /chat/ask – Ask a tutoring question
POST /upload-doc – Upload study materials
GET /search-docs – Find related concepts
GET /get-study-tips – Personalized learning tips
POST /submit-feedback – Student or teacher feedback

8. Authentication

JWT tokens or API keys
Role-based access for students, teachers, and admins

9. User Interface

Sidebar navigation
Progress charts and KPI cards
Tabs for tutoring, summaries, and forecasts
Real-time interaction and downloadable reports

10. Testing

Unit testing for summarization and tips
API testing with Swagger/Postman

Manual testing for file uploads and forecasts
Edge case handling for large files and invalid inputs

ś12. Known Issues

Limited multilingual support in early version
Forecasting accuracy improves with more data

13. Future Enhancements

Integrate AR/VR lessons
Add voice-based tutoring
Include gamified learning features
Expand teacher dashboards