Project Documentation

1. Introduction

• Project Title: Edututor AI

• Team Leader: Kanya S

• Team Members:

• Rahul R

• Vidhya T

• Sri Rama Jayanthi S

2. Project Overview

Purpose:

The purpose of Edututor AI is to empower students and educators by providing intelligent, interactive, and personalized learning assistance. Leveraging AI and real-time data, Edututor AI helps enhance educational experiences by offering contextual explanations, instant answers, and adaptive learning recommendations. For educators, it serves as a teaching aid—automating administrative tasks, summarizing complex subjects, and tracking student progress to support effective learning strategies. Ultimately, this system bridges technology and education, making learning more efficient, engaging, and accessible.

Features:

• Conversational Interface: Natural language interaction, allowing students and educators to ask questions and get real-time explanations.

• Content Summarization: Simplifies complex educational materials into understandable summaries.

• Personalized Learning Paths: Customized study plans based on individual performance.

• Performance Forecasting: Estimates performance trends and identifies improvement areas.

• Interactive Quizzes Generator: Automatically generates quizzes from study materials.

• Feedback Loop: Collects feedback to refine materials and system responses.

• Multimodal Input Support: Accepts text, PDFs, and CSVs for analysis and tracking.

• User-Friendly Interface: Intuitive dashboard for students and educators.

3. Architecture

Frontend (Streamlit): Interactive web UI with dashboards, file uploads, chat, feedback forms, and report viewers.

Backend (FastAPI): REST API endpoints for document processing, summarization, and performance analysis.

LLM Integration (IBM Watsonx Granite): Uses LLM for language understanding and generating learning recommendations.

Vector Search (Pinecone): Embeds educational documents for semantic search.

ML Modules: Lightweight ML models for performance forecasting and anomaly detection.

4. Setup Instructions

Prerequisites: Python 3.9+, pip, virtualenv, API keys for IBM Watsonx and Pinecone, Internet access.

Installation Process: Clone repo, install dependencies, configure .env, run FastAPI server, launch Streamlit frontend, upload data, interact.

5. Folder Structure

app/: Backend logic

app/api/: Modular API routes

ui/: Streamlit frontend components

smart_dashboard.py: Main dashboard entry

granite_llm.py: Interfaces IBM Watsonx Granite

document_embedder.py: Converts documents into embeddings

kpi_file_forecaster.py: Forecasts student performance

anomaly_file_checker.py: Flags irregular patterns

report_generator.py: Generates learning reports

6. Running the Application

Launch FastAPI server, run Streamlit dashboard, navigate via sidebar, upload data, interact, view reports.

7. API Documentation

POST /chat/ask, POST /upload-doc, GET /search-docs, GET /get-eco-tips (replaced by personalized learning tips), POST /submit-feedback.

8. Authentication

Open environment for demonstration; future: JWT, OAuth2, Role-based access (student, educator, admin).

9. User Interface

Minimalist design, sidebar navigation, KPI visualizations, tabbed chat and forecasting, real-time forms, PDF download.

10. Testing

Unit Testing, API Testing (Swagger, Postman), Manual Testing (file uploads, chat), Edge Case Handling.

11. Known Issues

(To be filled later)

12. Future Enhancements

User session management, history tracking, enhanced adaptive recommendations.