Edu Tutor AI – Project Documentation

# 1. Introduction

Project Title: Edu Tutor AI – Personalized Learning with Generative AI and IBM

Team leader: ( Vishnu Durga)

Team Members: ( Loganidhi, Mahalakshmi, Soundhar)

# 2. Project Overview

Purpose:  
Edu Tutor AI is an intelligent learning platform that uses Generative AI and IBM Watson services to deliver a personalized, adaptive learning experience. It helps students understand complex concepts, practice interactively, and track progress while giving teachers tools for performance analytics.

Key Objectives:  
- Personalize learning paths based on a student’s strengths, weaknesses, and pace.  
- Provide AI-generated explanations, quizzes, and real-time doubt resolution.  
- Deliver actionable insights to teachers and parents about learner progress.  
- Ensure scalability and accessibility for diverse learners.

Key Features:  
- User Authentication & Profiles: Student, Teacher, and Admin roles.  
- AI-Powered Learning Assistance: Generative AI for explanations, hints, and concept summaries.  
- Interactive Quizzes & Assessments: Auto-generated questions for continuous evaluation.  
- Progress Tracking & Analytics: Dashboards for teachers, students, and parents.  
- Multilingual Support: IBM Watson Language Translator for content delivery.  
- 24/7 AI Chatbot Support: Doubt-solving assistant using Watson Assistant.

# 3. Architecture

System Components:  
- Frontend: React.js for UI and dynamic content rendering.  
- Backend: Node.js / Express.js for API handling.  
  
AI Services:  
- IBM Watson Assistant – Conversational AI for query resolution.  
- IBM Watson Discovery – Extracting key concepts from large datasets.  
- IBM Watson Natural Language Understanding – Sentiment and topic analysis.  
  
Database: MongoDB / Firebase for storing user profiles, progress, and content.  
  
State Management:  
- Global State: Redux Toolkit for consistent data across the app.  
- Local State: useState and useReducer for component-level states.  
  
Routing:  
- react-router-dom for navigating between: Home, Courses, AI Tutor, Reports/Dashboard.

# 4. Setup Instructions

Prerequisites:  
- Node.js & npm  
- React.js  
- IBM Cloud Account (for Watson services)

Installation Steps:  
1. Install Node.js and npm.  
2. Clone the project repository: git clone <repo-url>  
3. Navigate to project folder: cd edu-tutor-ai  
4. Install dependencies: npm install  
5. Configure IBM Watson API keys in .env file.  
6. Start the development server: npm run dev  
7. Open the application in browser: http://localhost:5173

# 5. Folder Structure

src/  
├── components/  
│ ├── Header.js  
│ ├── Sidebar.js  
│ ├── AIChatbot.js  
│ ├── CourseCard.js  
├── pages/  
│ ├── Home.js  
│ ├── Courses.js  
│ ├── Dashboard.js  
│ ├── AITutor.js  
├── hooks/  
├── utils/  
├── state/  
└── index.js

# 6. Running the Application

Frontend:  
- npm run dev  
  
Backend:  
- npm run server  
  
AI Services Configuration:  
- Create IBM Cloud Watson Assistant instance.  
- Get API keys and add them to .env.  
- Connect Watson APIs in backend server for AI responses.

# 7. Component Documentation

AIChatbot.js: Handles user queries and fetches answers using Watson Assistant.  
Dashboard.js: Displays progress, performance analytics, and AI-generated recommendations.  
Courses.js: Shows available courses with difficulty and completion percentage.  
AITutor.js: Generates real-time explanations and practice questions.  
App.js: Main entry point handling routing and global state.

# 8. State Management

Global State: Redux Toolkit manages user authentication, courses, and progress data.  
Local State: Used for real-time input (e.g., chat messages, quiz answers).

# 9. User Interface

Include:  
- Home Page screenshot  
- AI Chatbot interface screenshot  
- Course list view screenshot  
- Analytics dashboard screenshot

# 10. Styling

CSS Frameworks: Tailwind CSS for responsive, modern UI.  
Custom Theming: Dark/Light mode support.  
Icons: React Icons for consistent UI elements.

# 11. Testing

Testing Approach:  
- Unit Testing: Jest for component-level testing.  
- Integration Testing: React Testing Library.  
- API Testing: Postman for Watson API calls.

# 12. Screenshots or Demo

Provide screenshots OR  
Add a link to a live demo/video presentation.

# 13. Known Issues

Latency during Watson API response on low-speed internet.  
Occasional incorrect difficulty level prediction.

# 14. Future Enhancements

Add voice-based tutoring using IBM Speech-to-Text and Text-to-Speech.  
Integrate Gamification – badges, leaderboards, and rewards.  
Add AI-generated lesson summaries and interactive flashcards.