**Edu Tutor AI**

**Project Documentation**

**1.Introduction**

• Project title : Edu Tutor AI

• Team member : V.Sabitha

• Team member : J.Rubika

• Team member : P.Santhiya

• Team member : T.Santhiya

**2. Project Overview**

**Purpose:**

Edu Tutor AI provides students with a personalized AI-powered learning assistant. It adapts to individual needs by offering real-time explanations, interactive quizzes, summarizations, and performance tracking. Teachers benefit from automated assessments, analytics, and feedback collection.

**Features:**

• Conversational Learning Assistant: Natural language interaction for answering academic questions.

• Quiz & Assignment Generator: Automated generation of quizzes, practice questions, and assignments from uploaded material.

• Personalized Study Plans: Adaptive schedules and topic recommendations based on performance.

• Progress Tracking Dashboard: Visual analytics of student learning progress.

• Content Summarization: Summarizes textbooks and lecture notes into concise study material.

• Multi-format Support: Accepts text, PDFs, presentations, and CSVs for analysis.

**3. Architecture**

* Frontend (Streamlit/Gradio): Interactive web UI with dashboards, file upload, chat, and analytics.
* Backend (FastAPI/Django): Manages APIs for chat, quiz generation, and summarization.
* LLM Integration (OpenAI GPT / Watsonx / LLaMA): Handles natural language processing and content generation.
* Database (PostgreSQL / Firebase): Stores user data, quizzes, performance, and feedback.
* Vector Search (Pinecone/FAISS): Enables semantic retrieval of documents and notes.
* ML Modules: Adaptive Quiz Generator and Student Performance Predictor for forecasting outcomes.

**4.** **Setup Instructions**

**• Prerequisites:-** Python 3.9+, pip, virtual environment, API keys, Internet connection.

* Clone repository
* Install dependencies via requirements.txt
* Create .env file with API keys and database credentials
* Start backend server (uvicorn main:app --reload)
* Run frontend dashboard (streamlit run app.py)
* Upload documents and interact with Edu Tutor AI

**5.Folder Structure**

* edu-tutor-ai/ ■■■ app # Backend logic (APIs, ML models) ■■■ app/api/ # Chat, quiz, summarization routes ■■■ ui/ # Frontend Streamlit/Gradio pages ■■■ vector\_store/ # Document embeddings ■■■ models/ # ML models for quiz generation & tracking ■■■ tutor\_dashboard.py # Launch script for frontend ■■■ llm\_connector.py # LLM communication (GPT/Watsonx) ■■■ quiz\_generator.py # Generates quizzes & assignments ■■■ progress\_tracker.py # Student progress analytics ■■■ summarizer.py

**6. Running the Application**

* Launch FastAPI server to expose backend endpoints.
* Run Streamlit/Gradio frontend dashboard.
* Navigate through dashboard menus (chat, quizzes, progress tracking).
* Upload study materials (PDFs, notes, etc.).
* View outputs such as quizzes, summaries, and student reports in real-time.

**7. API Documentation**

The backend exposes APIs that make Edu Tutor AI modular and developer-friendly. Key endpoints include:

* **POST /chat/ask**
  + Input: Natural language query from the student.
  + Output: AI-generated answer with explanations, references, or study tips.
  + Use case: “Explain Newton’s second law” → returns simplified explanation.
* **POST /upload-doc**
  + Input: Document file (PDF, text, CSV).
  + Output: Document embedded in the vector database for search and quiz generation.
  + Use case: Upload lecture slides for automated quiz generation.
* **GET /generate-quiz**
  + Input: Reference to uploaded notes/text.
  + Output: A set of questions (MCQs, short answers) with or without answers.
  + Use case: Teachers can generate quizzes from a chapter in seconds.
* **GET /summarize**
  + Input: Document ID or text block.
  + Output: Concise, point-based summary.
  + Use case: Summarize long textbook chapters into quick revision notes.
* **GET /progress-report**
  + Input: Student ID or class group.
  + Output: Performance metrics, weak topic identification, and learning recommendations.
  + Use case: Teachers monitor overall class progress.
* **POST /submit-feedback**
  + Input: Feedback form (student or teacher).
  + Output: Stored in database for analytics.
  + Use case: Students suggest improvements or report unclear AI answers.

**8.** **Authentication**

Edu Tutor AI can run in open demo mode. For secure deployments:

* Token-based authentication (JWT, API keys).
* OAuth2 for integration with Google/Microsoft accounts.
* Role-based access (Admin, Teacher, Student).
* User session tracking and history logging.

**9. User Interface**

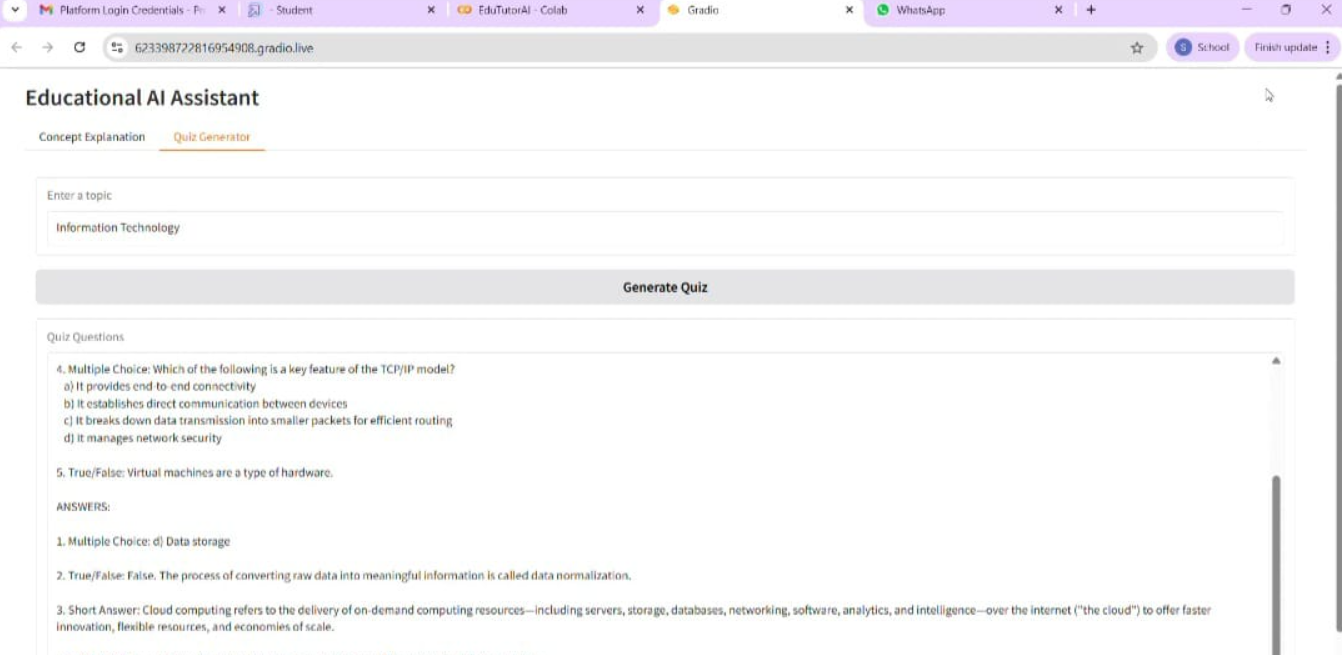
The UI is designed to be **minimalist, user-friendly, and responsive**:

* **Student Dashboard**
  + Provides chat interaction, quiz practice, personalized study plans, and graphical progress charts.
  + Students can download their quizzes and summaries for offline revision.
* **Teacher Dashboard**
  + Offers tools to generate assignments, monitor student performance, and collect feedback.
  + Teachers can export class performance data in CSV format.
* **Sidebar Navigation**
  + Contains links to each module: Chat, Quizzes, Progress Tracking, Summarization, Feedback.
  + Enables quick switching without reloading the app.
* **Export Options**
  + Reports and quizzes can be downloaded in **PDF/CSV formats**, making it easy to share or archive.
* **Accessibility**
  + Simple design ensures even non-technical users (students, school staff) can navigate easily.

**10. Testing**

1. Unit Testing: Validated core quiz generation and summarization functions.
2. API Testing: Conducted using Swagger UI and Postman.
3. Manual Testing: Verified real-time outputs for quizzes and summaries.
4. Edge Case Handling: Malformed files, empty queries, and large datasets

**11**. **Screenshot**



**12.** **Known Issues**

* Occasional inaccuracies in quiz difficulty adjustment.
* Summarizer may overlook rare keywords in technical subjects.
* Performance may vary depending on document size and API limits.

**13. Future Enhancements**

1. **Support for Multiple Languages and Regional Content**
   * To make Edu Tutor AI more inclusive, future versions will integrate **multi-language support**.
   * This allows students from different linguistic backgrounds to ask questions and receive explanations in their preferred language.
2. **Integration with Popular LMS Platforms (Google Classroom, Moodle)**
   * Edu Tutor AI will be integrated with **Learning Management Systems (LMS)** widely used in schools and universities.
   * For example, teachers using Google Classroom or Moodle will be able to directly sync course materials, generate quizzes inside the LMS, and push AI-generated reports back into the platform.
3. **Voice-based Tutoring for Hands-free Learning**
   * Students will interact with Edu Tutor AI using **voice commands**, making it act like a personal tutor.
   * The AI can read out explanations, ask oral quiz questions, and provide spoken feedback..
4. **Augmented and Virtual Reality Modules for Immersive Education**
   * AR/VR integration will take the learning experience beyond text and quizzes.

For example:

* + In AR, students could visualize 3D models of human anatomy, solar systems, or chemical structures.
  + In VR, they could explore historical events, virtual labs, or simulations in a fully immersive environment.
  + This makes learning more **interactive, engaging, and memorable**