**1. Introduction**

**Project Title**

**:** Edu Tutor AI : Personalized Learning

**Team Leader :** Sandhiya V

**Team Member :** Thaseen K

**Team Member :** Deepak MS

**Team Member :** Deepan K

**2. Project Overview**

Edu Tutor AI aims to bridge gaps in traditional learning by offering 24/7 tutoring support. The application allows users to ask subject-related questions, receive explanations, generate quizzes, summarize study material, and even interact with voice-based learning features. It can be integrated with learning management systems (LMS) and supports multiple subjects and educational levels.

Key Features:

AI-powered Q&A

Quiz generation

Lesson summarization

Voice-enabled chat

User progress tracking

Admin dashboard for managing content and users

**3. Architecture**

Edu Tutor AI follows a modular, service-oriented architecture that includes the following components:

[Frontend (React/Flutter)]

|

[API Gateway (Node.js/Express)]

|

[Authentication Service] [AI/NLP Engine (Python, OpenAI API)]

| |

[Database (MongoDB/PostgreSQL)] |

| |

[Admin Panel / Dashboard] |

Tech Stack:

Frontend: ReactJS (web) / Flutter (mobile)

Backend: Node.js + Express

AI Engine: Python, OpenAI GPT-4 API

Database: MongoDB / PostgreSQL

Authentication: JWT / OAuth2.0

Hosting: AWS / Vercel / Firebase

**4. Setup Instructions**

Pre-requisites:

Node.js (v16+)

Python 3.8+

MongoDB/PostgreSQL

API keys (OpenAI, Firebase, etc.)

Installation Steps:

# Clone the repo

git clone https://github.com/your-repo/edu-tutor-ai.git

cd edu-tutor-ai

# Install backend dependencies

cd backend

npm install

# Install frontend dependencies

cd ../frontend

npm install

# Set environment variables

cp .env.example .env

# Add API keys and DB URIs

# Run backend

cd ../backend

npm run dev

# Run frontend

cd ../frontend

npm start

**5. Folder Structure**

edu-tutor-ai/

├── backend/

│ ├── controllers/

│ ├── routes/

│ ├── services/

│ ├── models/

│ ├── utils/

├── frontend/

│ ├── src/

│ │ ├── components/

│ │ ├── pages/

│ │ ├── services/

│ │ └── assets/

├── ai-engine/

│ ├── prompt\_templates/

│ ├── summarizer.py

│ ├── quiz\_generator.py

├── docs/

├── tests/

└── README.md

**6. Running the Application**

To run both the frontend and backend:

Start the backend:

cd backend

npm run dev

Start the frontend:

cd frontend

npm start

Access the app at:

http://localhost:3000

**7. API Documentation**

All API endpoints are RESTful and documented via Swagger.

Sample Endpoints:

POST /api/auth/register – Register new user

POST /api/auth/login – Login and receive JWT

POST /api/ask – Ask a question to AI

POST /api/quiz/generate – Generate quiz from topic

POST /api/summary – Summarize given content

Authentication is required for all /api/\* routes using JWT tokens.

**8. Authentication**

Edu Tutor AI uses JWT-based authentication:

On login, a token is issued and stored in the client.

Protected routes validate the token.

Role-based access control is applied (student, tutor, admin).

Supports integration with OAuth2.0 providers like Google or Microsoft for institutional access.

**9. User Interface**

Modules:

Student Dashboard – Ask questions, track progress, generate quizzes

Tutor Dashboard – View user queries, create content

Admin Panel – Manage users, content, AI behavior

Key Features:

Responsive design (desktop/tablet/mobile)

Chat-style AI interaction

Voice input and output support

Light and dark themes

**10. Testing**

Testing Types:

Unit Testing: Jest (JS) / Pytest (Python)

Integration Testing: Supertest for APIs

End-to-End Testing: Cypress

Example:

# Run backend tests

cd backend

npm test

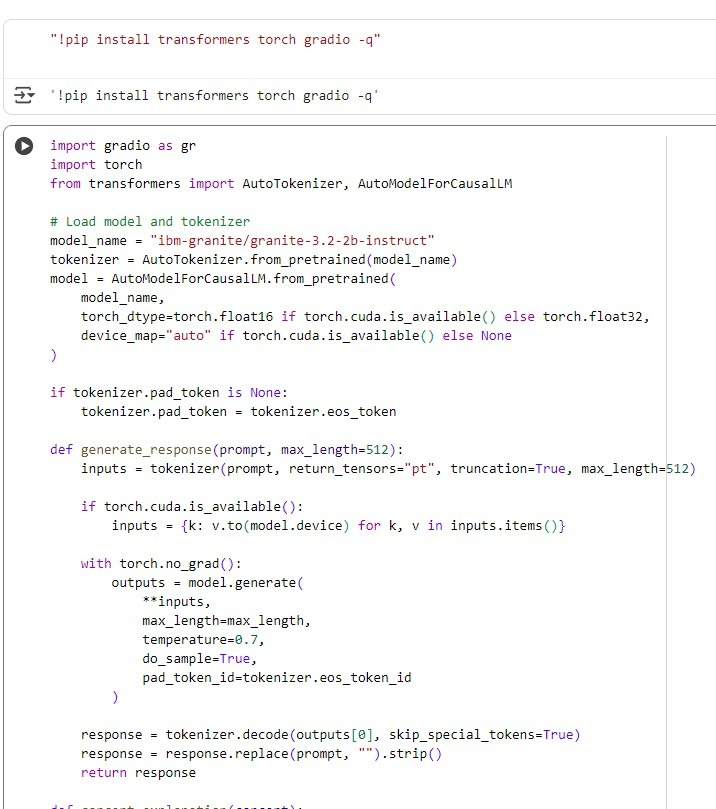
# Run frontend tests

cd frontend

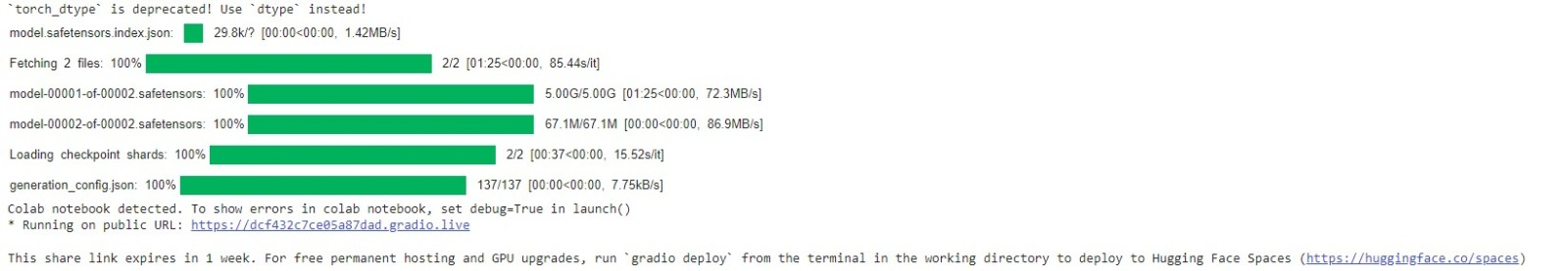
npm test

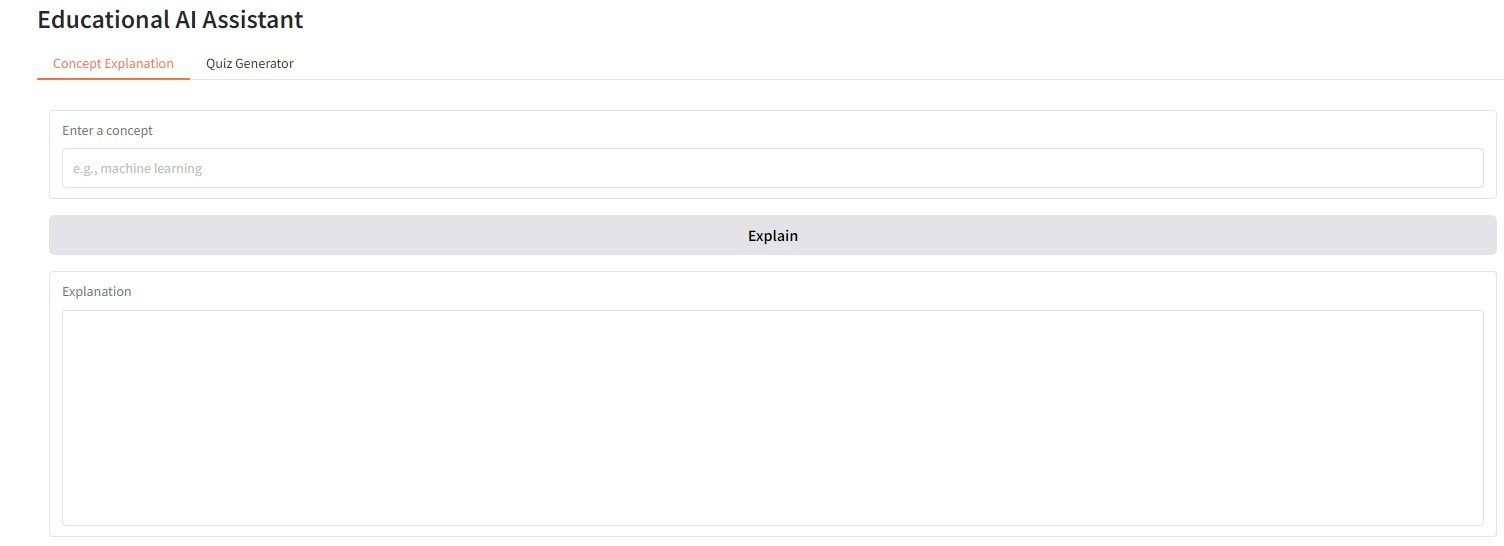
Test coverage reports are generated in the /coverage directory.

**11. Screenshots**









Quiz Generation

Admin Dashboard

**12. Known Issues**

Occasional latency in AI responses (due to API rate limits)

Limited offline functionality

Mobile app performance optimization pending

Context retention in long chat sessions can be inconsistent

**13. Future Enhancements**

Add support for more languages and localization

Implement AI tutor avatars with emotion-based responses

Integrate with LMS platforms (Moodle, Google Classroom)

Offline mode with limited AI capabilities

Adaptive learning engine for personalized learning paths

Parent and teacher analytics dashboards