EDU TUTOR AI:Personalized Learning
Project Document

1. Introduction

- **Project Title:** EduTutor AI
- **Team:**
  • Team Leader: Aysha Farshana SS
  • Team Member: Aasifah A
  • Team Member: Afrin R
  • Team Member: Mubeen G

2. Project Overview

Purpose:
EduTutor AI is an intelligent educational assistant designed to make learning more interactive, accessible, and personalized for students. It combines the power of AI with user-friendly interfaces to explain concepts in simple language, conduct quizzes, provide instant feedback, and adapt to a learner's pace.

For learners, EduTutor AI acts like a personal tutor that can explain topics in natural language, clarify doubts, and test knowledge through quizzes. For teachers, it serves as a supportive tool to automate quiz creation, analyze student performance, and generate reports.

Features:

- **Conversational Interface**
  Key Point: Natural language learning support
  Functionality: Students can ask questions, get explanations, and learn concepts in plain English.

- **Concept Explanation**
  Key Point: Simplified understanding
  Functionality: Explains topics (e.g., Python, Operating Systems) with clear examples.

- **Quiz Generator**
  Key Point: Interactive assessments
  Functionality: Generates quizzes based on subjects chosen by the teacher or student.

- **Instant Feedback**
  Key Point: Self-paced learning
  Functionality: Provides real-time scoring, hints, and corrections for quiz answers.

- **Performance Tracking**
  Key Point: Progress monitoring
  Functionality: Stores and visualizes learner performance for continuous improvement.

- **Content Flexibility**
  Key Point: Multi-subject support
  Functionality: Allows integration of different subjects like Python programming, Operating Systems, etc.

- **User-Friendly Interface**
  Key Point: Accessibility for all learners
  Functionality: Built with simplicity so that even non-technical students can use it easily.

3. Architecture

Frontend (Streamlit):
The frontend uses Streamlit to provide a simple and interactive web-based UI. It includes:

- Dashboards for quiz scores and progress
- Question-and-answer interface for explanations
- Quiz-taking pages
- File upload support for adding study materials

Backend (FastAPI):
FastAPI acts as the backend framework to handle:

- Quiz generation
- Answer validation
- Explanation delivery
- User data management

LLM Integration (OpenAI / IBM Watsonx / HuggingFace Models):
Large language models are used for:

- Generating explanations in simple language
- Creating quiz questions
- Providing hints and clarifications

Database (SQLite / Firebase / Pinecone for embeddings):
Used for:

- Storing questions, answers, and user progress
- Supporting semantic search for uploaded study materials

ML Modules (Adaptive Learning):

- Simple models analyze student performance
- Suggest easier or harder questions based on past performance

4. Setup Instructions

Prerequisites:

- Python 3.9 or later
- pip and virtual environment tools
- API keys (for LLM integration if required)
- Internet access

Installation Process:

1. Clone the repository
2. Install dependencies from `requirements.txt`
3. Configure environment variables (`.env` file)
4. Run the FastAPI backend server
5. Launch the Streamlit frontend
6. Interact with EduTutor AI through quizzes and explanations

5. Folder Structure

- **app/** – Contains FastAPI backend logic (quiz, explain, tracking modules)
- **app/api/** – Modular API routes for quiz, feedback, explanations
- **ui/** – Streamlit frontend components
- **edututor_dashboard.py** – Main script for launching the Streamlit dashboard
- **quiz_generator.py** – Generates quizzes from selected topics
- **explain_module.py** – Handles AI explanations for study content
- **progress_tracker.py** – Tracks student scores and progress reports
- **report_generator.py** – Creates performance summaries for teachers/students

6. Running the Application

To start the project:

1. Launch the FastAPI server to expose backend endpoints.
2. Run the Streamlit dashboard for the user interface.
3. Navigate using the sidebar to:
    - Take quizzes
    - Ask questions
    - View reports
4. Get real-time results and explanations.

7. API Documentation

Sample APIs:

- **POST /quiz/generate** – Generates a quiz based on a subject
- **POST /quiz/submit** – Submits answers and returns results
- **POST /explain** – Provides simple explanations for a topic
- **GET /progress** – Fetches learner performance data
- **POST /feedback** – Accepts user feedback for improvement

8. Authentication

For the demo version, EduTutor AI runs in an open environment. For secure deployments:

- Token-based authentication (JWT / API keys)
- Role-based access (student, teacher, admin)
- Planned enhancements: personalized user sessions and history tracking

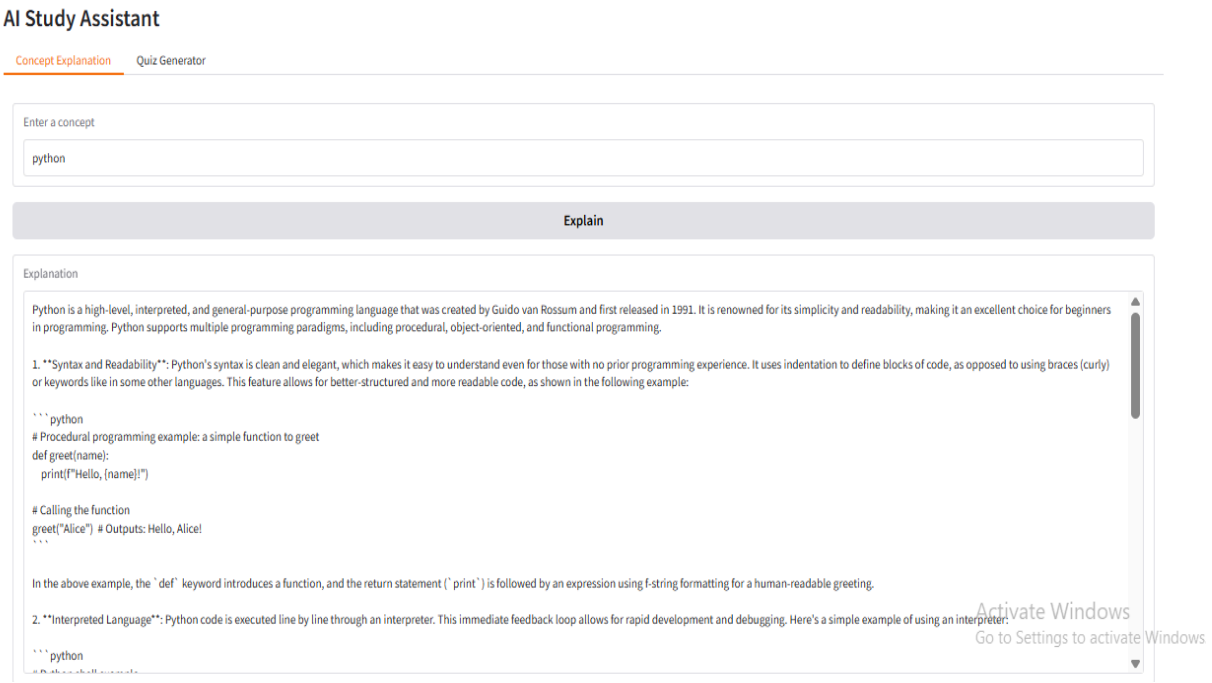
9. User Interface

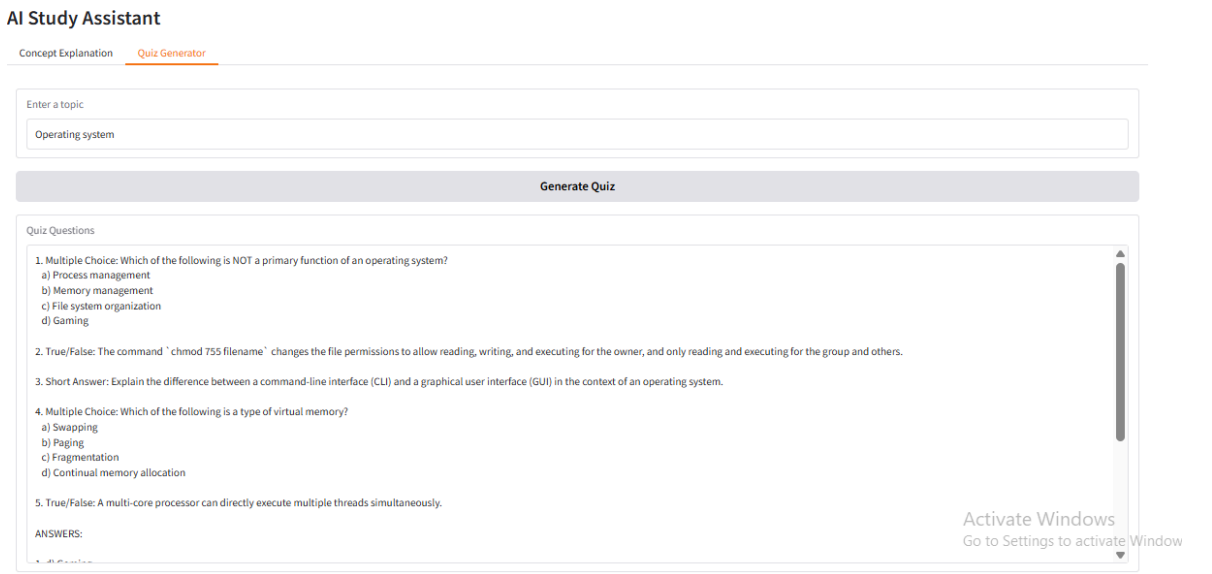
- Sidebar navigation for quizzes, explanations, and reports
- Real-time quiz answering with instant feedback
- Graphical progress charts
- Simple report download option

10. Testing

- **Unit Testing:** For quiz and explanation modules
- **API Testing:** Using Postman and Swagger UI
- **Manual Testing:** Checking quiz flow, scoring accuracy, and explanations
- **Edge Case Handling:** Wrong answers, empty inputs, repeated quizzes

11. Screenshots





12. Known Issues

- Quiz generation sometimes repeats questions
- Limited offline functionality without API access
- Current version supports only selected subjects (Python, OS)

13. Future Enhancements

- Support for more subjects (DBMS, Networks, etc.)
- Voice-based interaction for accessibility
- Gamified quizzes with levels and badges
- AI-driven personalized learning paths
- Mobile app version for wider reach