Edu Tutor - Project Documentation

1. Introduction

• Project Title: Edu Tutor AI:Personlized learning with genrative AI and LMS intergration

- Team Leader:Swetha R
- Team Member: Shalini M
- Team Member: Shalini R
  - Team Member: Sasi Priya M
  - Team Member: Sharmila V

2. Project Overview

• Purpose:

The purpose of Edu Tutor is to provide students with an AI-powered learning assistant that explains academic concepts in simple language, generates quizzes for practice, and offers personalized learning support. By leveraging AI and real-time data, Edu Tutor acts as a virtual tutor, helping learners across different subjects with interactive explanations, practice exercises, and progress tracking.

Features:

- Conversational Interface (natural language learning support)
- Concept Explanation (simplified subject understanding)
- Quiz Generator (MCQs, true/false, short answer practice)
- Personalized Learning Path (adaptive content suggestions)
- Progress Tracking (student performance monitoring)
- Resource Recommendations (study materials and references)
- Multimodal Input Support (text, PDFs, images for analysis)
- Gradio/Streamlit UI (user-friendly interface for students and teachers)

3. Architecture

Frontend: Streamlit or Gradio-based interactive dashboard with pages for explanations, quizzes, and student progress tracking.
Backend: FastAPI backend powering concept explanations, quiz generation, and student progress APIs.
LLM Integration: IBM Watsonx Granite or Hugging Face models for natural language processing.
Database: Stores user performance, quiz results, and progress history.
ML Modules: Adaptive learning engine recommending topics based on student weaknesses.

4. Setup Instructions

Prerequisites:
o Python 3.9 or later

o pip and virtual environment tools
o API keys for LLM service
o Internet access

Installation Process:
o Clone the repository
o Install dependencies from requirements.txt
o Configure API keys in .env
o Run FastAPI backend
o Launch frontend via Streamlit/Gradio
o Start interacting with Edu Tutor

5. Folder Structure

app/ – Contains backend logic including routers, models, and integration modules.
app/api/ – Subdirectory for modular API routes like quiz, explanation, and tracking.
ui/ – Contains frontend components for pages and layouts.
edu_dashboard.py – Entry script for launching the main dashboard.
llm_integration.py – Handles communication with LLMs.
quiz_generator.py – Creates quizzes for given topics.
progress_tracker.py – Tracks and visualizes student learning progress.

6. Running the Application

➢ Launch the FastAPI server to expose backend endpoints.
➢ Run the Streamlit/Gradio dashboard.
➢ Navigate through tabs for concept explanation, quiz generation, and progress tracking.
➢ Upload study materials if needed.
➢ Receive real-time explanations, quizzes, and personalized learning suggestions.

7. API Documentation

- POST /explain – Provides AI-generated explanation of a concept
- POST /generate-quiz – Generates quizzes for a given subject
- GET /track-progress – Returns student learning progress
- POST /upload-material – Uploads documents for analysis
- POST /feedback – Stores student/teacher feedback

8. Authentication

Edu Tutor supports secure deployment with:
- Token-based authentication (JWT/API keys)
- OAuth2 integration for teachers and admins
- Role-based access (student, teacher, admin)
- Session history for personalized experiences

9. User Interface

The Edu Tutor interface is designed to be simple and engaging for students. It includes tabbed layouts for explanations, quizzes, and progress tracking. The UI also supports PDF report downloads, progress charts, and adaptive suggestions.

10. Testing

Testing includes:
- Unit Testing: For quiz generation and explanation functions
- API Testing: Via Swagger UI/Postman
- Manual Testing: For quiz responses, explanations, and progress tracking
- Edge Case Handling: Invalid queries, large documents, unexpected inputs

Edu Tutor was validated to ensure reliability for students and educators.

11. Issues and Future Enhancements

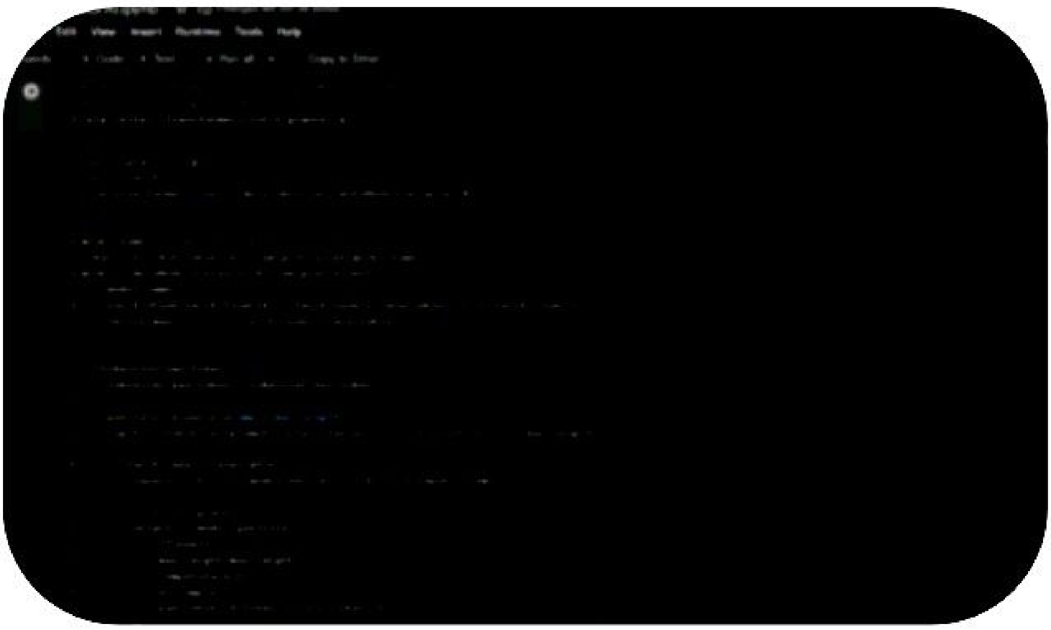
Issues Faced in the Project:
- Integration challenges with third-party APIs and LLM models caused occasional delays in response.
- Handling large document uploads led to performance bottlenecks during testing.
- Some UI/UX adjustments were needed to improve navigation for first-time users.
- Quiz generation sometimes provided repetitive or low-difficulty questions.
- Authentication required additional fine-tuning for role-based access.

Future Enhancements:
- Improve scalability to support higher student traffic with minimal latency.
- Enhance adaptive learning engine with deeper personalization using student analytics.
- Integrate voice-based interaction for better accessibility.
- Add gamification features such as leaderboards and reward points for students.
- Enable multilingual support for broader student engagement.
- Expand support for more file types including PPT, video lectures, and audio materials.

11. Project Screenshots

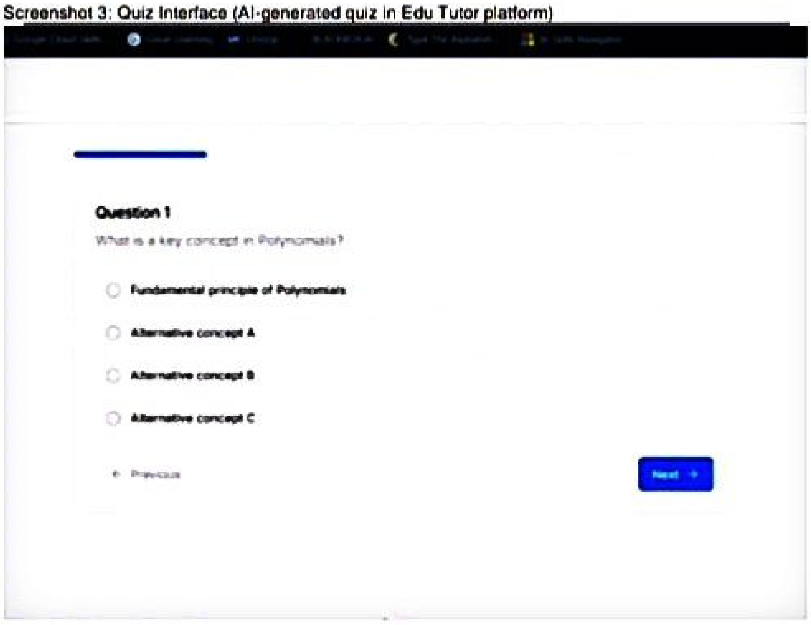
Screenshot 1: Model Integration (IBM Granite model search on Hugging Face)

Screenshot 2: Model Implementation (Colab notebook running Granite model code)



```
[illegible]
```

Screenshot 3: Quiz Interface (AI-generated quiz in Edu Tutor platform)



Question 1

What is a key concept in Polynomials?

- Fundamental principle of Polynomials
- Alternative concept A
- Alternative concept B
- Alternative concept C

12. Issues Faced in the Project

- Integration challenges with third-party APIs and LLM models caused occasional delays in responses.
- Handling large document uploads led to performance bottlenecks during testing.
- Some UI/UX adjustments were required to improve navigation for first-time users.
- Quiz generation sometimes provided repetitive or low-difficulty questions.
- Authentication required additional fine-tuning for role-based access.

13. Future Enhancements

- Improve scalability to support higher student traffic with minimal latency.
- Enhance the adaptive learning engine with deeper personalization using student analytics.
- Integrate voice-based interaction for better accessibility.
- Add gamification features such as leaderboards and reward points for students.
- Enable multilingual support for broader student engagement.
- Expand support for more file types including PPT, video lectures, and audio materials.