EduTutor AI with IBM

Team Members :

\* Vijayalakshmi V

\* Tamilvani M

\* Thavaselvi T

\* Yogalakshmi M

1. Introduction:

**Project Title:** EduTutor AI with IBM

**Purpose:**

The purpose of EduTutor AI with IBM is to provide an intelligent educational assistant that supports

learners and educators in optimizing the learning experience. By leveraging AI, real-time feedback, and

data-driven recommendations, EduTutor AI helps students improve their academic performance while

offering personalized guidance.

2. Features:

\* Conversational Interface for natural interaction

\* Content Summarization for quick learning

\* Performance Forecasting using data

\* Quiz Generator for practice

\* Study Tips Generator personalized for users

\* Feedback System for interactive improvement

\* Resource Recommendations aligned with content

\* Multimodal Input Support for diverse formats

\* User-Friendly Dashboard

**3. Architecture:**

**Frontend (Streamlit)**: Interactive UI with dashboards, file uploads, chat, reports, and feedback forms.

**Backend (Fast API):** API endpoints for document summarization, quizzes, learning recommendations,and user interaction.

**LLM Integration (IBM Watsonx Granite):** Natural language understanding for content generation and summarization.

**Vector Search (Pinecone):** Semantic search across educational documents using embeddings.

**ML Modules:** Prediction models for academic forecasting and anomaly detection using Scikit-learn.

4. Setup Instructions:

**Prerequisites:**

\* Python 3.9 or later

\* pip and virtual environment tools

\* API keys for IBM Watsonx and Pinecone

\* Internet access

**Installation Process:**

\* Clone the repository

\* Install dependencies from requirements.txt

\* Configure credentials in a .env file

\* Run the backend server using FastAPI

\* Launch the frontend via Streamlit

\* Upload learning materials and start interacting

5. Running the Application:

Start the FastAPI server, launch the Streamlit interface, upload content, and use the features in real time to access educational tools and resources.

6. Folder Structure:

app/ – Backend logic (chat, summarization, forecasting, etc.)

ui/ – Student & teacher dashboards (Streamlit pages)

document\_embedder.py – Converts study materials into searchable embeddings

performance\_forecaster.py – Predicts exam readiness

report\_generator.py – Creates personalized study reports

7. API Documentation:

Available endpoints include querying educational content, uploading documents, fetching summaries, and providing feedback—all documented via Swagger UI for easy testing and development.

8. Authentication:

The system supports token-based authentication, OAuth2, and role-based access controls to ensure secure usage. Future enhancements will include user session tracking and history logs.

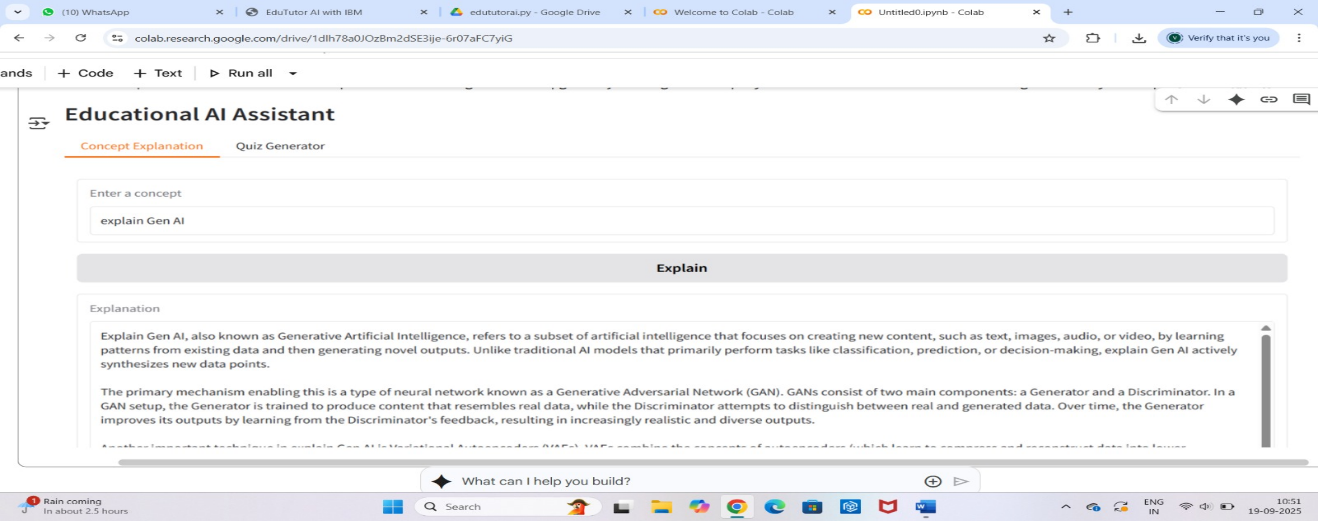
9. User Interface:

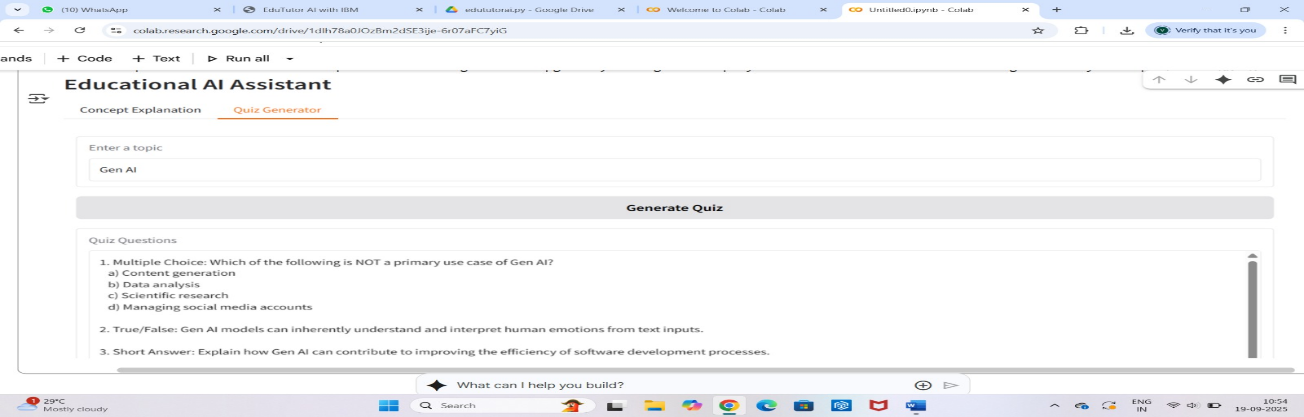
A minimalist and accessible interface designed for non-technical users, with navigation panels, real-time interaction, data visualizations, and intuitive layouts.

10. Testing:

Testing includes unit tests for functions, API validation through Swagger and Postman, and manual reviews to ensure consistency, accuracy, and robustness.

11.Screenshots:





12.Known Issues:

✓Login Issue – Sometimes student/teacher login does not work properly.

✓ API Delay – Responses from the AI assistant may take extra time.

✓File Upload Limit – Large PDF/CSV uploads may fail.

✓Mobile Compatibility – UI does not display correctly on some mobile devices.

✓ Summary Accuracy – Long documents may lose key points in summarization.

13. Future Enhancements:

Plans include expanding AI capabilities, integrating more learning analytics, and improving accessibility features for diverse user groups.