EDU TUTOR AI:
PERSONALIZED LEARNING WITH GENERATIVE AI AND LMS INTEGRATION

1. Introduction

- Team member: Sharmila M
- Team member: Sharmila J
- Team member: Srimathi D
- Team member: Sumithra R

2. Project overview

- **Purpose :**

The purpose of this project is to develop an intelligent learning system that delivers personalised educational content to students by combining the power of Generative AI with Learning Management System (LMS) integration. It aims to address the limitations of traditional one-size-fits-all learning approaches by adapting content, assessments, and feedback to suit each learner's individual needs, pace, and performance.

By integrating with existing LMS platforms, the system ensures seamless data flow and provides educators with real-time insights, ultimately enhancing learning outcomes, saving teacher time, and making education more engaging, accessible, and effective for all learners.

- **Features:**

Introduction

Key Points: Traditional LMS lacks personalisation. Generative AI adapts learning based on student needs. Enhances digital education using automation.

Functionality: Introduces the role of AI in modern learning. Explains how AI integrates with LMS for better outcomes.

Problem Statement

Key Points: Static content limits student engagement. Teachers spend time on repetitive tasks. No adaptive feedback in current systems.

Functionality: Identifies the gap in current LMS. Sets the foundation for AI-powered improvements.

Objectives

Key Points: Deliver personalised content and quizzes. Automate content generation and assessment. Track and improve student performance.

Functionality: Defines system goals like adaptation, integration, and analytics. Aligns AI and LMS features with learning outcomes

System Modules

Key Points: Student, Teacher, and Admin modules. Role-based access and functionality. Each module has a unique dashboard.

Functionality: Students receive personalised content.

3. Architecture

Frontend (Stream lit):

User Interface (UI Layer): Students access personalised lessons, quizzes, and feedback. Teachers access dashboards to track learner progress and generate AI-based content. Built using HTML/CSS/JavaScript or React.

LMS Integration Layer: Connects to LMS platforms (e.g., Moodle, Google Classroom) via APIs. Fetches user data like grades, attendance, and completed modules. Updates LMS with new scores and content.

AI Engine (Generative AI Layer): Uses OpenAI GPT or similar models to Generate custom lessons and summaries. Create adaptive quizzes and explanations. Provide intelligent feedback.

Backend Layer: Manages requests between UI, AI, and LMS. Handles authentication, user roles, and session management. Built using Python (Flask/Django) or Node.js.

Database Layer:

1. Student logs in via LMS → Data sent to AI Engine

2. AI generates content/quizzes → Sent to student via UI

3. Student interacts with content → Results sent back to LMS

4. Teacher views progress → Receives AI suggestions

4. Setup Instructions

Prerequisites:

o Python 3.9 or later

o pip and virtual environment tools

o API keys for IBM Watsonx and Pinecone

o Internet access to access cloud services

Installation Process:

o Clone the repository

o Install dependencies from requirements.txt

o Create a .env file and configure credentials

o Run the backend server using Fast API

o Launch the frontend via Stream lit

o Upload data and interact with the modules

5. Folder Structure

app/ – Contains all Fast API backend logic including routers, models, and integration modules.

app/api/ – Subdirectory for modular API routes like chat, feedback, report, and document vectorization.

ui/ – Contains frontend components for Stream lit pages, card layouts, and form UIs.

smart_dashboard.py – Entry script for launching the main Stream lit dashboard.

granite_llm.py – Handles all communication with IBM Watsonx Granite model including summarization and chat.

document_embedder.py – Converts documents to embeddings and stores in Pinecone.

kpi_file_forecaster.py – Forecasts future energy/water trends using regression.

anomaly_file_checker.py – Flags unusual values in uploaded KPI data.

report_generator.py – Constructs AI-generated sustainability reports.

6. Running the Application

To start the project:

➢ Launch the FastAPI server to expose backend endpoints.

➢ Run the Streamlit dashboard to access the web interface.

➢ Navigate through pages via the sidebar.

➢ Upload documents or CSVs, interact with the chat assistant, and view outputs like reports, summaries, and predictions.

➢ All interactions are real-time and use backend APIs to dynamically update the frontend.

Frontend (Stream lit):

The frontend is built with Stream lit, offering an interactive web UI with multiple pages including dashboards, file uploads, chat interface, feedback forms, and report viewers. Navigation is handled through a sidebar using the stream lit-option-menu library. Each page is modularized for scalability.

Backend (Fast API):

Fast API serves as the backend REST framework that powers API endpoints for document processing, chat interactions, eco tip generation, report creation, and vector embedding. It is optimized for asynchronous performance and easy Swagger integration.

7. API Documentation

Backend APIs available include: POST /chat/ask – Accepts a user query and responds with an AI-generated message

POST /upload-doc – Uploads and embeds documents in Pinecone

GET /search-docs – Returns semantically similar policies to the input query

GET /get-eco-tips – Provides sustainability tips for selected topics like energy, water, or waste

POST /submit-feedback – Stores citizen feedback for later review or analytics Each endpoint is tested and documented in Swagger UI for quick inspection and trial during development.

8. Authentication

Each endpoint is tested and documented in Swagger UI for quick inspection and trial during development.

This version of the project runs in an open environment for demonstration.

However, secure deployments can integrate:

• Token-based authentication (JWT or API keys)

• OAuth2 with IBM Cloud credentials

• Role-based access (admin, citizen, researcher)

- Planned enhancements include user sessions and history tracking.8. Authentication

9. User Interface

The interface is minimalist and functional, focusing on accessibility for non-technical users. It includes:

- Sidebar with navigation
- KPI visualizations with summary cards
- Tabbed layouts for chat, eco tips, and forecasting Real-time form handling
- PDF report download capability

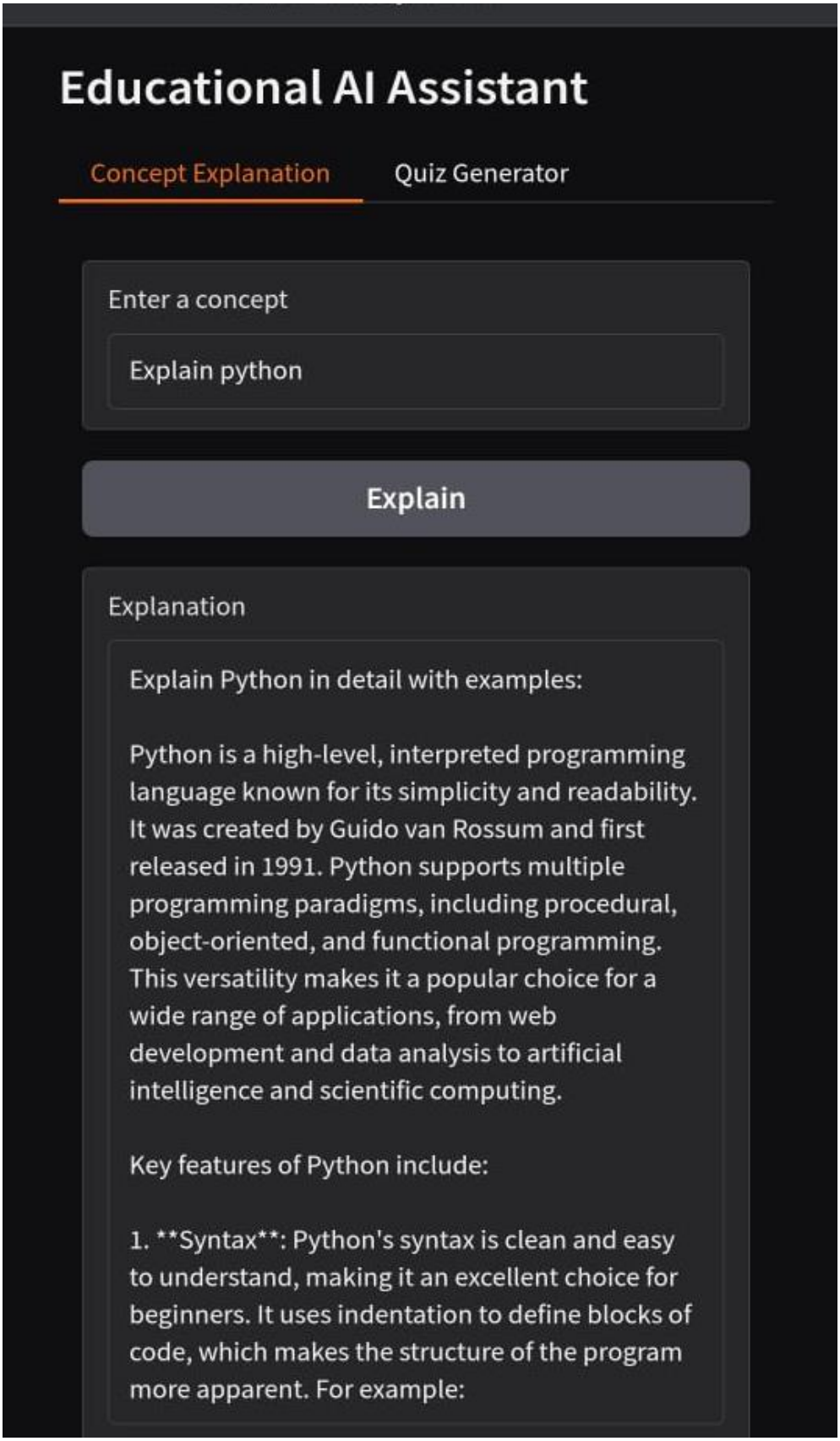
The design prioritizes clarity, speed, and user guidance with help texts and intuitive flows.

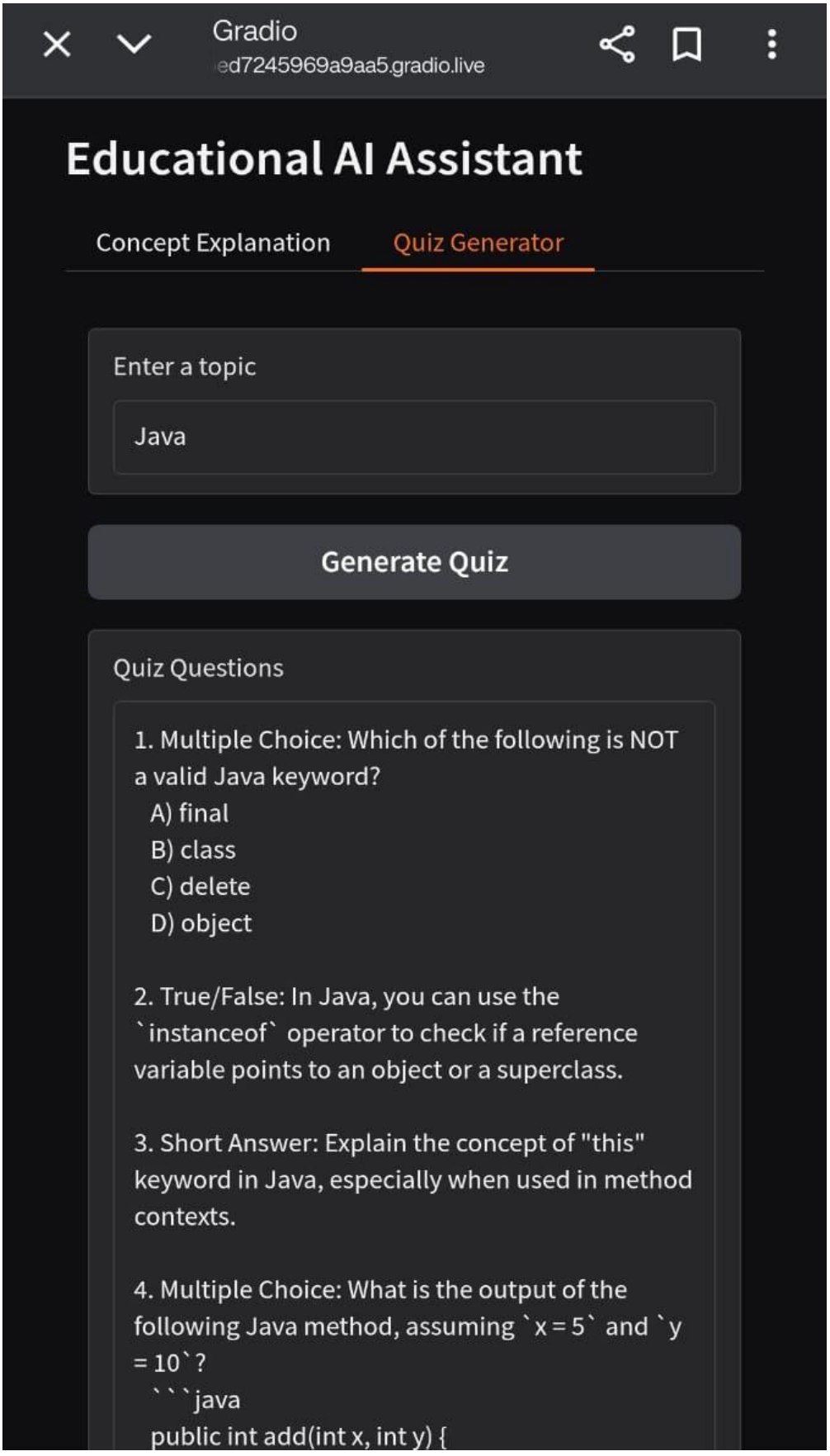
10. Testing

Testing was done in multiple phases:

- Unit Testing: For prompt engineering functions and utility
- API Testing: Via Swagger UI, Postman, and test scripts
- Manual Testing: For file uploads, chat responses, and output consistency
- Edge Case Handling: Malformed inputs, large files, invalid API keys Each function was validated to ensure reliability in both offline and API- connected modes

11. Screen shots




Gradio
ed7245969a9aa5.gradio.live
Educational AI Assistant
Concept Explanation
Quiz Generator
Enter a topic
Java
Generate Quiz
Quiz Questions
1. Multiple Choice: Which of the following is NOT a valid Java keyword?
A) final
B) class
C) delete
D) object
2. True/False: In Java, you can use the `instanceof` operator to check if a reference variable points to an object or a superclass.
3. Short Answer: Explain the concept of "this" keyword in Java, especially when used in method contexts.
4. Multiple Choice: What is the output of the following Java method, assuming `x = 5` and `y = 10`?
```java
public int add(int x, int y) {
```

12. Known Issues

Data Privacy and Security Concerns

Integrating AI models with Learning Management Systems (LMS) necessitates handling sensitive student data. Ensuring compliance with data protection regulations and maintaining robust security measures are critical to prevent unauthorized access and misuse of personal information.

Model Accuracy and Hallucination Risks

While generative AI models can produce coherent responses, they may sometimes generate inaccurate or misleading information, a phenomenon known as "hallucination." This poses risks in educational settings where factual correctness is paramount. Implementing mechanisms like Dynamic Course Content Integration (DCCI) can help mitigate such issues by structuring retrieved content within the AI's context window to ensure accuracy and relevance .

13. Future enhancement

Enhanced Personalization Through Dynamic Content Integration

The Dynamic Course Content Integration (DCCI) mechanism is being refined to seamlessly connect EduTutor AI with LMS platforms like Canvas. By structuring retrieved content within the AI's context window, DCCI ensures that responses are accurate, relevant, and aligned with the curriculum, thereby improving student engagement and comprehension.

Ethical Safeguards and Bias Mitigation

Future developments will focus on implementing robust ethical safeguards and bias mitigation strategies. This includes enhancing guardrail models to prevent the generation of harmful or biased content, ensuring that EduTutor AI provides equitable and trustworthy educational support.

Open-Source Collaboration and Community Engagement

IBM's commitment to open-source development continues with the release of models under the Apache 2.0 license. This approach encourages community collaboration, allowing educators and developers to contribute to the evolution of EduTutor AI, ensuring it meets the diverse needs of learners worldwide.