EduTutor AI: LMS Integration Project Documentation

---

1. Introduction

1.1 Overview EduTutor AI is a next-generation personalized education platform aimed at transforming the traditional academic experience for both students and educators. It leverages state-of-the-art artificial intelligence—specifically IBM Watsonx's Granite models—to dynamically generate quizzes, offer real-time feedback, and provide adaptive content recommendations. With its seamless integration into Google Classroom, EduTutor AI aligns directly with classroom curricula and automates instructional support.

The system's core functionality includes AI-powered quiz generation based on selected topics and user performance history. It evaluates submitted answers instantly and delivers intelligent feedback tailored to the student's knowledge level. For educators, EduTutor AI presents a comprehensive analytics dashboard that visualizes class performance trends, individual scores, quiz histories, and learning progress over time. By integrating Pinecone's vector database, it enables efficient similarity searches and maintains rich historical metadata to personalize future assessments.

EduTutor AI addresses the gap between rigid classroom education and flexible personalized learning by providing a scalable, modular, and intuitive solution accessible through a browser-based interface built using Streamlit. Its goal is to foster better academic outcomes by adapting to learners' unique needs while saving educators time and effort through automation and actionable insights.

1.2 Vision and Goals

- Deliver adaptive and customized learning experiences.
- Automate the quiz and assessment process using AI.
- Provide educators with performance tracking tools and class-wide analytics.
- Ensure seamless integration with existing LMS platforms like Google Classroom.
- Foster increased student engagement, self-awareness, and motivation through tailored feedback.
- Create a modular, scalable, and theme-customizable interface.

---

2. Problem Statement & Motivation

2.1 Challenges in Traditional Learning The current educational ecosystem—especially in online and blended learning environments—often lacks personalization, adaptability, and real-time feedback. Students are frequently subjected to a one-size-fits-all instructional model, which ignores individual learning paces, preferences, and competency gaps. Moreover, educators are burdened with repetitive administrative tasks, such as manual quiz creation, performance tracking, and student progress evaluation, reducing the time available for student engagement and mentorship.

Student-Specific Challenges:

- Lack of interactive learning tools, resulting in reduced engagement.
- Inability to pinpoint areas of weakness due to generic assessments.

- Absence of real-time feedback that could drive timely improvements.
- Limited visibility into their learning trajectory and no personalized path to follow.

Educator-Specific Challenges:

- Manually creating tailored quizzes for each student is time-consuming.
- Difficulties in monitoring individual and class performance trends.
- Lack of automation in syncing assignments and content with LMS platforms.
- No centralized system that provides a holistic view of learning analytics.

2.2 Educational System Limitations Educational institutions often use fragmented tools that do not communicate effectively with each other. Google Classroom, for instance, is widely used but lacks built-in AI capabilities for adaptive testing. Similarly, existing AI models are either too generic or require expensive infrastructure. Students and educators alike lack access to a unified, cost-effective system that integrates AI personalization, curriculum data, and user-friendly analytics.

2.3 Motivation Behind EduTutor AI EduTutor AI is conceived as a unified platform that brings together:

- **AI-powered customization:** Generate quizzes and feedback based on learner history.
- **Seamless LMS integration:** Sync data from Google Classroom to align AI content with course material.
- **Streamlined educator experience:** Provide dashboards that automatically visualize performance trends.
- **Engagement tools:** Use themes, task reminders, notifications, and interactive elements to motivate users.

By resolving inefficiencies in both learning and instruction, EduTutor AI enhances educational quality, accessibility, and equity. It is designed for scalability in school, college, and professional training environments, aiming to become a core tool in modern education.

2.4 Empathy-Driven Design EduTutor AI's features are informed by insights gathered through:

- Surveys from students who reported the need for more personalized academic help.
- Interviews with educators expressing difficulty in generating targeted assessments.
- Research studies on how real-time feedback influences learning outcomes.

2.5 Vision for Transformation By leveraging IBM Watsonx, LangChain, and Pinecone, EduTutor AI not only provides technical robustness but also envisions a future where learning is truly individualized. It's a step toward transforming the static experience of assessments into dynamic, student-centered interactions.

The core motivation behind EduTutor AI is to bridge the gap between generalized education and personalized learning using scalable, intelligent, and integrated tools.**

---

3. Objectives

The primary objectives of EduTutor AI are:

- **Personalization of Content:** Deliver custom quizzes to each student based on topic selection, prior performance, and difficulty preference.
- **Real-time Evaluation:** Provide immediate scoring and feedback to help learners identify strengths and weaknesses.
- **Educator Empowerment:** Supply teachers with actionable insights through dashboards and visual analytics.
- **Google Classroom Integration:** Automatically import class and subject data to streamline quiz alignment and enhance productivity.
- **Scalability and Modularity:** Design the system with a modular backend and customizable front end to support future expansions.
- **Improved Engagement:** Use AI-powered interactions like smart content creation and assistant panels to foster interest and continuous learning.

---

4. System Architecture

EduTutor AI follows a modular, scalable system design with integration points across artificial intelligence, data storage, API services, and user-facing interfaces. The architecture is designed to support both synchronous and asynchronous interactions, ensuring fast response times, seamless data handling, and easy extensibility. This section provides a detailed breakdown of each architectural layer, their responsibilities, and how they interact to deliver an end-to-end intelligent learning experience.

4.1 Architectural Goals

- Ensure modularity for easy updates and maintenance
- Provide high availability and low latency for quiz generation and evaluation
- Support integration with external systems such as Google Classroom
- Enable scalability through microservices and cloud hosting compatibility
- Use secure and efficient communication between frontend and backend layers

4.2 Layered Architecture

1. **UI Layer – Streamlit Frontend**

2. Designed for intuitive user interaction, it offers dynamic dashboards and role-based interfaces.

3. Provides access to quiz generation, progress tracking, theme customization, and real-time notifications.

4. Uses Plotly for interactive data visualizations and integrates UI state using `st.session_state`.

5. **Application Layer – FastAPI Backend**

6. Manages business logic for login, quiz generation, answer evaluation, and metadata updates.
7. Offers fast, asynchronous API endpoints for responsive frontend communication.
8. Acts as a middle layer connecting UI to AI models and data services.
9. **AI Integration Layer – Watsonx + LangChain**
10. IBM Watsonx Granite LLMs are integrated through LangChain for context-aware quiz generation.
11. Prompt templates are used dynamically to generate subject-relevant MCQs.
12. AI responses are parsed into structured formats (JSON) for quiz delivery and evaluation.
13. **Data Layer – SQLite and Pinecone Vector DB**
14. **SQLite** manages structured data such as user credentials, quiz logs, and feedback.
15. **Pinecone** stores vector embeddings of user quiz histories for personalized learning suggestions.
16. Supports metadata tagging to enable rapid similarity searches and context adaptation.
17. **Integration Layer – Google Classroom API**
18. Syncs course lists, subjects, and assignment data for both students and educators.
19. Uses `google-auth-oauthlib` for OAuth login and authorization.
20. Enables consistent alignment with school curricula.

4.3 Component Communication Flow

- A student or educator interacts via the Streamlit frontend.
- Requests are routed to FastAPI, which determines whether to fetch data, sync classes, or invoke AI.
- AI queries are processed through LangChain and Watsonx, and the response is relayed to the UI.
- All user activity and performance metrics are logged in SQLite and Pinecone.

4.4 Data Handling Strategy

- **Authentication:** OAuth tokens and manual login sessions are stored securely.
- **Quiz Metadata:** Includes topic, score, difficulty, timestamp, and feedback.
- **Embedding Vectors:** Enable future quiz generation that adapts based on similarity and historical data.

4.5 Scalability Considerations

- Designed to support containerized deployment (Docker-ready architecture).
- Can migrate from SQLite to PostgreSQL or other scalable DBMS as user base grows.
- Pinecone’s horizontal scalability ensures vector search remains efficient with larger datasets.

- Stateless backend endpoints allow for load balancing and parallel processing.

4.6 Security Measures

- All sensitive data (API keys, OAuth credentials) is stored in `.env` using `python-dotenv`.
- Backend implements validation and sanitization for user input.
- Google OAuth ensures encrypted, secure user login and authorization.

4.7 Diagram Description (for visual placement) *A recommended diagram to visualize:*

- **Frontend (Streamlit UI)** → **Backend (FastAPI)** → **AI Model (LangChain + Watsonx)** → **Storage (SQLite, Pinecone)**
- External sync: **Google Classroom API** interacts with Backend layer for user and class data

Together, these layers ensure that EduTutor AI remains robust, fast, and adaptable to different educational settings, providing a future-ready infrastructure for AI-assisted learning.

---

5. User Flow

The user experience within EduTutor AI is streamlined, intuitive, and role-specific. Each user—student or educator—interacts with the platform in a manner tailored to their goals. This section outlines detailed user workflows, highlighting each stage and system response for common actions.

5.1 Student Workflow

1. **Authentication:** The student accesses the platform via manual login or Google OAuth. Once verified, the homepage displays a personalized greeting (face card) and task notifications.
2. **Classroom Sync:** If using Google login, the system syncs classroom and subject data automatically from Google Classroom.
3. **Quiz Request:** Students can select a subject, topic, and difficulty level or use the "AI Generated Quiz" panel to type a natural-language prompt (e.g., "Give me a Python basics quiz").
4. **Quiz Generation:** The FastAPI backend invokes Watsonx Granite LLM via LangChain, which dynamically generates a set of MCQs based on context.
5. **Quiz Submission:** Students answer the questions in the frontend and submit. Answers are validated and scored instantly by the backend.
6. **Feedback Delivery:** The platform evaluates responses and returns adaptive feedback, categorizing performance (e.g., Beginner, Intermediate, Advanced) and offering tips.
7. **Progress Dashboard:** Results are stored in Pinecone and visualized using interactive charts. Students can view history, review previous quizzes, and compare performance over time.
8. **Theme and Customization:** Students can toggle between themes (dark/light/purple) to personalize their interface.
9. **Task Tracker:** The system notifies students of new quizzes, classwork, or reminders synced from Google Classroom.

5.2 Educator Workflow

1. **Authentication:** Educators log in through a similar process, gaining access to a role-specific dashboard.
2. **Classroom Sync:** Upon login, subjects and student lists are synced with Google Classroom.
3. **Performance Overview:** Educators view a dashboard summarizing quiz participation, scores, and topics attempted.
4. **Student-Level Detail:** Teachers can click into a student profile to see quiz history, diagnostic test results, and progress over time.
5. **Quiz Oversight:** While quizzes are auto-generated, educators can trigger topic-specific quizzes or assist in quiz customization.
6. **AI Assistant:** Educators can use the assistant to generate concept explanations, summaries, or even assignments.
7. **Notifications & Alerts:** Educators receive updates when students submit assessments, perform poorly, or miss deadlines.
8. **Insights Reporting:** Downloadable analytics reports help in lesson planning and performance tracking.

5.3 Additional UX Flows and Features

- **Quiz Activity Logs:** Students and educators can click on any past quiz attempt to view answers, feedback, and scores.
- **Interactive AI Assistant:** Located on the homepage, responds to input prompts with contextual educational material.
- **Smart Content Panel:** Allows users to generate learning material (e.g., explanations, summaries) using AI.
- **Multitheme Selector:** Dropdown for switching between visual themes, stored in session state.
- **Notification Bell:** Lists classroom tasks, new assignments, or recent activities from Google Classroom.

This rich user journey ensures that both students and educators can engage with EduTutor AI in a way that enhances productivity, simplifies learning workflows, and personalizes the academic experience.

---

6. Features

- **Dynamic Quiz Generation:** Create MCQs from selected topics and difficulty using Watsonx + Granite.
- **Adaptive Feedback Engine:** Instantly scores and evaluates answers with tailored recommendations.
- **Google Classroom Integration:** Sync subjects, classes, and assignments using API.
- **Real-Time Dashboards:** Student and educator dashboards display historical data, charts, and performance insights.
- **Multi-theme UI:** Switch between light, dark, and color-themed interfaces (e.g., Purple Sunset).
- **AI Assistant & Smart Content Creator:** Generate explanations or summaries interactively.
- **Daily Tasks & Notifications:** View classroom activities and reminders.
- **Subject Expansion:** Supports computer languages, engineering domains, history, physics, algebra.
- **Quiz Activity:** Clickable entries show past attempts and feedback.

---

7. Functional Modules

The architecture of EduTutor AI is composed of multiple interrelated modules that work cohesively to offer a complete educational platform. Each functional module is responsible for delivering specific services that contribute to the platform's personalized learning experience. Below is a comprehensive description of each module, its functionality, and how it contributes to system operations.

7.1 Authentication Module

- Supports two types of logins: manual (username/password) and Google OAuth.
- Uses `google-auth-oauthlib` for secure Google login integration.
- Manages session handling, authentication tokens, and redirection logic.
- Ensures role-based access by determining whether the user is a student or educator.

7.2 Quiz Engine Module

- Receives input from the user (topic, difficulty, subject).
- Uses LangChain to send prompts to IBM Watsonx Granite LLMs.
- Processes AI-generated output into structured MCQ formats (JSON).
- Manages quiz timing, question shuffling, and session-based delivery.

7.3 Evaluation Module

- Parses student responses and compares them against correct answers.
- Calculates scores and assigns performance categories (Beginner, Intermediate, Advanced).
- Provides adaptive feedback including suggestions for improvement.
- Updates student progress records in both SQLite and Pinecone.

7.4 Google Sync Module

- Interacts with Google Classroom API to import classroom metadata.
- Syncs student enrollment lists, subject names, and assignment data.
- Automates quiz alignment with curriculum topics.
- Keeps platform content updated with Google Classroom changes.

7.5 Dashboard Module

- Delivers separate dashboards for students and educators.
- Displays user-specific data such as scores, quiz history, and insights.
- For educators, it offers analytics charts, group performance, and progress tracking.
- Utilizes Plotly for generating visually rich graphs.

7.6 AI Assistant Module

- Allows students or educators to input text-based queries.
- Uses LangChain + Watsonx to generate educational content (summaries, definitions, explanations).
- Interfaces with the Smart Content Panel on the homepage.
- Enhances engagement and supports self-directed learning.

7.7 Theme and Layout Manager

• Enables real-time theme switching (dark, light, and color themes like "Purple Sunset").
• Stores selected theme in `st.session_state` for persistent experience.
• Manages dashboard layout visibility based on user role and preferences.

7.8 Notification Module

• Displays alerts for daily tasks, pending quizzes, or new assignments.
• Integrates with Google Classroom to fetch class-specific tasks.
• Includes a notification bell icon on the homepage for real-time updates.
• Sends periodic reminders to enhance learner consistency.

7.9 Quiz Activity & History Module

• Logs every quiz attempt with timestamp, score, and performance level.
• Allows users to review completed quizzes and compare past results.
• Helps educators track quiz submission rates and common problem areas.

Each module in EduTutor AI has been developed to be modular, reusable, and extendable—supporting both immediate needs and future scalability. These modules collectively contribute to a rich, interactive, and intelligent educational ecosystem.** Alerts for upcoming quizzes, tasks, and classroom updates.

---

8. Technology Stack

EduTutor AI is constructed using a modern, open-source technology stack that supports rapid development, secure operations, and seamless AI integration. The stack is carefully chosen to optimize functionality across multiple layers, including the frontend, backend, AI processing, database management, and third-party integration. Below is a detailed breakdown of each component in the technology stack.

Layer	Technology	Description
Frontend	Streamlit	Used to build the interactive user interface including dashboards, quiz pages, and AI assistant panels. It enables rapid prototyping with Python and supports reactive layouts.
Backend	FastAPI	Serves as the main REST API backend to handle all business logic, quiz generation requests, evaluation logic, and metadata management. Known for high performance with asynchronous support.
AI Engine	IBM Watsonx + LangChain	The core AI stack responsible for generating dynamic quizzes. LangChain acts as a wrapper to communicate prompts to IBM Watsonx's Granite model, which processes content and returns structured output.

Layer	Technology	Description
Database	SQLite	Lightweight relational database that stores user credentials, performance data, quiz logs, and configuration metadata. Used for development and prototyping.
Vector DB	Pinecone	Manages vector embeddings of user history and quiz metadata to enable fast similarity search and adaptive content recommendations.
Authentication	google-auth-oauthlib + manual login	Provides secure authentication mechanisms via Google OAuth and fallback manual login for users without a Google account.
Visualization	Plotly	Used within Streamlit to render interactive graphs and analytics charts for both students and educators.
Env Manager	python-dotenv	Handles secure storage of API credentials, secret keys, and environment-specific configuration data loaded from `.env` files.

Design Philosophy:

- Use minimal dependencies and standard libraries to ensure easier deployment and low overhead.
- Select scalable and secure services that can transition to production-ready alternatives like PostgreSQL and Dockerized deployments.
- Focus on open-source tools to maintain transparency and community support.

This technology stack ensures that EduTutor AI remains highly extensible, developer-friendly, and compatible with both cloud and local environments.

---

13. Appendix

The appendix provides supplementary materials to support technical understanding, implementation, and customization of EduTutor AI. This includes sample code snippets, configuration references, and development tips.

13.1 Sample Quiz Evaluation Logic

```
from ai_quiz import AIQuizGenerator

quiz_engine = AIQuizGenerator()
questions = quiz_engine.generate_quiz("Python", difficulty_level=2,
subject="Computer Science")

user_answers = [0, 2, 1, 1, 3]
feedback = quiz_engine.evaluate_answers(questions, user_answers)
```

```
print(feedback["performance_level"])
```

This example shows how quizzes are generated and evaluated using the AIQuizGenerator module.

13.2 LangChain Prompt Template Example

```
"""
Generate a 5-question multiple choice quiz on the topic of {topic}. Each
question should include four options and one correct answer.
Format the output as a JSON object:
{
  "questions": [
    {
      "question": "...",
      "options": ["A", "B", "C", "D"],
      "answer": 1
    },
    ...
  ]
}
"""
```

13.3 .env File Structure Reference

```
WATSONX_MODEL_ID=granite-13b-instruct-v2
WATSONX_API_KEY=your_api_key_here
WATSONX_ENDPOINT=https://us-south.ml.cloud.ibm.com
WATSONX_PROJECT_ID=your_project_id
PINECONE_API_KEY=your_pinecone_api_key
PINECONE_INDEX_NAME=edututor
google_client_id=your_google_oauth_client_id
google_client_secret=your_google_oauth_client_secret
```

13.4 Recommended Folder Structure

```
edututor-ai/
│
├── main.py                 # FastAPI backend entry
├── frontend.py             # Streamlit frontend interface
├── requirements.txt        # Python dependencies
├── .env                    # Environment variables
├── ai_quiz.py              # AI-based quiz generation logic
├── database/
```

```
│   └── edututor.db          # SQLite DB file
├── static/
│   └── styles.css           # Custom CSS for UI
└── utils/
    └── helpers.py           # Utility functions
```

This appendix is a resource for developers who want to extend, customize, or deploy EduTutor AI in their own environments or institutions.