**EduTutor AI: Personalized Learning with Generative AI and LMS Integration**

1. Introduction

1.1 Project Title

EduTutor AI: Personalized Learning with Generative AI and LMS Integration

1.2 Team details

Solo team (only me)

2. Project Overview

2.1 Purpose

EduTutor AI is an intelligent educational platform designed to simplify the quiz creation process for educators and enhance the learning experience for students using cutting-edge technologies like IBM Granite, Google Classroom APIs, and Pinecone vector database. The system enables educators to generate AI-powered multiple-choice questions, sync with Google Classroom, and manage student performance in real time.

2.2 Features

* AI-Powered Quiz Generation: Educators can create customizable MCQs on any topic using the IBM Granite language model.
* Google OAuth Simulation: Secure login system simulating Google authentication.
* Classroom Data Sync: Mocked integration with Google Classroom to fetch class and student information.
* Interactive Student Interface: Students can take quizzes, submit answers, and receive immediate feedback.
* Performance Dashboard: Educators can monitor quiz statistics and student progress.
* Mock Pinecone Integration: Simulated storage of quizzes and submissions for scalable deployment.

3. Architecture

3.1 Frontend

The frontend is implemented using Gradio , a Python-based web framework ideal for building machine learning demos and prototypes. It provides:

* Interactive forms for user login.
* Tabs for role-specific interfaces (educator & student).
* Markdown rendering for dynamic content display.
* Dropdowns, buttons, and input fields for quiz generation and submission.

3.2 Backend

The backend is written in Python and organized into a single class SimplifiedEduTutorAI that encapsulates all business logic including:

* User authentication and session management.
* AI quiz generation using Hugging Face Transformers pipeline.
* Google Classroom data synchronization.
* Submission processing and feedback generation.
* In-memory data handling for quizzes, users, and submissions.

3.3 Database

Currently, data is stored in memory using dictionaries to simulate persistent storage:

* quizzes: Stores full quiz data with correct answers.
* student\_quizzes: Stores quiz data without answers for student view.
* submissions: Tracks student responses and results.
* users: Simulates logged-in user sessions.
* classroom\_data: Holds mock Google Classroom information.

In production, this would be replaced by Pinecone DB for scalable vector storage of quiz and submission data.

4. Setup Instructions

4.1 Prerequisites

To run this project locally or in Google Colab, ensure you have the following installed:

Software Requirements

* Python 3.8+
* Jupyter Notebook or Google Colab environment
* Internet access (for downloading models and libraries)

Libraries Used

!pip install transformers torch gradio accelerate

!pip install pinecone-client google-auth google-auth-oauthlib google-auth-httplib2

!pip install google-api-python-client pandas

4.2 Installation Steps

1. Open Google Colab or a local Jupyter notebook.
2. Create a new notebook and paste the provided code.
3. Run the installation commands at the top of the script.
4. Execute the entire notebook to load the AI model and launch the Gradio interface.
5. Use the Gradio UI to interact with the system as either an educator or student.

5. Folder Structure

Since this is a single-file application built for Colab, there is no traditional folder structure. However, the logical organization within the file is as follows:

5.1 Code Organization

* Imports: All necessary libraries are imported at the beginning.
* Class Definition (SimplifiedEduTutorAI):
  + Model loading and setup
  + Authentication and session management
  + Quiz generation and parsing
  + Submission processing
  + Dashboard generation
* Interface Functions: Functions to connect Gradio components to backend logic.
* Gradio Interface: Declared using gr.Blocks() for a clean UI layout.

6. Running the Application

6.1 Local Execution

This application is primarily designed for Google Colab execution. To run it locally:

1. Ensure all dependencies are installed.
2. Run the script in a Python environment supporting PyTorch and Transformers.
3. Launch the Gradio interface via demo.launch().

6.2 Starting the App

After running the entire notebook, the Gradio interface will appear embedded in the notebook cell or open in a new browser tab if launched externally.

Commands

if \_\_name\_\_ == "\_\_main\_\_":

demo.launch(share=True, debug=True)

7. API Documentation

While this is not a RESTful API-based system, the internal methods function similarly to endpoints. Here’s a summary:

|  |  |
| --- | --- |
| simulate\_google\_login(email, role) | Authenticates a user with Google-like OAuth flow. |
| sync\_google\_classroom(user\_id) | Synchronizes mock classroom data for educators. |
| generate\_ai\_quiz(topic, num\_questions, difficulty) | Generates a quiz using IBM Granite model. |
| submit\_quiz(quiz\_id, answers) | Processes student submissions and calculates scores. |
| get\_educator\_dashboard() | Returns formatted dashboard for educators. |
| get\_student\_dashboard() | Returns formatted dashboard for students. |

8. Authentication

8.1 User Roles

There are two user roles in the system:

* Educator : Can generate quizzes and sync Google Classroom data.
* Student : Can take quizzes and review results.

8.2 Login Process

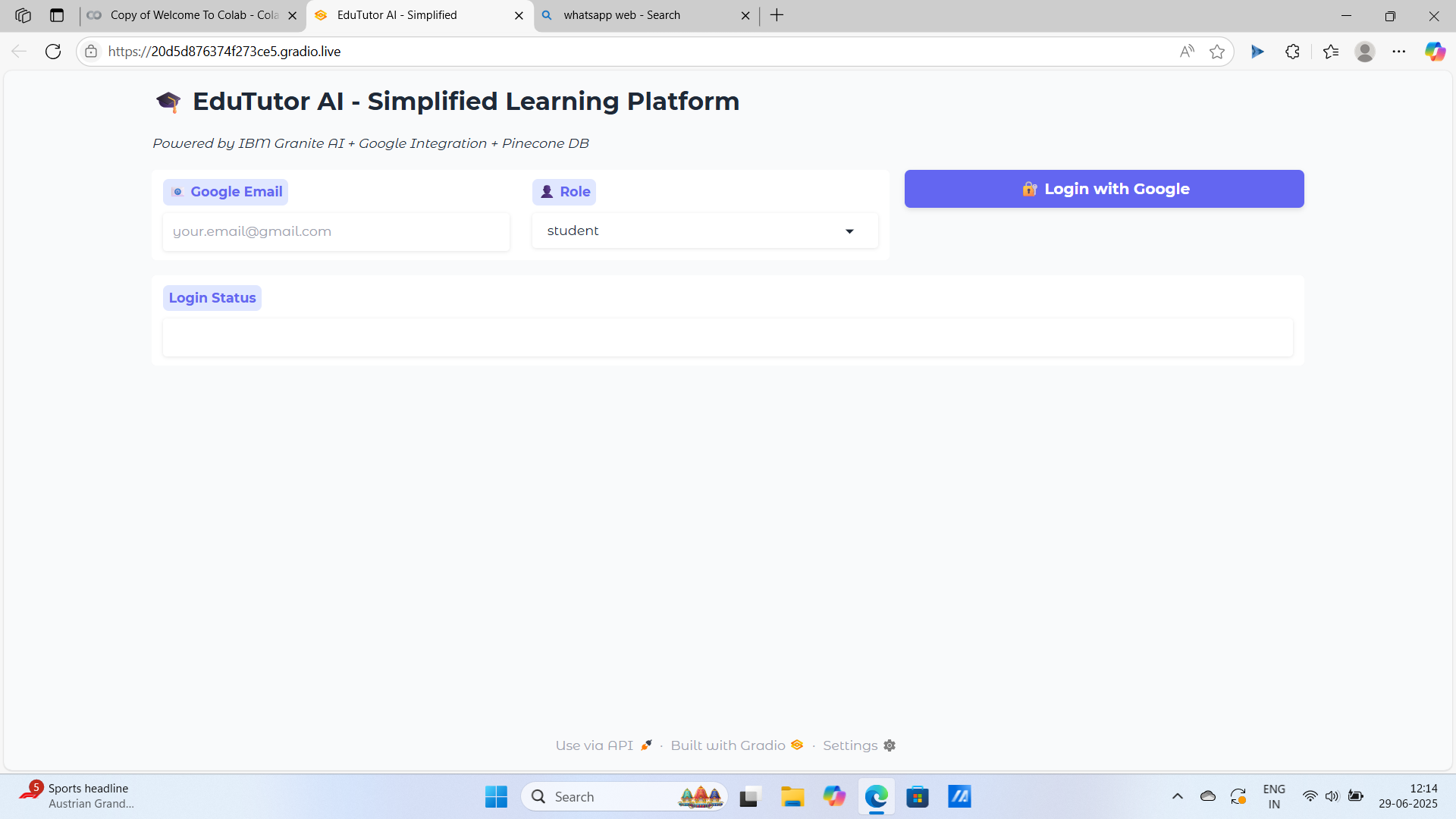
* A simulated Google OAuth login is used.
* Users provide their email and select a role.
* Upon successful login, a session is created with user details and role.

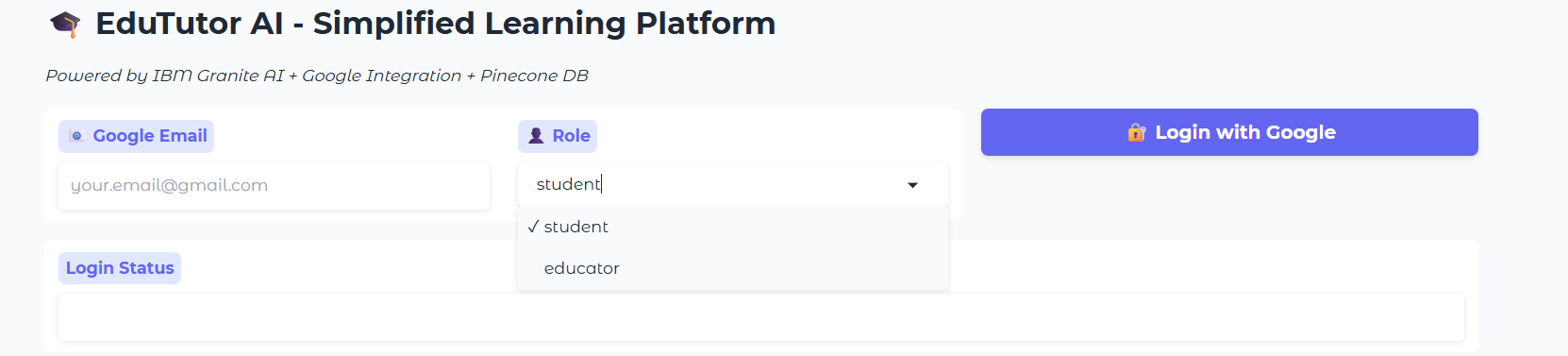
8.3 Session Management

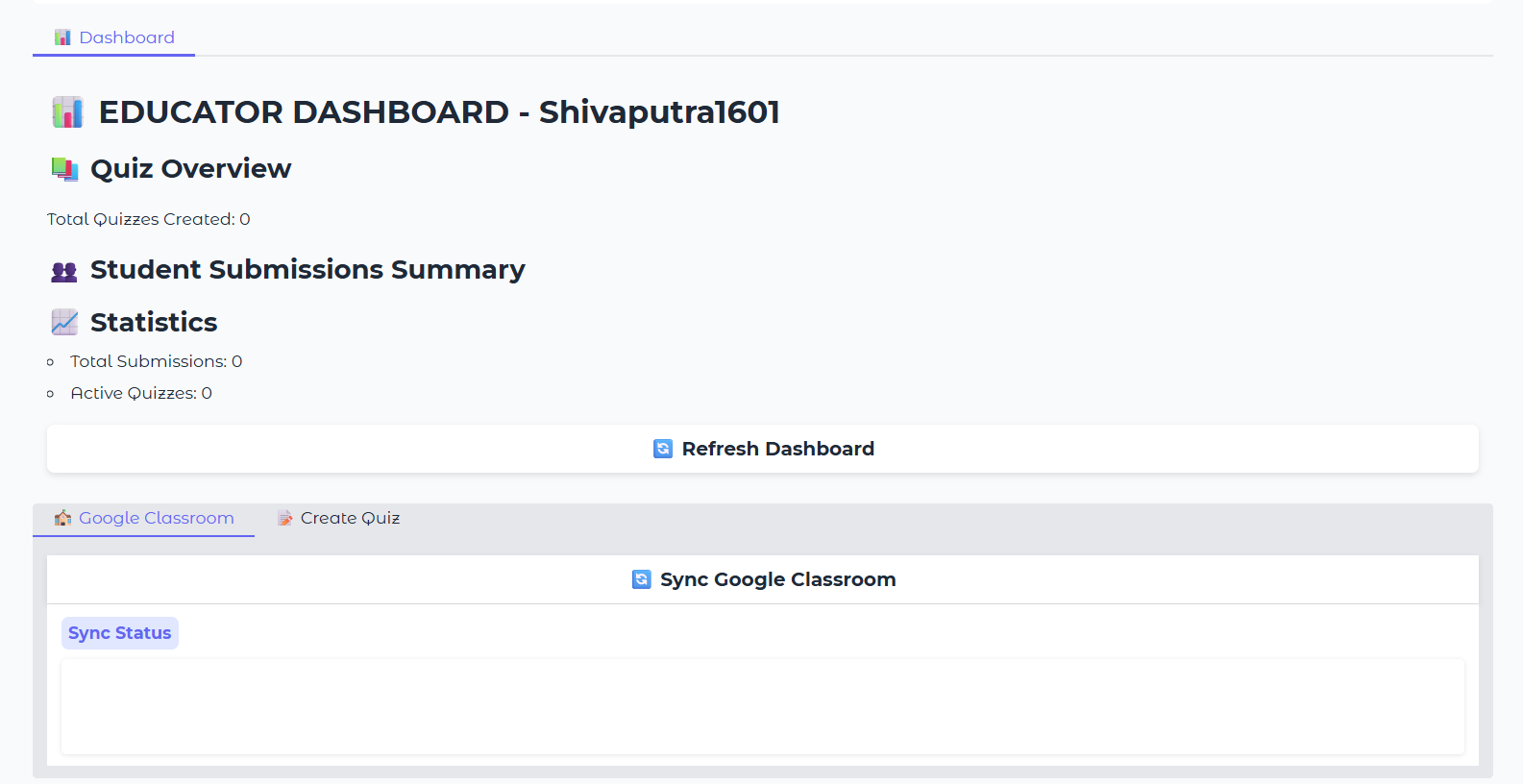
User sessions are tracked in memory using the current\_user and user\_role attributes of the SimplifiedEduTutorAI class.

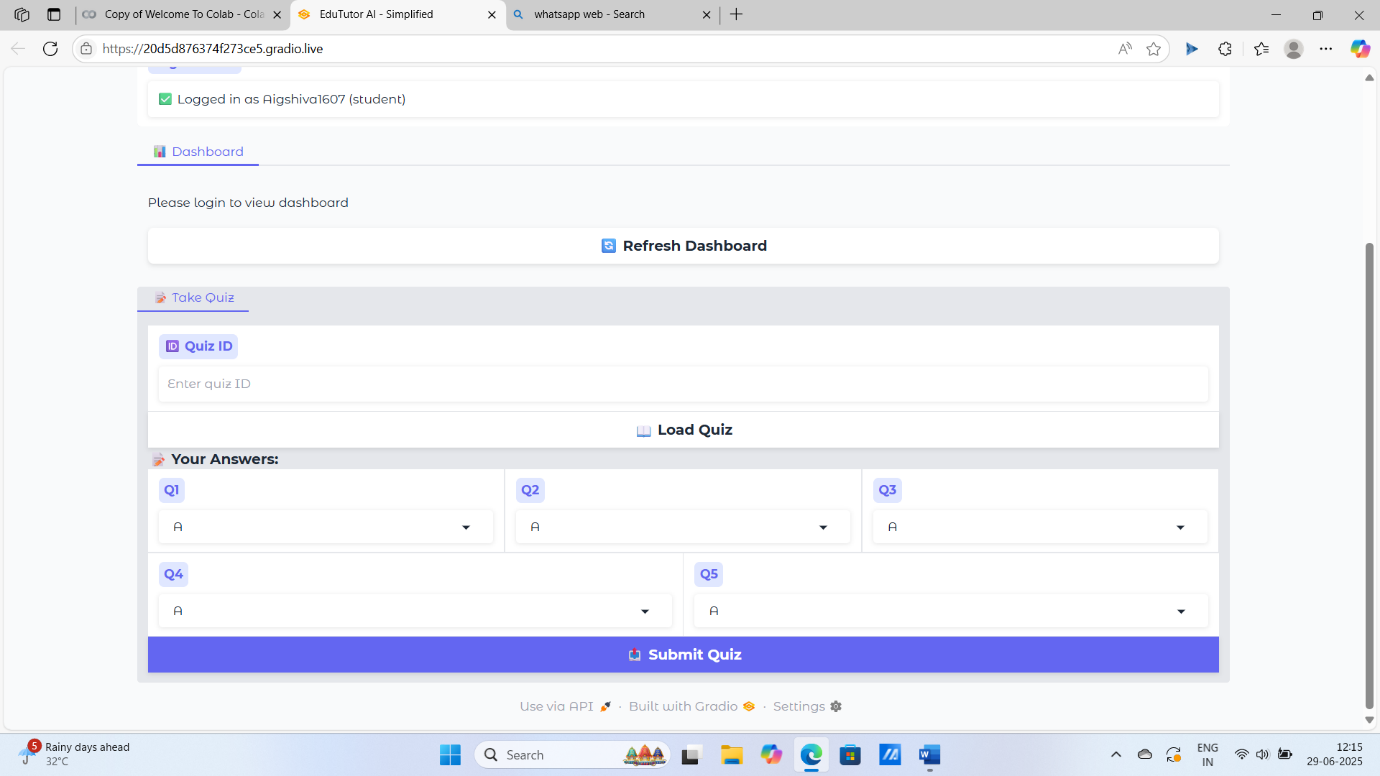
9. User Interface

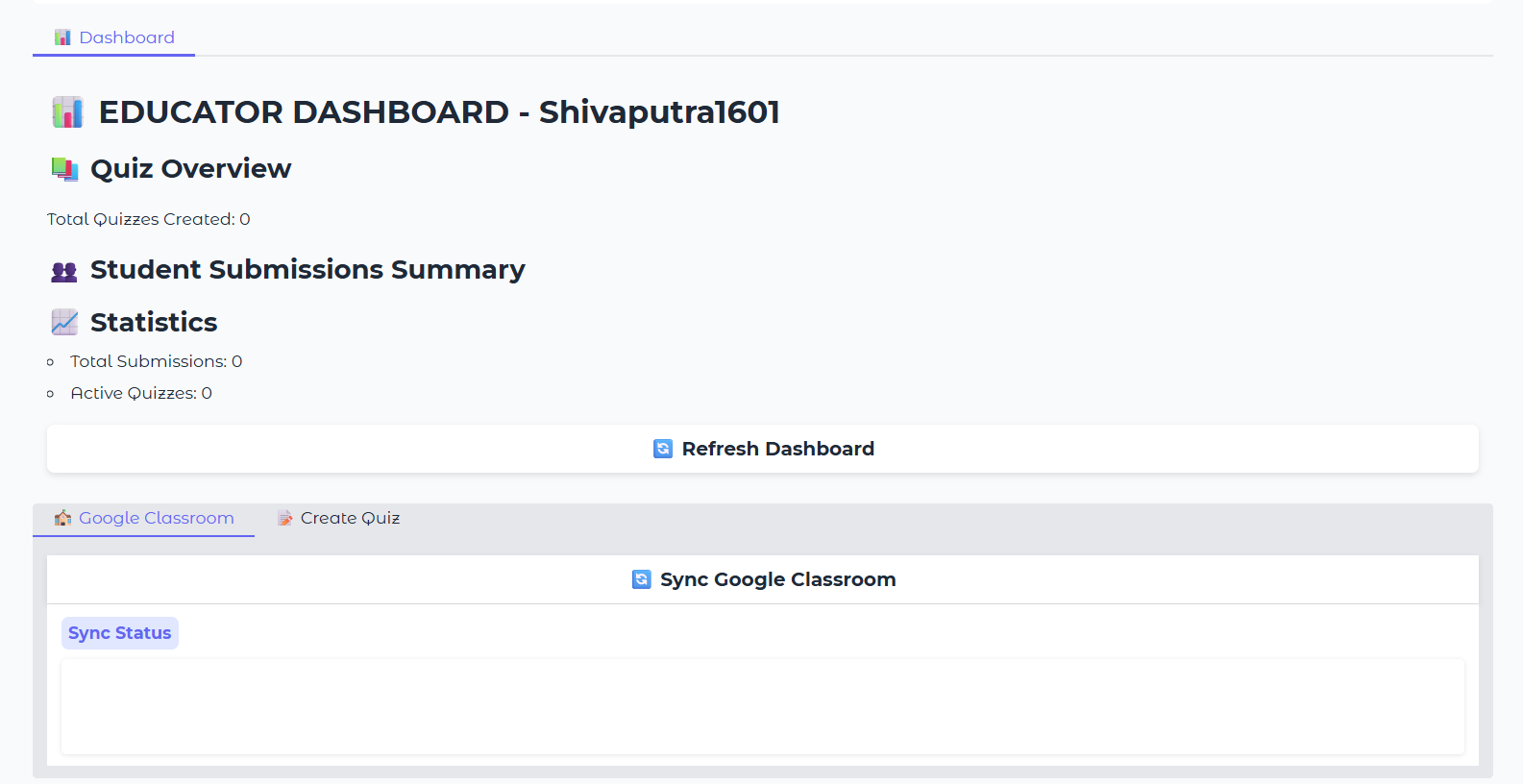
9.1 Screenshots

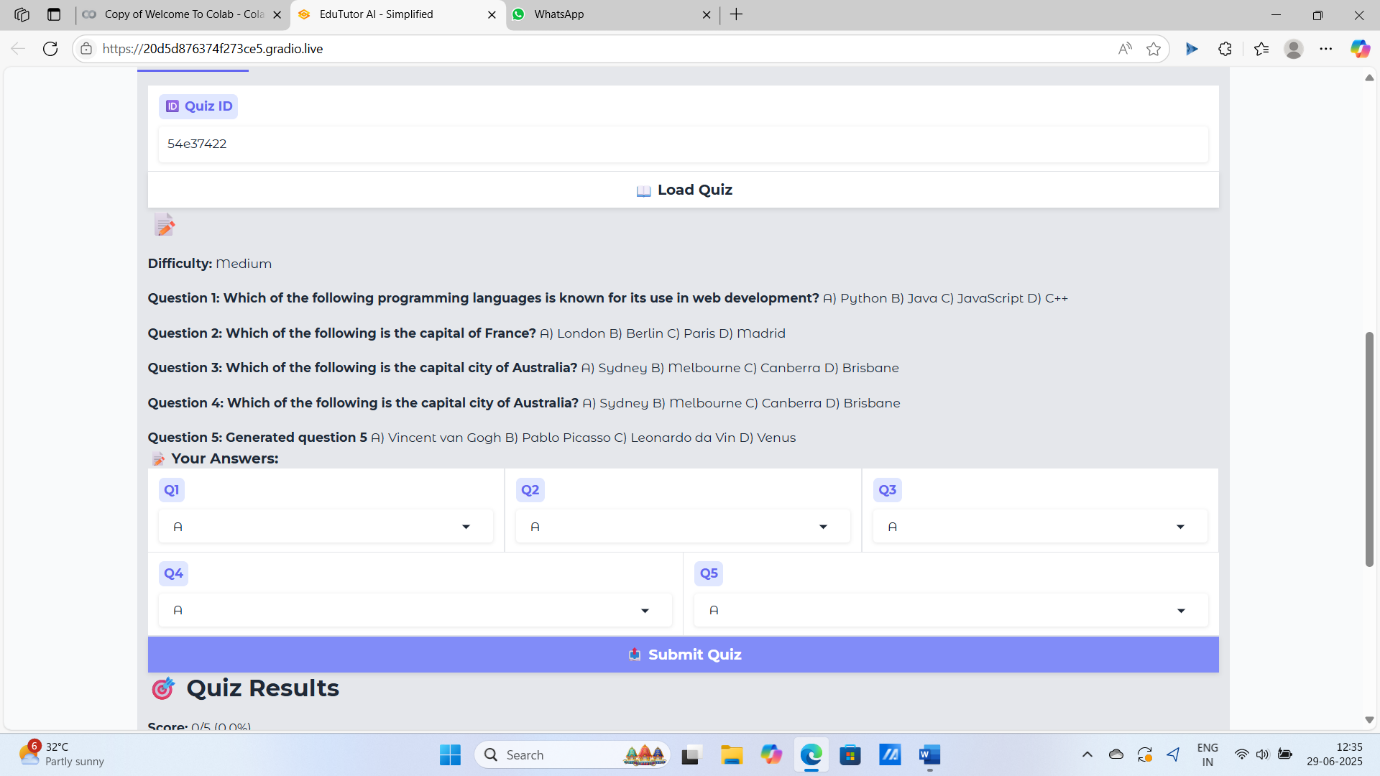


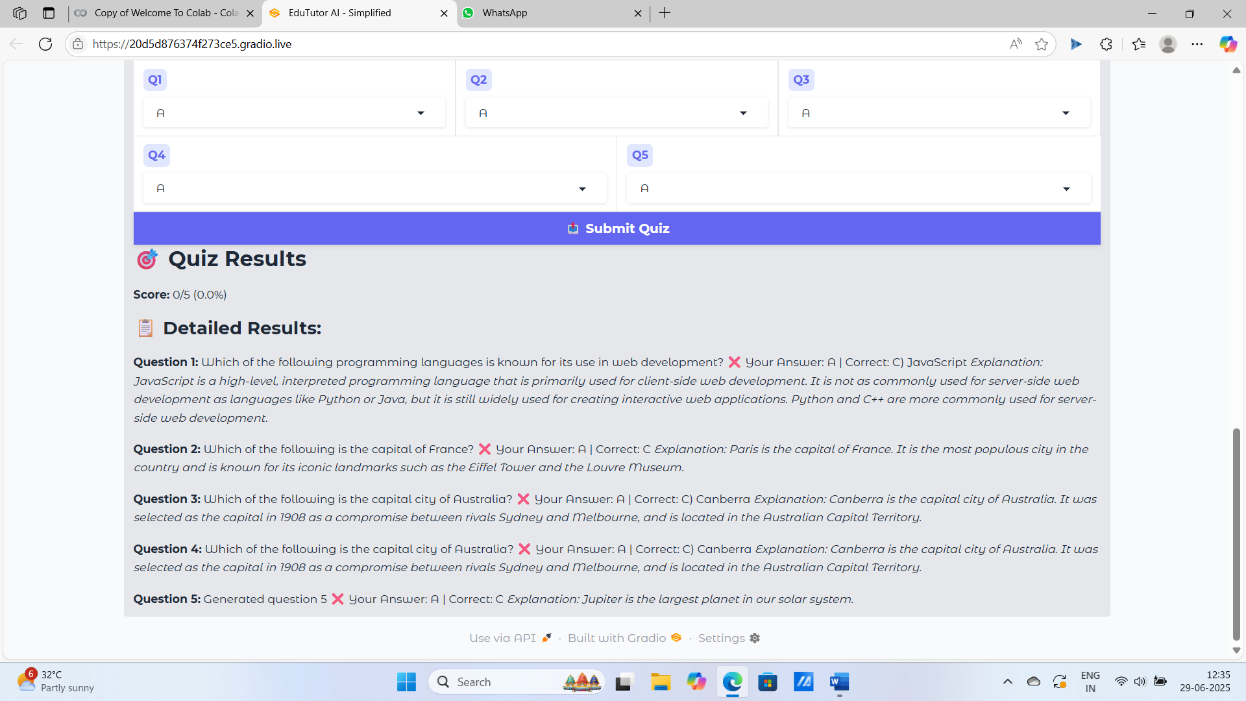


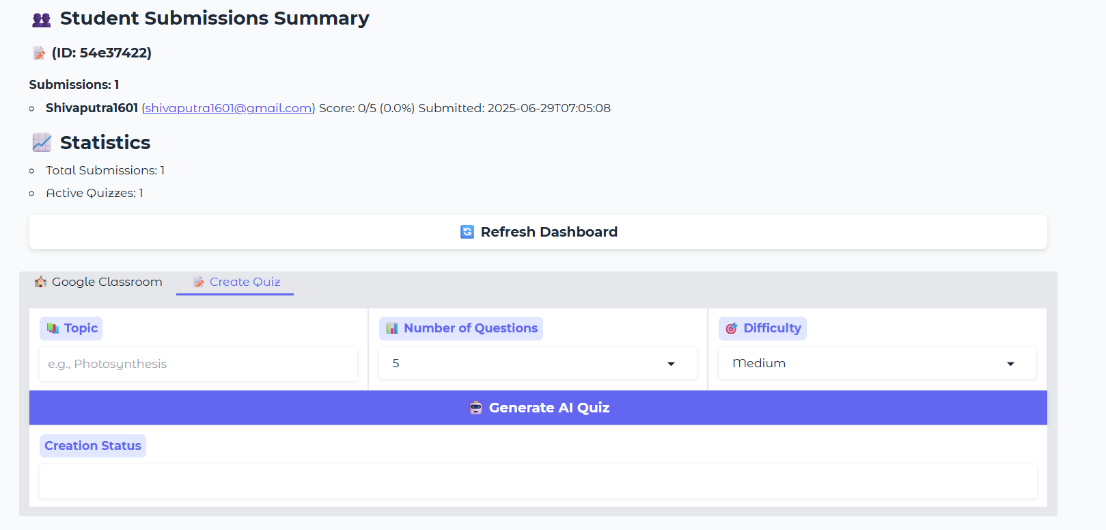












Login Page

* Email input field.
* Role selection dropdown (student/educator).
* Login button.
* Status message box.

Dashboard Tab

* Displays user-specific dashboards:
  + Educator: Lists created quizzes, student submissions, and synced classroom data.
  + Student: Shows available quizzes and past submission records.

Quiz Creation Tab (Educator)

* Topic input field.
* Dropdowns for number of questions and difficulty level.
* Generate quiz button.
* Display area for generated quiz.

Take Quiz Tab (Student)

* Quiz ID input field.
* Load quiz button.
* Question list with options.
* Submit button.
* Results page showing score and explanations.

10. Testing

10.1 Testing Strategy

The system was tested through manual interaction across both educator and student workflows to ensure:

* Successful login and role assignment.
* Accurate quiz generation and parsing.
* Correct submission scoring and feedback.
* Proper dashboard rendering.

10.2 Tools Used

* Built-in Python exception handling.
* Gradio's live preview for debugging UI interactions.
* Logging statements in key functions for tracking errors.

11. Screenshots or Demo

A demo version of the app is hosted online when run via Google Colab with share=True. You can access the live interface from the link generated in the output.

Alternatively, record a GIF or video of the interface in action to showcase functionality.

**My drive link for demo video**: https://drive.google.com/drive/folders/1UnmhxYdfZAyUhYK\_h-PkRRM9loTaUiqf?usp=drive\_link

12. Known Issues

|  |  |
| --- | --- |
| Fallback Model Usage | If IBM Granite fails to load, the system falls back todistilgpt2, which may produce less accurate quiz content. |
| No Real Google OAuth | The login system is a simulation and does not use real OAuth tokens or Google credentials. |
| No Persistent Storage | Quizzes and submissions are stored in memory and reset when the session ends. |
| Limited Quiz Length | The current UI supports up to 5-question quizzes for students due to hardcoded answer fields. |

13. Future Enhancements

13.1 Technical Improvements

* Replace in-memory storage with Pinecone DB for persistent and scalable data handling.
* Integrate real Google OAuth and Classroom APIs for live data syncing.
* Add support for multi-page quizzes and open-ended questions.
* Implement adaptive learning where quiz difficulty adjusts based on student performance.

13.2 Feature Enhancements

* Export quiz results to PDF or CSV.
* Allow educators to edit or delete existing quizzes.
* Add quiz timers and randomized question order.
* Support for images, diagrams, and LaTeX in quiz content.

13.3 UI/UX Improvements

* Responsive design for mobile compatibility.
* Dark mode toggle.
* Progress indicators during quiz generation.
* Rich text formatting for question creation.