Live Quiz App Documentation

# Introduction

The **Live Quiz App** is built using **FastAPI** for the backend and a simple frontend. To access the quiz, users must first log in through their Google account using **Firebase Authentication**. After logging in, they can either attend a quiz or edit the default set of questions stored in a **MongoDB** database.

Key Features:

Real-Time Score Updates: The quiz starts with a score of 0. The user earns +4 points for every correct answer and -1 point for every incorrect answer. The score updates in real time and is displayed at the top right corner of the website.

Quiz Attempt Summary: After completing the quiz, the final score and the number of questions attempted are shown.

Custom Quiz Creation: Users can either manually add questions to the quiz or use the OpenAI-powered feature to generate questions based on a topic.

## Scope – what I could do and couldn’t

What I Accomplished:

1. **API Development:** Implemented the quiz API with full CRUD operations using **FastAPI** and ran it through **uvicorn**.

2. **Database Integration:** Stored and edited quiz questions in **MongoDB** in a JSON format.

3. **Firebase Authentication:** Integrated Firebase for logging in via Google account.

4. **Real-Time Score Updates:** Implemented a real-time score display using **JavaScript**.

5. **Frontend Development:** Developed the frontend using **HTML** and **CSS**, ensuring it is dynamic and fully functional.

6.**Virtual Environment Setup:** Created and configured Python modules within a **VS Code virtual environment**.

What I couldn’t do:

1. AI-Powered Quiz Generation:

Issue: During testing, the AI-powered quiz generation feature consumed all free tokens and stopped functioning.

Fix: Switching to a paid API plan (e.g., GPT-4) should resolve the issue by providing more tokens for API calls.

1. Timer Functionality:

Issue: I attempted to use Python’s asyncio module to create a timer function, but encountered errors indicating the module was not found. Later, I tried to implement it with JavaScript, but abandoned the idea.

Fix: Further study and experimentation with the correct modules should help implement the timer functionality.

1. Real-Time Synchronization:

Issue: I struggled to find the right resources to implement real-time synchronization in FastAPI.

Fix: Exploring JavaScript for similar functionality may help me understand how to approach real-time synchronization in Python.

1. Project Deployment:

Issue: Due to lack of experience, I was unable to deploy the project, especially given the complexity of the backend. I considered using Docker for deployment.

Fix: With more time and practice, I plan to learn the proper deployment techniques for large-scale projects and explore Docker or other deployment methods.

## Requirements

To run this project, you need to set up a virtual environment and install all the required libraries. The necessary dependencies are listed in the requirements.txt file. To install the dependencies, use the following command:























