QuizMaster

A Quiz Application

Aryan Mahida – 92200133011

Jay Mangukiya - 92200133040

Brief Introduction

- In today's time, learning is becoming more interactive and collaborative through digital devices and advance technologies, and students often look for platforms that can make studying fun as well as effective. In order to work on that, I along with my peers have already made a Generative AI based web application for educational purposes which is called **BookTube**. But it was just a platform to learn and study. There is also a need of a tool which evaluates the knowledge as well. Which can be satisfied by any quiz platform.
- Although many quiz and learning apps already exist, most of them either focus on large audiences, have limited real time features, or require heavy setup and subscriptions. Just like Kahoot! And Slido. With this project, we simply aim to create a simple yet powerful quiz application which contains a unique functionality of 1v1 play mode, that works both on the web and mobile. The goal is to provide real time competition between two individuals, lightweight deployment, and easy access across devices. This idea directly connects with the ICT field where web development, real-time systems, and cloud solutions play a major role.

Problem Statement and Objectives

- Most of the existing quiz applications are either static or don't allow especially one to one competitive play mode. Many are designed for bigger groups like an entire class of 15-30 students, while some require complex hosting setups or even paid subscriptions. For students who just want to quickly connect with a friend through a lightweight cloud server, there are hardly any good options, that too without paying any amount also. This gap shows the need for a real time quiz platform that is simple, accessible, and encourages healthy competition in learning, which also gives reasonable accesses and authorities to a quiz admin to set up a quiz.

- To design and build a real time quiz application for web and mobile.
- To provide an easy to attend quiz application for students.
- To include 1v1 functionality for users to play against each other.
- To design a clean, user friendly interface which is easier to use and navigate.
- To include features like a leaderboard and basic analytics to track performance and compare with competitors.
- To provide admin access to quiz, such as time allotted to a question, points per question, difficulty etc.
- To provide question banks to admin for better question management and meaningful handling.
- To provide competitive quiz environment on the go.

Technical Highlights

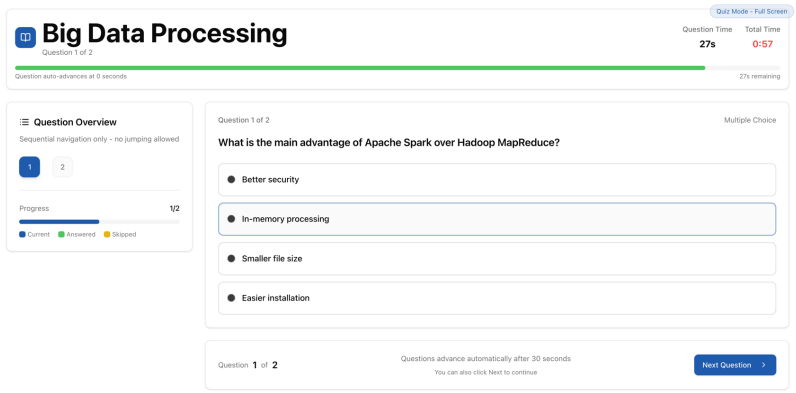
- The frontend uses React + Typescript, with reusable components, React hooks, context and Websocket support for live updates. Styling is handled with TailwindCSS and shadcn/ui, which leads to a clean and responsive interface for admins, teachers, and students.
- The backend runs on NodeJS + ExpressJS with an organized structure: controllers, services, middleware, and Sequelize ORM for postgreSQL. Real time sessions are handled by socket.io, for smooth score 1v1 communication between 2 users.
- For security purposes, we have used JWT authentication, bcrypt password hashing, and role-based access control. For scalability, the project is containerized with Docker, making deployment flexible for both small servers and cloud orchestration.
- Also, for checking the error rates, throughput etc. we have used Prometheus for monitoring purposes, which also makes the debugging easier.
- Other than that is the intelligent question bank, supporting bulk imports (CSV/Excel) and hierarchical categorization for better management.
- The question answer during the quizzes are encrypted so that there remains no chance for any cheating and plagiarism, making sure that the scores are valid.

Technical Solution for Quiz functionality

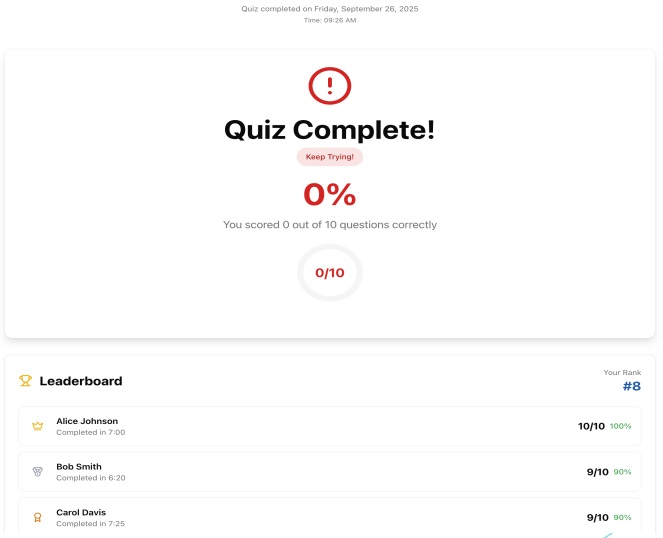
- When a quiz starts, the backend controller will fetch questions from PostgreSQL via sequelize ORM. These questions are shown to players through the socket.io room, which is synchronized with a timer. This makes sure that all players receive the same content and countdown simultaneously.
- Player responses are captured over socket.io events and validated in the middleware layer (auth + input validation). The server processes correctness checks using the question bank, and scores are updated in memory via Redis data store.
- To avoid cheating and plagiarism, the server handles all timing and scoring logic. Players only display question and can submit answers. Live leaderboards are calculated after each round and broadcast back through the room channel.
- At the end of a quiz, redis session data is stored to PostgreSQL for permanent storage. The Dockerized deployment is responsible for the quiz engine, database, and redis run in isolated but connected containers. By this, the Docker makes the application platform and hardware independent.

Outcomes

Quiz interface



Leaderboard



User testing

- The quiz functionality was tested with about 10 concurrent users who played quiz in a group of 2.

Conclusion

- This proposal is a practical, feasible, and student driven solution to the lack of simple real-time 1v1 quiz applications. The project has the clear objectives, direct relevance to ICT domain, and the ability to expand from a local setup to a scalable online platform.