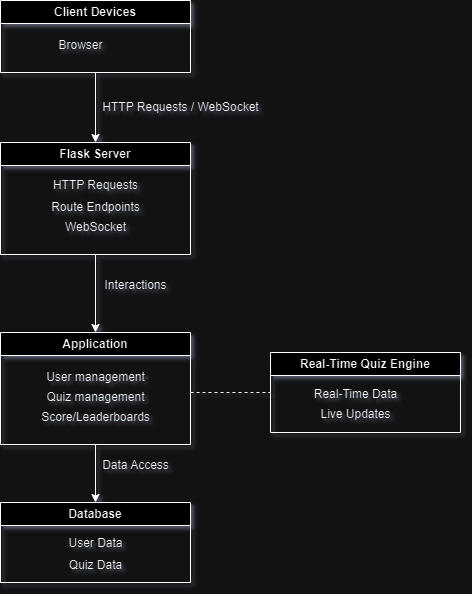
Architecture Diagram:



----------------------------------------------------------------------------

Component Description:

1. Client (Browser)

The user interacts with the quiz application through a web browser or mobile app. The client sends HTTP requests to the server, receives quiz questions, and submits answers. It also connects to the WebSocket server for real-time updates like live scores and leaderboard updates.

2. Flask Server (Backend API)

This is the main backend server handling the application's business logic. It processes client requests, manages user authentication, sessions, quiz logic, and interacts with the database. The Flask server is responsible for fetching questions, recording answers, and updating the user’s scores.

3. WebSocket Server (Real-Time Communication)

This server manages real-time communication between the client and the backend. It enables real-time features such as live score updates and leaderboard changes during the quiz session. WebSockets allow two-way communication for real-time data push.

4. MySQL Database (Data Storage)

Stores persistent data for the application. This includes user information, quiz questions, user answers, scores, and leaderboard rankings. It is the central data store that the Flask server interacts with.

----------------------------------------------------------------------------

Data Flow:

1. User Joining the Quiz:

- A user logs into the quiz application via the client (browser or mobile). The client sends an HTTP request to the Flask server for authentication.

- Flask server verifies the user’s credentials and creates a session using a unique session ID.

- Once authenticated, the Flask server retrieves the quiz data (questions, answers) from the MySQL database and sends it to the client.

2. Quiz Interaction:

- The client sends the user's answers back to the Flask server via HTTP POST requests.

- The Flask server processes the answers, updates the user’s score, and stores this in the MySQL database.

- The Flask server also sends the updated score to the WebSocket server.

3. Real-Time Score Updates:

- The WebSocket server broadcasts real-time score updates to all clients participating in the quiz. This ensures that each user can see their live rank and score compared to others in real time.

4. Leaderboard Update:

- After each quiz question, the Flask server calculates the new rankings

- The Flask server then sends the updated leaderboard data to the WebSocket server, which pushes the new leaderboard to all connected clients.

-----------------------------------------------------------------------------

Technologies and Tools

1. Flask (Python):

Flask is a lightweight and flexible framework for creating web applications. It is ideal for this application as it is simple to set up, easy to scale, and has good community support. Flask handles the main server-side logic and connects well with databases like MySQL.

2. WebSocket Server (using Flask-SocketIO):

Flask-SocketIO provides WebSocket functionality for Flask applications. It allows real-time communication between the client and the server, which is crucial for updating scores and leaderboards without refreshing the page.

3. MySQL:

A relational database is a perfect fit for structured data like quiz questions, answers, and scores. MySQL’s powerful querying capabilities and data integrity features make it reliable for this purpose.

4. HTML5, CSS, Jinja, and JavaScript (Frontend):

The client-side application is built using standard web technologies. HTML5 is used for structure, CSS for styling, and Jinja as a templating engine to dynamically render data from the Flask server on the client side. Jinja helps display server-side data such as quiz questions, scores, and leaderboard information directly in the HTML. JavaScript and jQuery are used for interactivity, handling form submissions, and enabling WebSocket-based real-time communication for live score and leaderboard updates