**README.md**

**Scoreboard API Service**

This document outlines the specifications for the Scoreboard API Service, which supports a website displaying a live-updated scoreboard of the top 10 users' scores. Users can increase their scores by performing certain actions, and these updates will be reflected in real-time.

**Features**

* Display the top 10 users' scores on the scoreboard.
* Live updates to the scoreboard when a user's score changes.
* Users can increase their scores by performing certain actions.
* Ensure secure and authorized score updates.

**API Endpoints**

**1. Get Top 10 Scores**

**Endpoint:** /api/scores/top

**Method:** GET

**Description:** Retrieves the top 10 users' scores.

**Response:**

[

{ "username": "user1", "score": 150 },

{ "username": "user2", "score": 145 },

...

]

**2. Update User Score**

**Endpoint:** /api/scores/update

**Method:** PUT

**Description:** Updates the score for a user after they perform an action.

**Request Body:**

{

"username": "user1",

"scoreIncrement": 5

}

**Response:**

{

"message": "Score updated successfully",

"newScore": 155

}

**Database Schema**

The database schema for storing user scores:

**Users Table**

| **Column** | **Type** | **Description** |
| --- | --- | --- |
| id | INTEGER | Primary key |
| username | TEXT | Unique username |
| score | INTEGER | User's score |

**Real-time Updates**

To achieve live updates on the scoreboard, we will use WebSockets. This allows the server to push updates to the clients whenever a score is updated.

**WebSocket Events**

**Event:** scoreUpdate

**Description:** Sent by the server to update clients with the latest scores.

**Payload:**

[

{ "username": "user1", "score": 155 },

{ "username": "user2", "score": 145 },

...

]

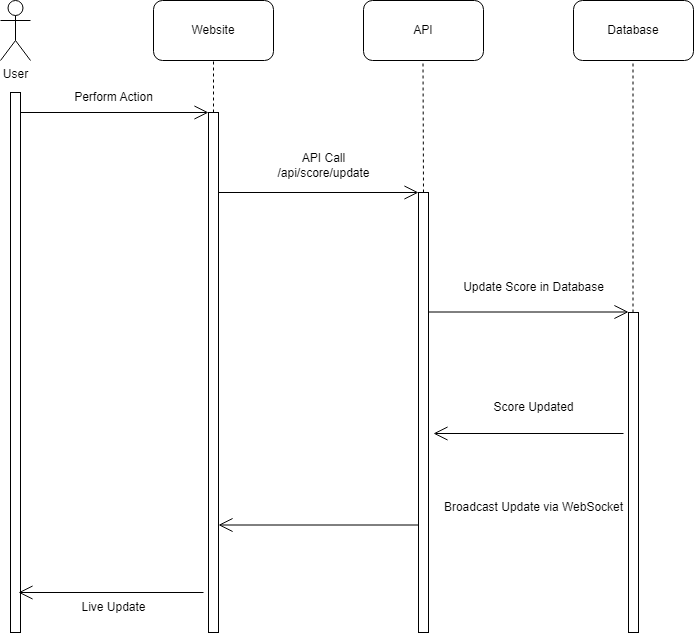
**Security**

To prevent malicious users from unauthorized score increases, we will implement the following security measures:

1. **Authentication:** Ensure users are authenticated before allowing score updates.
2. **Validation:** Validate the scoreIncrement value to prevent abnormal increases.
3. **Rate Limiting:** Implement rate limiting to prevent abuse by rapidly increasing scores.

**Diagram**

The following diagram illustrates the flow of execution for updating scores and the real-time update mechanism:



**Figure 1: Execution Flow Diagram**

**Future Improvements**

1. **Enhanced Security:** Implement additional security measures such as JWT tokens for authentication and authorization.
2. **Caching:** Use caching mechanisms like Redis to reduce database load for frequently accessed data.
3. **Scalability:** Design the system to handle higher loads by distributing the WebSocket connections and load balancing the API server.
4. **Analytics:** Add endpoints for detailed analytics on user scores and actions.