WebSocket-Based Real-Time Animation System

1. How would you design and implement this full-stack WebSocket-based animation system?

The system consists of a React (Vite) frontend for handling animations and user interactions, and a Node.js WebSocket backend for managing real-time communication.

- Frontend:
  * Establishes a WebSocket connection to the server.
  * Provides 'Start' and 'Stop' buttons to control animation flow.
  * Renders real-time animation data received from the server.

- Backend:
  * Implements a WebSocket server using 'ws'.
  * Listens for client messages ('start' or 'stop') and broadcasts animation data accordingly.
  * Manages multiple client connections efficiently.

2. Provide a basic code snippet for handling WebSocket connections in both the client-side (React) and server-side (Node.js).

Client-Side (React - WebSocket Connection & Animation Handling):

```
import React, { useEffect, useState, useRef } from 'react';
const WebSocketAnimation = () => {
    const [socket, setSocket] = useState(null);
    const [animationData, setAnimationData] = useState(null);
    const canvasRef = useRef(null);
    useEffect(() => {
        const ws = new WebSocket('ws://localhost:5001');
        ws.onopen = () => console.log('Connected');
        ws.onmessage = (event) => setAnimationData(JSON.parse(event.data));
        ws.onclose = () => console.log('Disconnected');
        setSocket(ws);
        return () => ws.close();
    }, []);
    return (
        <div>
            <canvas ref={canvasRef} width={400} height={400} style={{ border: '1px solid black' }} />
            <button onClick={() => socket?.send(JSON.stringify({ action: 'start' }))}>Start</button>
            <button onClick={() => socket?.send(JSON.stringify({ action: 'stop' }))}>Stop</button>
        </div>
    );
};
export default WebSocketAnimation;
```

```

Server-Side (Node.js - WebSocket Handling):

```
import { WebSocketServer } from 'ws';
const wss = new WebSocketServer({ port: 5001 });
wss.on('connection', (ws) => {
    console.log('Client connected');
    let animationInterval;
    ws.on('message', (message) => {
        const { action } = JSON.parse(message);
        if (action === 'start') {
            clearInterval(animationInterval);
            animationInterval = setInterval(() => {
                const animationData = { x: Math.random() * 400, y: Math.random() * 400 };
                ws.send(JSON.stringify(animationData));
            }, 100);
        } else if (action === 'stop') {
            clearInterval(animationInterval);
        }
    });
    ws.on('close', () => {
        console.log('Client disconnected');
        clearInterval(animationInterval);
    });
});
```

### 3. How would you ensure smooth performance and handle multiple clients efficiently?

- Optimize client-side rendering using requestAnimationFrame.
- Use non-blocking asynchronous operations in the backend.
- Broadcast animation data efficiently using WebSocket optimizations.
- Utilize connection pooling and keep active connections lightweight.

### 4. What strategies would you use to handle WebSocket disconnections and reconnections?

- Implement automatic reconnection with an exponential backoff strategy.
- Detect connection loss and attempt a graceful reconnect.
- Use heartbeat pings to keep idle connections active.
- Store session state on the backend to restore animations upon reconnection.

### 5. How would you scale this application to support thousands of concurrent users?

- Horizontal scaling with multiple WebSocket servers using a load balancer.
- Use Redis Pub/Sub to distribute WebSocket messages across multiple servers.
- Deploy using Kubernetes for containerized scalability.
```

- Implement WebSocket sharding to distribute connections evenly.

6. What security measures would you implement to prevent WebSocket abuse?

- Rate limiting to prevent excessive requests per user.
- JWT-based authentication to ensure only authorized users can connect.
- IP filtering and WebSocket firewall rules to mitigate bot-based attacks.
- Monitor WebSocket traffic for suspicious patterns and take preventive actions.

Abhishek Kumar
+918986183108