Real-Time Chat Application Using Socket.io

Saksham Rathore(23BCS10814)

Objective:

Learn how to build a real-time chat application using WebSocket connections with Socket.io in a Node.js backend and React frontend. Understand two-way communication between server and clients, manage connected users, and update the UI instantly as messages are exchanged.

Description:

Create a Node.js server using Express and integrate Socket.io to handle WebSocket connections. Build a React frontend that connects to the backend Socket.io server. Implement a simple chat UI where users can enter their name and send messages to a shared room. Display the list of messages in real time to all connected users without refreshing the page. Handle basic events like connecting, disconnecting, and broadcasting messages to all clients.

Architecture and Files:

Files included:
1) Backend (server)
  - package.json
  - index.js
2) Frontend (React app)
  - package.json
  - src/App.js
  - src/index.js
  - src/index.css (optional)
3) README and this report

Server Code (index.js):

```
const express = require('express');
const http = require('http');
const { Server } = require('socket.io');
const cors = require('cors');

const app = express();
app.use(cors());
const server = http.createServer(app);
const io = new Server(server, { cors: { origin: '*', methods: ['GET', 'POST'] } });

io.on('connection', (socket) => {
  console.log('a user connected:', socket.id);
  socket.on('chat message', (msg) => io.emit('chat message', msg));
  socket.on('disconnect', () => console.log('user disconnected:', socket.id));
});

server.listen(4000, () => console.log('Server running on port 4000'));
```

Frontend Code (App.js):

```
import React, { useEffect, useState, useRef } from 'react';
import io from 'socket.io-client';
const socket = io('http://localhost:4000');

function App() {
  const [name, setName] = useState('');
  const [message, setMessage] = useState('');
```

```
  const [messages, setMessages] = useState([]);

  useEffect(() => {
    socket.on('chat message', (msg) => setMessages(prev => [...prev, msg]));
    return () => socket.disconnect();
  }, []);

  const sendMessage = () => {
    if (name && message) {
      const msg = { name, text: message, time: new Date().toLocaleTimeString() };
      socket.emit('chat message', msg);
      setMessage('');
    }
  };

  return (
    <div>
      <h2>Real-Time Chat</h2>
      <input placeholder="Your name" value={name} onChange={e => setName(e.target.value)} />
      <div style={{border:'1px solid #ccc', minHeight:200, padding:10, margin:10}}>
        {messages.map((m,i)=><div key={i}><b>{m.name}</b> [{m.time}]: {m.text}</div>)}
      </div>
      <input placeholder="Type message..." value={message}
        onChange={e=>setMessage(e.target.value)} onKeyDown={e=>e.key==='Enter'&&sendMessage()} />
      <button onClick={sendMessage}>Send</button>
    </div>
  );
}
export default App;
```

Setup & Testing:

1. In the server folder, run: npm install && node index.js
2. In the client folder, run: npm install && npm start
3. Open http://localhost:3000 in two browsers, enter names, and send messages.
4. Messages appear instantly across all clients.

Example Output: