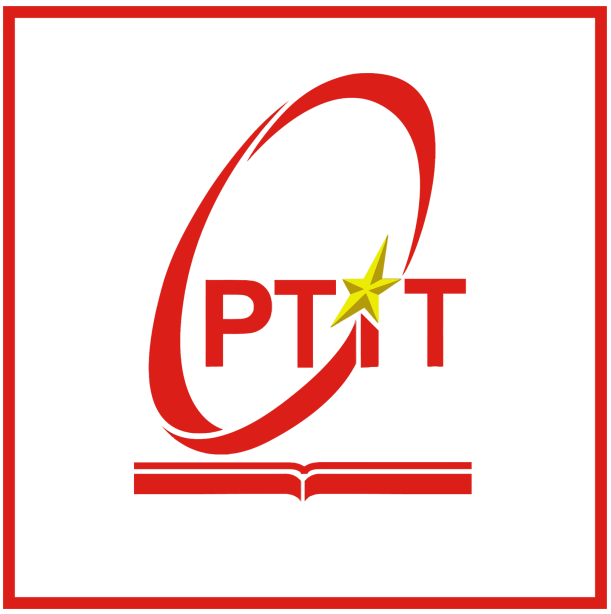
BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO THỰC TẬP CƠ SỞ TUẦN 11

TRIỂN KHAI
UX/UI VÀ FRONT END, BACK END
(TIẾP)

Giảng viên hướng dẫn: TS. Kim Ngọc Bách

Sinh viên thực hiện:

- Nguyễn Quang Trung - B22DCDT321

MỤC LỤC

Tổng quan về Socket.IO

Socket.IO là một thư viện mạnh mẽ cho phép xây dựng các ứng dụng web thời gian thực dựa trên giao thức WebSocket, đồng thời hỗ trợ fallback sang các giao thức khác khi cần thiết. Socket.IO cung cấp API đơn giản cho cả phía server (Node.js) và client (JavaScript), giúp việc triển khai các tính năng như chat, thông báo, cập nhật trạng thái... trở nên dễ dàng và hiệu quả.

Quá trình triển khai

- Cài đặt và cấu hình

```
import { Server } from "socket.io";
import { Message } from "../models/message.model.js";

export const initializeSocket = (server) => {
    const io = new Server(server, {
        cors: {
            origin: "http://localhost:5173",
            credentials: true,
        },
    });

    const userSockets = new Map(); // { userId: socketId}
    const userActivities = new Map(); // {userId: activity}

    io.on("connection", (socket) => {
        console.log("New client connected:", socket.id);

        socket.on("user_connected", (userId) => {
            console.log("User connected:", userId);
            userSockets.set(userId, socket.id);
            userActivities.set(userId, "Idle");

            // broadcast to all connected sockets that this user just logged
in
            io.emit("user_connected", userId);

            socket.emit("users_online", Array.from(userSockets.keys()));
```

```
        io.emit("activities", Array.from(userActivities.entries()));
    });

    socket.on("update_activity", ({ userId, activity }) => {
        console.log("activity updated", userId, activity);
        userActivities.set(userId, activity);
        io.emit("activity_updated", { userId, activity });
    });

    socket.on("send_message", async (data) => {
        try {
            const { senderId, receiverId, content } = data;

            const message = await Message.create({
                senderId,
                receiverId,
                content,
            });

            // send to receiver in realtime, if they're online
            const receiverSocketId = userSockets.get(receiverId);
            if (receiverSocketId) {
                io.to(receiverSocketId).emit("receive_message", message);
            }

            socket.emit("message_sent", message);
        } catch (error) {
            console.error("Message error:", error);
            socket.emit("message_error", error.message);
        }
    });

    socket.on("disconnect", () => {
        console.log("Client disconnected:", socket.id);
        let disconnectedUserId;
        for (const [userId, socketId] of userSockets.entries()) {
            // find disconnected user
            if (socketId === socket.id) {
                disconnectedUserId = userId;
                userSockets.delete(userId);
                userActivities.delete(userId);
                break;
```

```
                }
            }
            if (disconnectedUserId) {
                io.emit("user_disconnected", disconnectedUserId);
            }
        });
    });
};
```

- Kết nối:

Khi người dùng truy cập vào trang chat, client sẽ khởi tạo kết nối tới server Socket.IO. Server xác thực người dùng (thông qua token hoặc session) và lưu lại thông tin kết nối.

- Gửi và nhận tin nhắn:

Khi người dùng gửi tin nhắn, client phát sự kiện (emit) tới server với nội dung tin nhắn và thông tin người nhận. Server nhận sự kiện, kiểm tra hợp lệ, sau đó chuyển tiếp tin nhắn tới client đích (người nhận) thông qua socket tương ứng.

- Nhận tin nhắn mới:

Client lắng nghe sự kiện nhận tin nhắn mới từ server. Khi có tin nhắn đến, giao diện chat sẽ tự động cập nhật, hiển thị nội dung mới mà không cần reload trang.

- Quản lý kết nối:

Server quản lý danh sách các socket đang kết nối, đảm bảo mỗi người dùng chỉ nhận được tin nhắn của mình. Khi người dùng rời khỏi trang hoặc mất kết nối, server sẽ cập nhật lại trạng thái.

- Xử lý các trường hợp đặc biệt

- Ngắt kết nối:

Khi client mất kết nối (do tắt trình duyệt, mất mạng...), server sẽ lắng nghe sự kiện disconnect để cập nhật lại trạng thái người dùng.

- Thông báo lỗi:

Nếu có lỗi trong quá trình gửi/nhận tin nhắn (ví dụ: người nhận không tồn tại, nội dung không hợp lệ...), server sẽ gửi thông báo lỗi về client để hiển thị cho người dùng.

→ Kết quả đạt được

- Chức năng chat real-time:

Đã triển khai thành công chức năng chat nhắn tin giữa các người dùng với tốc độ truyền tải tức thời, không cần reload trang.

- Đồng bộ dữ liệu:

Tin nhắn được đồng bộ giữa các client, đảm bảo tính nhất quán và liên tục.

- Trải nghiệm người dùng:

Giao diện chat mượt mà, phản hồi nhanh, nâng cao trải nghiệm sử dụng hệ thống.

- Mã nguồn rõ ràng:

Code backend và frontend được tổ chức rõ ràng, dễ bảo trì, dễ mở rộng cho các tính năng real-time khác trong tương lai (ví dụ: thông báo, trạng thái online/offline...).

- Khó khăn và hướng giải quyết

- Quản lý nhiều kết nối:

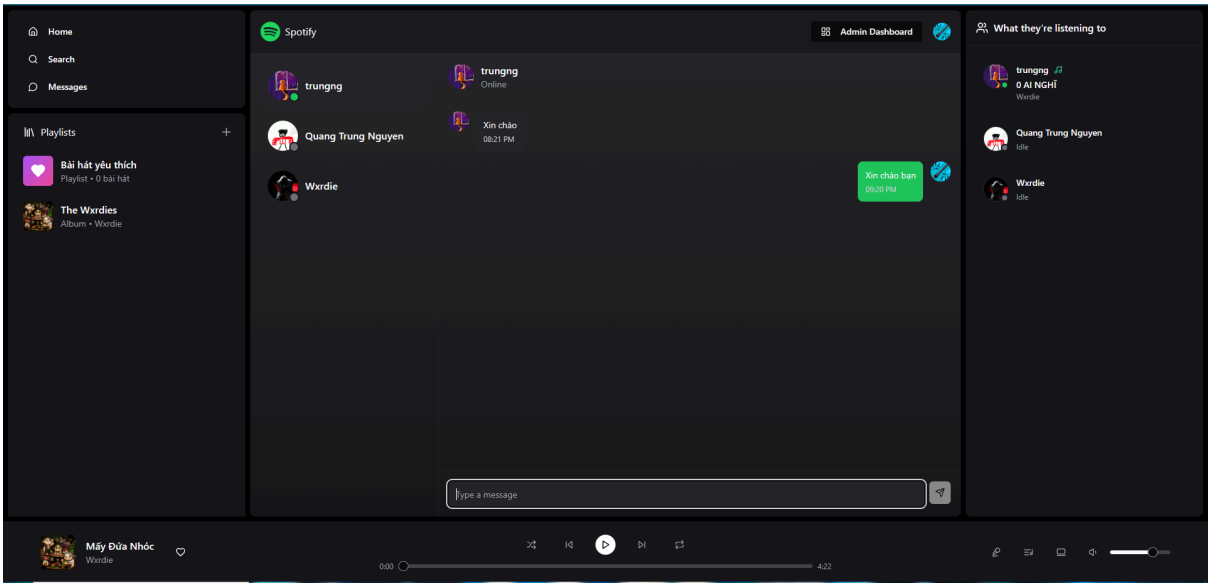
Khi nhiều người dùng cùng online, việc quản lý socket và phân phối tin nhắn cần tối ưu để tránh trùng lặp hoặc thất lạc tin nhắn. Nhóm đã sử dụng các cấu trúc dữ liệu phù hợp để lưu mapping giữa userId và socketId.

- Bảo mật:

Đảm bảo chỉ người nhận hợp lệ mới nhận được tin nhắn, tránh rò rỉ thông tin. Đã tích hợp xác thực token khi kết nối socket.

- Xử lý mất kết nối:

Đã bổ sung logic tự động reconnect cho client khi mất kết nối tạm thời.


Home
Search
Messages
Playlists
Bài hát yêu thích
Playlist • 0 bài hát
The Wxrdies
Album • Wxrdie
Spotify
Admin Dashboard
trungng
Online
Xin chào
08:21 PM
Quang Trung Nguyen
Wxrdie
Xin chào bạn
09:20 PM
Type a message
What they're listening to
trungng
0 AI NGHĨ
Wxrdie
Quang Trung Nguyen
Idle
Wxrdie
Idle
Mấy Đứa Nhóc
Wxrdie
0:00
4:22