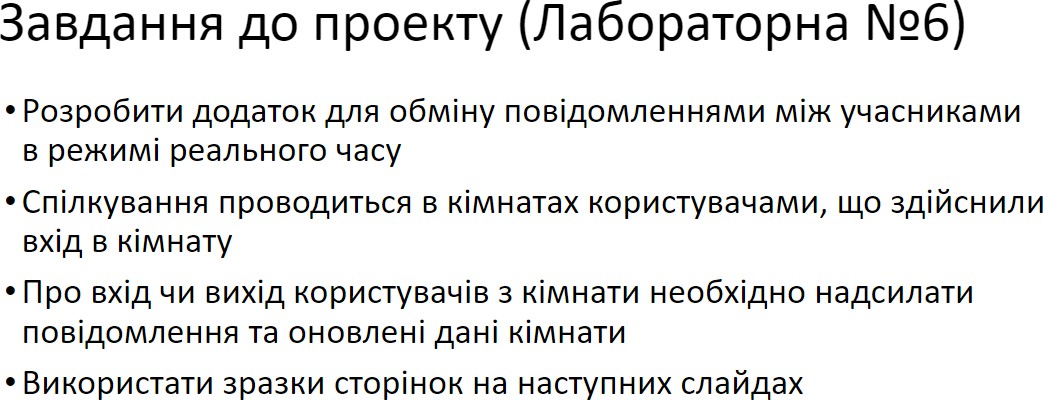
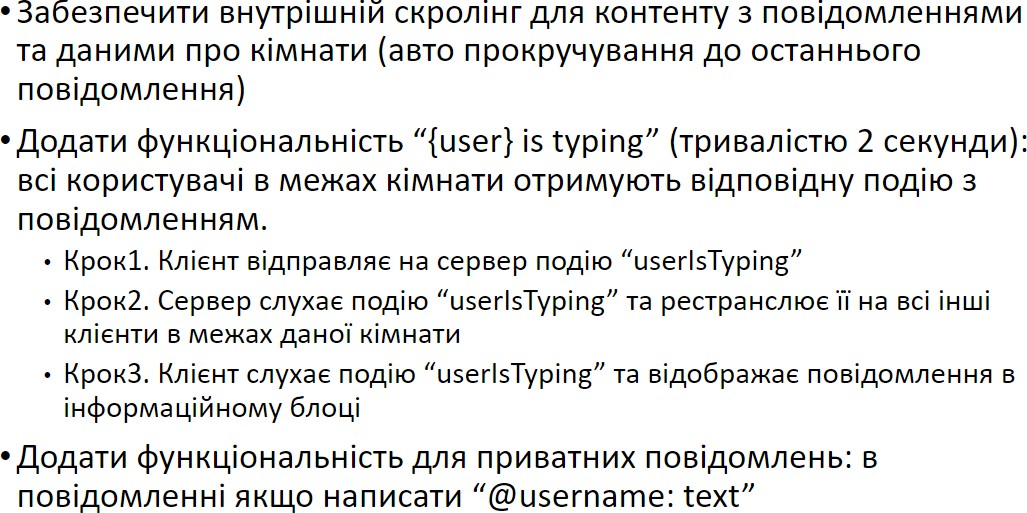
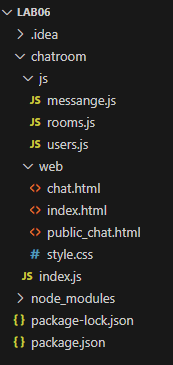
**ЛАБОРАТОРНА РОБОТА № 6**

Протокол WebSocket. Використання Socket.io для розробки чат- додатків.

Project03 “Chat Application”







**Лістинг проекту:**

**Message.js:**

const generateMessage = (username, text) => {

    return {

        username,

        text,

        createdAt: new Date().getTime()

    }

}

const generateLocationMessage = (username, url) => {

    return {

        username,

        url,

        createdAt: new Date().getTime()

    }

}

module.exports = {

    generateMessage,

    generateLocationMessage

}

**rooms.js:**

const private\_rooms = []

const getPrivateRooms = () => {

    return private\_rooms;

}

const isRoomPrivate = (room) => {

    room = room.trim().toLowerCase()

    if (inArray(private\_rooms, room))

        return true;

    return false;

}

const setRoomPrivate = (state, room) => {

    room = room.trim().toLowerCase()

    if(state === true)

        removeFromArr(private\_rooms, room)

    else

    if(!inArray(private\_rooms, room))

        private\_rooms.push(room);

}

function inArray(array, item)

{

    let index = array.indexOf(item);

    if (index !== -1)

        return true

    return false;

}

function removeFromArr(array, item)

{

    let index = array.indexOf(item);

    if (index !== -1) {

        array.splice(index, 1);

    }

}

module.exports = {

    getPrivateRooms,

    isRoomPrivate,

    setRoomPrivate

}

**users.js:**

const users = []

const addUser = ({ id, username, room }) => {

    // Clean the data

    username = username.trim().toLowerCase()

    room = room.trim().toLowerCase()

    // Validate the data

    if (!username || !room) {

        return {

            error: 'Username and room are required!'

        }

    }

    // Check for existing user

    const existingUser = users.find((user) => {

        return user.room === room && user.username === username

    })

    // Validate username

    if (existingUser) {

        return {

            error: 'Username is in use!'

        }

    }

    // Store user

    const user = { id, username, room }

    users.push(user)

    return { user }

}

const removeUser = (id) => {

    const index = users.findIndex((user) => user.id === id)

    if (index !== -1) {

        return users.splice(index, 1)[0]

    }

}

const getUser = (id) => {

    return users.find((user) => user.id === id)

}

const getUsersInRoom = (room) => {

    room = room.trim().toLowerCase()

    return users.filter((user) => user.room === room)

}

const getUserByUsername = (username) => {

    return users.find((user) => user.username === username.toLowerCase());

}

module.exports = {

    addUser,

    removeUser,

    getUser,

    getUsersInRoom,

    getUserByUsername // Add this line

}

**index.js:**

const express = require('express')

const http = require('http')

const socketio = require('socket.io')

const hbs = require('hbs');

const {generateMessage} = require("./js/messange")

const {addUser, removeUser, getUser, getUsersInRoom, getUserByUsername} = require('./js/users')

const {isRoomPrivate, setRoomPrivate} = require('./js/rooms')

const app = express()

const server = http.createServer(app)

const io = socketio(server)

app.set('view engine', 'hbs');

app.use(express.static(\_\_dirname + '/web'));

io.on('connection', (socket) => {

    socket.on('join', (options, callback) => {

        const {error, user} = addUser({id: socket.id, ...options})

        if(isRoomPrivate(user.room)) {

            removeUser(socket.id)

            return;

        }

        if (error) {

            return callback(error)

        }

        socket.join(user.room)

        socket.emit('message', generateMessage('Admin', `Welcome, ${user.username}!`))

        socket.broadcast.to(user.room).emit('message', generateMessage('Admin', `${user.username} has joined!`))

        io.to(user.room).emit('roomData', {

            users: getUsersInRoom(user.room)

        })

        callback()

    })

    socket.on('sendMessage', ({ message, to }, callback) => {

        const user = getUser(socket.id);

        if (to) {

            const recipient = getUserByUsername(to);

            if (recipient && recipient.room === user.room) {

                io.to(recipient.id).emit('message', generateMessage(user.username, message, true));

                callback();

                return;

            } else {

                callback('Recipient not found or not in the same room.');

                return;

            }

        }

        io.to(user.room).emit('message', generateMessage(user.username, message));

        callback();

    });

    socket.on('setPrivate', (message, callback) => {

        const user = getUser(socket.id)

        setRoomPrivate(!message.is\_checked, user.room)

        io.to(user.room).emit('message', generateMessage('Admin', `Changed room settings`))

        callback()

    })

    socket.on('userTyping', (message, callback) => {

        const user = getUser(socket.id)

        io.to(user.room).emit('userTyping', generateMessage(`${user.username}`, `${user.username} is typing...`))

        callback()

    })

    socket.on('disconnect', () => {

        const user = removeUser(socket.id)

        if (user) {

            io.to(user.room).emit('message', generateMessage('Admin', `${user.username} has left!`))

            io.to(user.room).emit('roomData', {

                users: getUsersInRoom(user.room),

                isPrivate: isRoomPrivate(user.room)

            })

        }

    })

});

app.get('/', (req, res) => {

    res.sendFile(\_\_dirname + '/web/index.html');

});

app.get('/chat', (req, res) => {

    res.sendFile(\_\_dirname + '/web/chat.html');

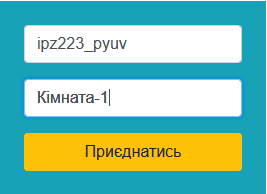
});

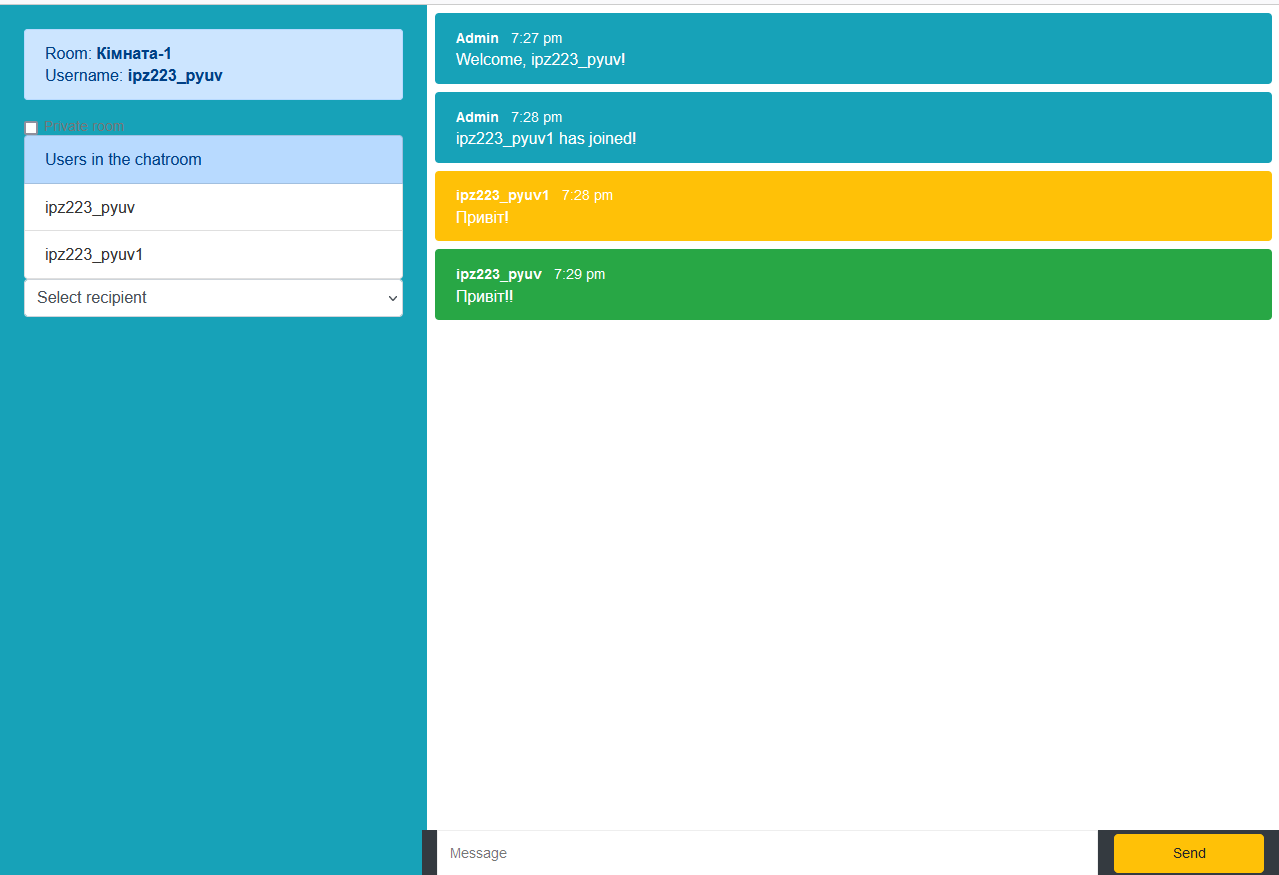
server.listen(3000, function () {

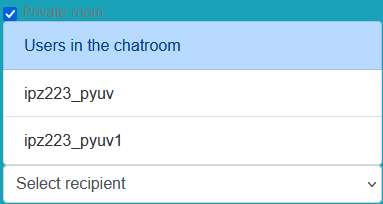
    console.log('listening on http://localhost:3000');

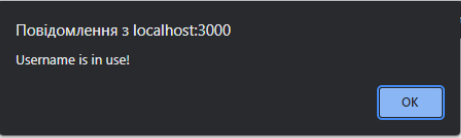
});

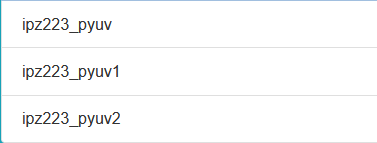
**Результат виконання завдання:**

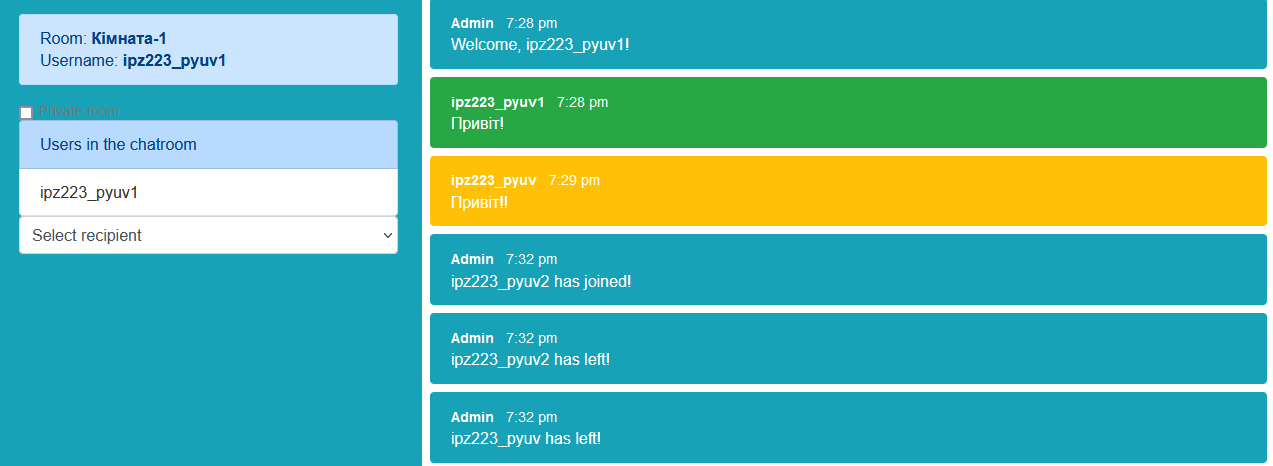












**Висновок:** в ході виконання лабораторної роботи познайомився з WebSocket та Socket.io