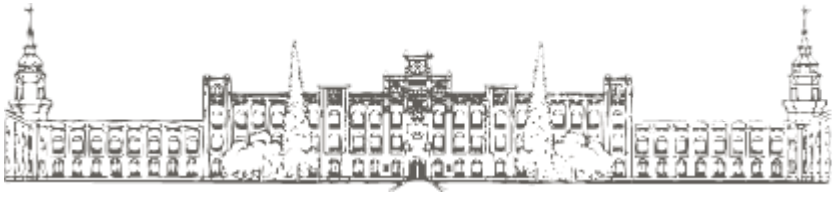


Міністерство освіти і науки України
Національний технічний університет України „КПІ
імені Ігоря Сікорського ”
Факультет інформатики та обчислювальної техніки
Кафедра інформатики і програмної інженерії

ЗВІТ

лабораторної роботи №1

з курсу «Основи WEB - технологій»

Тема: «Протокол WebSocket. Використання Socket.io для розробки чат-додатків»

Перевірив:

Викл. Альбрехт Й.О.

Виконала:

ст. Мельникова Катерина гр. ІС-12

Київ 2024

Завдання1. Розробити додаток для обміну повідомленнями між учасниками в режимі реального часу (chat) за допомогою бібліотеки SocketIO.

Хід роботи

Повний код проєкту можна переглянути за посиланням:
https://github.com/katerynamelnykova/web-technologies-labs

Файл server.js:

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

const app = express();
const server = http.createServer(app);
const io = new Server(server);

let usersInRooms = {};

app.get('/', (req, res) => {
   res.sendFile(__dirname + '/index.html');
});



io.on('connection', (socket) => {
   socket.on('join room', ({ username, room }) => {
       socket.join(room);
       socket.username = username;
       socket.room = room;

       if (!usersInRooms[room]) {
           usersInRooms[room] = [];
       }
       usersInRooms[room].push(username);

       io.to(room).emit('user joined', {
           username: username,
           usersInRoom: usersInRooms[room],
       });
       io.to(room).emit('system message', {
           message: `User ${username} joined the room`
       })
```

```
    });

    socket.on('chat message', (msg) => {
        const time = new Date().toLocaleTimeString();
        io.to(socket.room).emit('chat message', {
            username: socket.username,
            time: time,
            message: msg
        });
    });

    socket.on('disconnect', () => {
        if (socket.room) {
            usersInRooms[socket.room] = usersInRooms[socket.room].filter(user => user !==
socket.username);
            io.to(socket.room).emit('user left', {
                username: socket.username,
                usersInRoom: usersInRooms[socket.room],
            });

            io.to(socket.room).emit('system message', {
                message: `User ${socket.username} left the room`
            })
        }
    });
});

server.listen(3000, () => {
    console.log('Сервер працює на порту 3000');
});
```

index.html

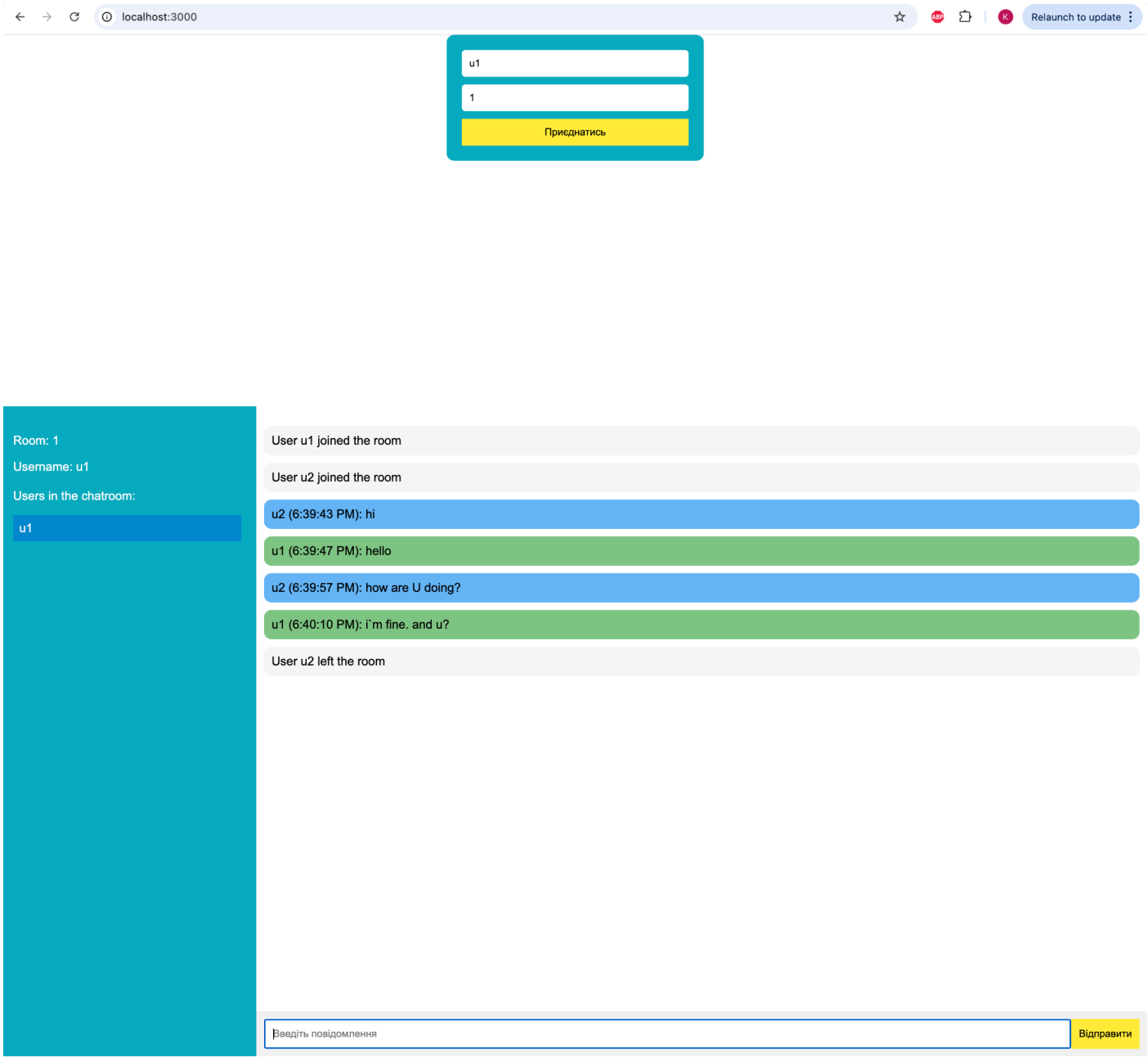
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Chat Room</title>
    <style>
```

```
        body { font-family: Arial, sans-serif; margin: 0; padding: 0; }
        .chat-container { display: flex; height: 100vh; }
        .sidebar { width: 300px; background: #00acc1; padding: 20px; color: white; }
        .chat-room-info { margin-bottom: 20px; }
        .user-list { list-style-type: none; padding: 0; }
        .user-list li { padding: 8px; background: #0288d1; margin-bottom: 5px; }
        .chat-window { flex: 1; display: flex; flex-direction: column; }
        .messages { flex: 1; list-style-type: none; padding: 10px; overflow-y: scroll; }
        .messages li { margin-bottom: 10px; padding: 10px; border-radius: 10px; }
        .user-message { background: #81c784; }
        .other-message { background: #64b5f6; }
        .system-message { background: #f2f2f2c3; }
        #form { display: flex; padding: 10px; background: #f2f2f2; }
        #input { flex: 1; padding: 10px; }
        button { padding: 10px; background: #ffeb3b; border: none; cursor: pointer; }
        #join-form { display: flex; flex-direction: column; width: 300px; margin: auto; padding:
20px; background: #00acc1; color: white; border-radius: 10px; }
        #join-form input { margin-bottom: 10px; padding: 10px; border: none; border-radius: 5px; }
        #join-form button { padding: 10px; background: #ffeb3b; border: none; cursor: pointer; }
    </style>
</head>
<body>

<div id="join-container">
    <form id="join-form">
        <input id="username" type="text" placeholder="Ваше ім'я" required>
        <input id="room" type="text" placeholder="Назва кімнати" required>
        <button type="submit">Приєднатись</button>
    </form>
</div>

<div id="chat-container" class="chat-container" style="display:none;">
    <div class="sidebar">
        <div class="chat-room-info">
            <p>Room: <span id="room-name"></span></p>
            <p>Username: <span id="user-name"></span></p>
        </div>
        <p>Users in the chatroom:</p>
        <ul id="users" class="user-list"></ul>
    </div>
    <div class="chat-window">
        <ul id="messages" class="messages"></ul>
        <form id="form">
            <input id="input" autocomplete="off" placeholder="Введіть повідомлення" />
            <button>Відправити</button>
```

```
        </form>
    </div>
</div>

<script src="/socket.io/socket.io.js"></script>
<script>
    var socket = io();
    var username, room;

    document.getElementById('join-form').addEventListener('submit', function(e) {
        e.preventDefault();
        username = document.getElementById('username').value;
        room = document.getElementById('room').value;

        if (username && room) {
            // Приховуємо форму і показуємо чат
            document.getElementById('join-container').style.display = 'none';
            document.getElementById('chat-container').style.display = 'flex';

            // Відправляємо дані на сервер
            socket.emit('join room', { username: username, room: room });

            // Оновлюємо інформацію в інтерфейсі чату
            document.getElementById('room-name').textContent = room;
            document.getElementById('user-name').textContent = username;
        }
    });

    var form = document.getElementById('form');
    var input = document.getElementById('input');

    form.addEventListener('submit', function(e) {
        e.preventDefault();
        if (input.value) {
            socket.emit('chat message', input.value);
            input.value = '';
        }
    });

    socket.on('chat message', function(data) {
        var item = document.createElement('li');
        var messageContent = `${data.username} (${data.time}): ${data.message}`;
        item.textContent = messageContent;
        item.classList.add(data.username === username ? 'user-message' : 'other-message');
        document.getElementById('messages').appendChild(item);
```

```
    });

    socket.on('system message', function(data) {
        var item = document.createElement('li');
        item.textContent = `${data.message}`;
        item.classList.add("system-message")
        document.getElementById('messages').appendChild(item);
    })

    socket.on('user joined', function(data) {
        updateUsersList(data.usersInRoom);
    });

    socket.on('user left', function(data) {
        updateUsersList(data.usersInRoom);
    });

    function updateUsersList(users) {
        var usersList = document.getElementById('users');
        usersList.innerHTML = '';
        users.forEach(function(user) {
            var userItem = document.createElement('li');
            userItem.textContent = user;
            usersList.appendChild(userItem);
        });
    }
</script>

</body>
</html>
```

Отримані результати:



Висновок

Під час виконання даної лабораторної роботи я навчилася використовувати бібліотеку SocketIO та розробила додаток для обміну повідомленнями між учасниками в режимі реального часу.