

Міністерство освіти і науки України

Національний технічний університет України „КПІ

імені Ігоря Сікорського ”

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

**ЗВІТ**

лабораторної работи №4

курсу «Основи WEB - технологій »

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

Перевірив: Виконав:

Доц. Голубєв Л. П. Гр. ІП-02

Трофимов Данило

Київ 2022

**Завдання 1.**

Розробити додаток для обміну повідомленнями між учасниками в режимі реального часу за допомогою бібліотеки SocketIO

**Хід роботи:**

За допомогою текстового редактора VSCode, було створено веб-додаток. Додаток має наступну структуру: index.js, index.html, style.css, app.css. Файл index.js являє собою веб-сервер на базі бібліотеки express з використанням Socket.io.

Лістинг файлу index.js

const express = require('express');

const http = require('http');

const socketIO = require('socket.io');

const app = express();

const server = http.createServer(app);

const io = socketIO(server);

app.use(express.static('./'));

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

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

});

const users = {};

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

    console.log('A user connected');

    socket.on('register', (user) => {

        users[socket.id] = user

        io.emit('userList', Object.values(users));

        socket.emit('message', {

            text: 'You have connected to the chat.',

            username: 'System',

            color: 'gray',

        });

        socket.broadcast.emit('message', {

            text: `${user.username} has joined the chat.`,

            username: 'System',

            color: 'gray',

        });

    });

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

        const user = users[socket.id];

        if (user) {

            io.emit('message', {

                text: message,

                username: user.username,

                color: user.color,

            });

        }

    });

    socket.on('disconnect', () => {

        const user = users[socket.id];

        if (user) {

            delete users[socket.id];

            io.emit('user-list', Object.values(users));

            socket.broadcast.emit('message', {

                text: `${user.username} has left the chat.`,

                username: 'System',

                color: 'gray',

            });

        }

    });

});

const port = process.env.PORT || 3000;

server.listen(port, () => {

    console.log(`Server is running on http://localhost:${port}`);

});

Файл index.html відповідає за взаємодію клієнтської частини із серверною. Тут відбуваєтсья реєстрація у чаті та виведення повідомлень нліхом надсилання та отримання повідомлень за допомогою socket.io

Лістинг файлу App.js:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Simple Chat</title>

    <link rel="stylesheet" href="./style.css">

</head>

<body>

    <h1>Simple Chat</h1>

    <div id="userList"></div>

    <div id="messages"></div>

    <input type="text" id="usernameInput" placeholder="Enter your name">

    <button id="registerButton">Join Chat</button>

    <div id="chat">

        <p id="username-init"> </p>

        <input type="text" id="messageInput" placeholder="Type your message">

        <button id="sendButton">Send</button>

    </div>

    <script src="/socket.io/socket.io.js"></script>

    <script>

        const socket = io();

        let userColor = getRandomColor();

        let username;

        function getRandomColor() {

            const letters = '0123456789ABCDEF';

            let color = '#';

            for (let i = 0; i < 6; i++) {

                color += letters[Math.floor(Math.random() \* 16)];

            }

            return color;

        }

        socket.on('userList', (users) => {

            const userListDiv = document.getElementById('userList');

            userListDiv.innerHTML = `<p>Users online:&nbsp</p>${users.map(user => `<p style="color:${user.color}">${user.username}&nbsp</p>`).join('')}`;

        });

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

            const messagesDiv = document.getElementById('messages');

            const messageNode = document.createElement('p');

            messageNode.innerHTML = `<span class="user-message" style="color:${message.color}">${message.username}: </span>${message.text}`;

            messagesDiv.appendChild(messageNode);

        });

        document.getElementById('sendButton').addEventListener('click', () => {

            addMessage()

        });

        document.getElementById('messageInput').addEventListener('keydown', (event) => {

            if (event.key === 'Enter') {

                addMessage()

            }

        });

        document.getElementById('registerButton').addEventListener('click', () => {

            addUser()

        });

        document.getElementById('usernameInput').addEventListener('keydown', (event) => {

            if (event.key === 'Enter') {

                addUser()

            }

        });

        function addMessage() {

            const messageInput = document.getElementById('messageInput');

            const message = messageInput.value;

            if (message) {

                socket.emit('message', message);

                messageInput.value = '';

            }

        }

        function addUser() {

            username = usernameInput.value;

            username = username.replace(/[^a-zA-Z0-9]/g, '');

            if (username) {

                const usernameInput = document.getElementById('usernameInput');

                const registerButton = document.getElementById('registerButton');

                const messageInput = document.getElementById('messageInput');

                const sendButton = document.getElementById('sendButton');

                const messages = document.getElementById('messages');

                socket.emit('register', { username, color: userColor });

                registerButton.style.display = 'none';

                usernameInput.style.display = 'none';

                messageInput.style.display = 'inline-block';

                sendButton.style.display = 'inline-block';

                messages.style.display = 'block';

                const messagesDiv = document.getElementById('chat');

                const messageUsername = document.getElementById('username-init');

                const messageNode = document.createElement('p');

                messageNode.innerHTML = username;

                messageNode.style.display = 'inline-block';

                messageNode.style.background = userColor

                messageNode.id = 'username';

                messagesDiv.replaceChild(messageNode, messageUsername);

            }

        }

    </script>

</body>

</html>

Останнім файлом є style.css. В ньому описатні стилі компонентів веб-сторінкию

Лістинг файлу style.css:

/\* Add some CSS for user colors \*/

.user-message {

    font-weight: bold;

    color: #3498db;

    /\* User message text color \*/

}

#messageInput {

    display: none;

    width: 88%;

    padding: 10px;

    border: 2px solid #ccc;

    border-radius: 5px;

    font-size: 16px;

}

#sendButton {

    display: none;

    padding: 10px 20px;

    background-color: #3498db;

    color: #fff;

    border: none;

    border-radius: 5px;

    cursor: pointer;

    font-size: 16px;

}

#userList {

    display: flex;

    flex-direction: row;

    margin-bottom: 20px;

}

#userList p {

    margin-right: 10px;

    font-size: 14px;

}

h1 {

    color: #3498db;

}

#username {

    padding: 11px 20px 11px 20px;

    border-radius: 10px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

}

#usernameInput {

    width: 93%;

    padding: 10px;

    border: 2px solid #ccc;

    border-radius: 5px;

    font-size: 16px;

}

#registerButton {

    padding: 10px 20px;

    background-color: #3498db;

    color: #fff;

    border: none;

    border-radius: 5px;

    cursor: pointer;

    font-size: 16px;

}

#messages {

    display: none;

    height: 70vh;

    overflow-y: auto;

    padding: 10px;

    border: 2px solid #ccc;

    border-radius: 5px;

    background-color: #f2f2f2;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

}

#userList {

    margin-bottom: 10px;

    display: flex;

    flex-direction: row;

}

#userList p {

    margin-right: 10px;

    font-size: 16px;

    font-weight: bold;

}

input[type="text"] {

    width: 70%;

    padding: 10px;

    border: 2px solid #ccc;

    border-radius: 5px;

    font-size: 18px;

}

button#registerButton {

    padding: 10px 20px;

    background-color: #3498db;

    color: #fff;

    border: none;

    border-radius: 5px;

    cursor: pointer;

    font-size: 16px;

    margin-top: 10px;

}

button#sendButton {

    padding: 10px 20px;

    background-color: #3498db;

    color: #fff;

    border: none;

    border-radius: 5px;

    cursor: pointer;

    font-size: 16px;

    margin-top: 10px;

}

button:hover {

    background-color: #2187c9;

}

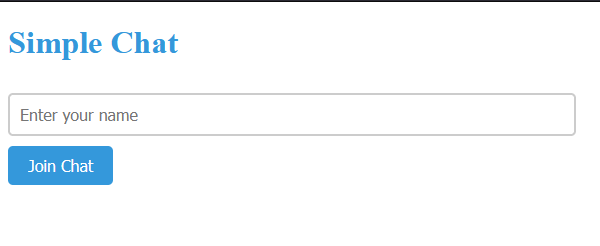
#userList p:hover {

    text-decoration: underline;

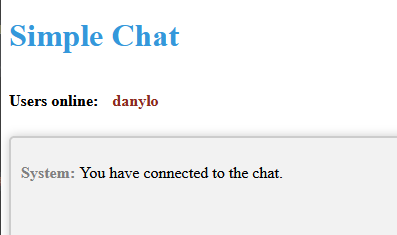
}

Отриманий результат:

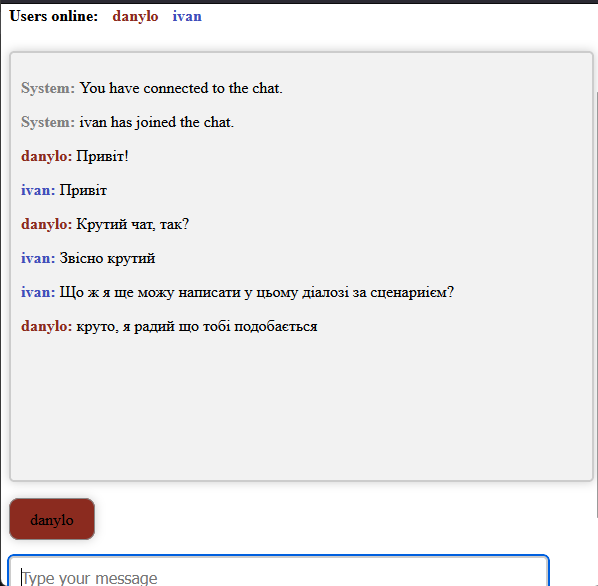
Сторінка при завантаженні:



Сторінка після реєстрації:



Сторінка після спілкування декількох користувачів:



Висновок:

В результаті виконання завдання ми дослідили роботу з веб-сокетами. Було реалізовано чат на основі Socket.io. У чаті можуть знаходитися багато користувачів одночасно. Кожен з них може надсилати повідомлення на отримувати повідомлення інших користувачів.

Посилання на репозиторій Git-hub: <https://github.com/DanyloTrofymov/web-basics/tree/lab4>