

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 роботи №4

курсу «Основи WEB - технологій »

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

Перевірив: Виконав:

Доц. Голубєв Л. П. Гр. ІТ-04

Чапча Святослав

Київ 2023

1. **Завдання**

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

1. **Хід роботи**

Наш сервер на index.js:

const express = require('express')

const app = express()

const http = require('http')

const cors = require('cors')

const { Server } = require('socket.io')

app.use(cors())

const server = http.createServer(app)

const io = new Server(server, {

  cors: {

    origin: 'http://localhost:3000',

    methods: ['GET', 'POST'],

  },

})

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

  console.log(`User Connected: ${socket.id}`)

  socket.on('join\_room', (data) => {

    socket.join(data)

    console.log(`User with ID: ${socket.id} joined room: ${data}`)

  })

  socket.on('send\_message', (data) => {

    socket.to(data.room).emit('receive\_message', data)

  })

  socket.on('disconnect', () => {

    console.log('User Disconnected', socket.id)

  })

})

server.listen(3001, () => {

  console.log('SERVER RUNNING')

})

Клієнт створений завдяки ReactJS, компонент App.js:

|  |
| --- |
| import "./App.css";  import io from "socket.io-client";  import { useState } from "react";  import Chat from "./Chat";  const socket = io.connect("http://localhost:3001");  function App() {    const [username, setUsername] = useState("");    const [room, setRoom] = useState("");    const [showChat, setShowChat] = useState(false);    const joinRoom = () => {      if (username !== "" && room !== "") {        socket.emit("join\_room", room);        setShowChat(true);      }    };    return (      <div className="App">        {!showChat ? (          <div className="joinChatContainer">            <h3>Join A Chat</h3>            <input              type="text"              placeholder="John..."              onChange={(event) => {                setUsername(event.target.value);              }}            />            <input              type="text"              placeholder="Room ID..."              onChange={(event) => {                setRoom(event.target.value);              }}            />            <button onClick={joinRoom}>Join A Room</button>          </div>        ) : (          <Chat socket={socket} username={username} room={room} />        )}      </div>    );  }  export default App; |

Компонент Chat.js:

import React, { useEffect, useState } from "react";

import ScrollToBottom from "react-scroll-to-bottom";

function Chat({ socket, username, room }) {

  const [currentMessage, setCurrentMessage] = useState("");

  const [messageList, setMessageList] = useState([]);

  const sendMessage = async () => {

    if (currentMessage !== "") {

      const messageData = {

        room: room,

        author: username,

        message: currentMessage,

        time:

          new Date(Date.now()).getHours() +

          ":" +

          new Date(Date.now()).getMinutes(),

      };

      await socket.emit("send\_message", messageData);

      setMessageList((list) => [...list, messageData]);

      setCurrentMessage("");

    }

  };

  useEffect(() => {

    socket.on("receive\_message", (data) => {

      setMessageList((list) => [...list, data]);

    });

  }, [socket]);

  return (

    <div className="chat-window">

      <div className="chat-header">

        <p>Live Chat</p>

      </div>

      <div className="chat-body">

        <ScrollToBottom className="message-container">

          {messageList.map((messageContent) => {

            return (

              <div

                className="message"

                id={username === messageContent.author ? "you" : "other"}

              >

                <div>

                  <div className="message-content">

                    <p>{messageContent.message}</p>

                  </div>

                  <div className="message-meta">

                    <p id="time">{messageContent.time}</p>

                    <p id="author">{messageContent.author}</p>

                  </div>

                </div>

              </div>

            );

          })}

        </ScrollToBottom>

      </div>

      <div className="chat-footer">

        <input

          type="text"

          value={currentMessage}

          placeholder="Hey..."

          onChange={(event) => {

            setCurrentMessage(event.target.value);

          }}

          onKeyPress={(event) => {

            event.key === "Enter" && sendMessage();

          }}

        />

        <button onClick={sendMessage}>&#9658;</button>

      </div>

    </div>

  );

}

export default Chat;

App.css

.App {

  width: 100vw;

  height: 100vh;

  background: #fff;

  color: #212121;

  font-family: "Open Sans", sans-serif;

  display: grid;

  place-items: center;

}

body {

  margin: 0%;

  padding: 0%;

}

@import url("https://fonts.googleapis.com/css?family=Open+Sans:300,400,700");

.joinChatContainer {

  display: flex;

  flex-direction: column;

  text-align: center;

}

.joinChatContainer h3 {

  font-size: 2.5rem;

  margin-bottom: 1rem;

}

.joinChatContainer input {

  width: 210px;

  height: 40px;

  margin: 7px;

  border: 2px solid #43a047;

  border-radius: 5px;

  padding: 5px;

  font-size: 16px;

}

.joinChatContainer button {

  width: 225px;

  height: 50px;

  margin: 7px;

  border: none;

  border-radius: 5px;

  padding: 5px;

  font-size: 16px;

  background: #43a047;

  color: #fff;

  cursor: pointer;

}

.joinChatContainer button:hover {

  background: #2e7d32;

}

.chat-window {

  width: 300px;

  height: 420px;

}

.chat-window p {

  margin: 0;

}

.chat-window .chat-header {

  height: 45px;

  border-radius: 6px;

  background: #263238;

  position: relative;

  cursor: pointer;

}

.chat-window .chat-header p {

  display: block;

  padding: 0 1em 0 2em;

  color: #fff;

  font-weight: 700;

  line-height: 45px;

}

.chat-window .chat-body {

  height: calc(450px - (45px + 70px));

  border: 1px solid #263238;

  background: #fff;

  position: relative;

}

.chat-window .chat-body .message-container {

  width: 100%;

  height: 100%;

  overflow-y: scroll;

  overflow-x: hidden;

}

.chat-window .chat-body .message-container::-webkit-scrollbar {

  display: none;

}

.chat-window .chat-body .message {

  height: auto;

  padding: 10px;

  display: flex;

}

.chat-window .chat-body .message .message-content {

  width: auto;

  height: auto;

  min-height: 40px;

  max-width: 120px;

  background-color: #43a047;

  border-radius: 5px;

  color: white;

  display: flex;

  align-items: center;

  margin-right: 5px;

  margin-left: 5px;

  padding-right: 5px;

  padding-left: 5px;

  overflow-wrap: break-word;

  word-break: break-word;

}

#you {

  justify-content: flex-start;

}

#you .message-content {

  justify-content: flex-start;

}

#you .message-meta {

  justify-content: flex-start;

  margin-left: 5px;

}

#other {

  justify-content: flex-end;

}

#other .message-content {

  justify-content: flex-end;

  background-color: cornflowerblue;

}

#other .message-meta {

  justify-content: flex-end;

  margin-right: 5px;

}

.message-meta #author {

  margin-left: 10px;

  font-weight: bold;

}

.chat-window .chat-body .message .message-meta {

  display: flex;

  font-size: 12px;

}

.chat-window .chat-footer {

  height: 40px;

  border: 1px solid #263238;

  border-top: none;

  display: flex;

}

.chat-window .chat-footer input {

  height: 100%;

  flex: 85%;

  border: 0;

  padding: 0 0.7em;

  font-size: 1em;

  border-right: 1px dotted #607d8b;

  outline: none;

  font-family: "Open Sans", sans-serif;

}

.chat-window .chat-footer button {

  border: 0;

  display: grid;

  place-items: center;

  cursor: pointer;

  flex: 15%;

  height: 100%;

  background: transparent;

  outline: none;

  font-size: 25px;

  color: lightgray;

}

.chat-window .chat-footer button:hover {

  color: #43a047;

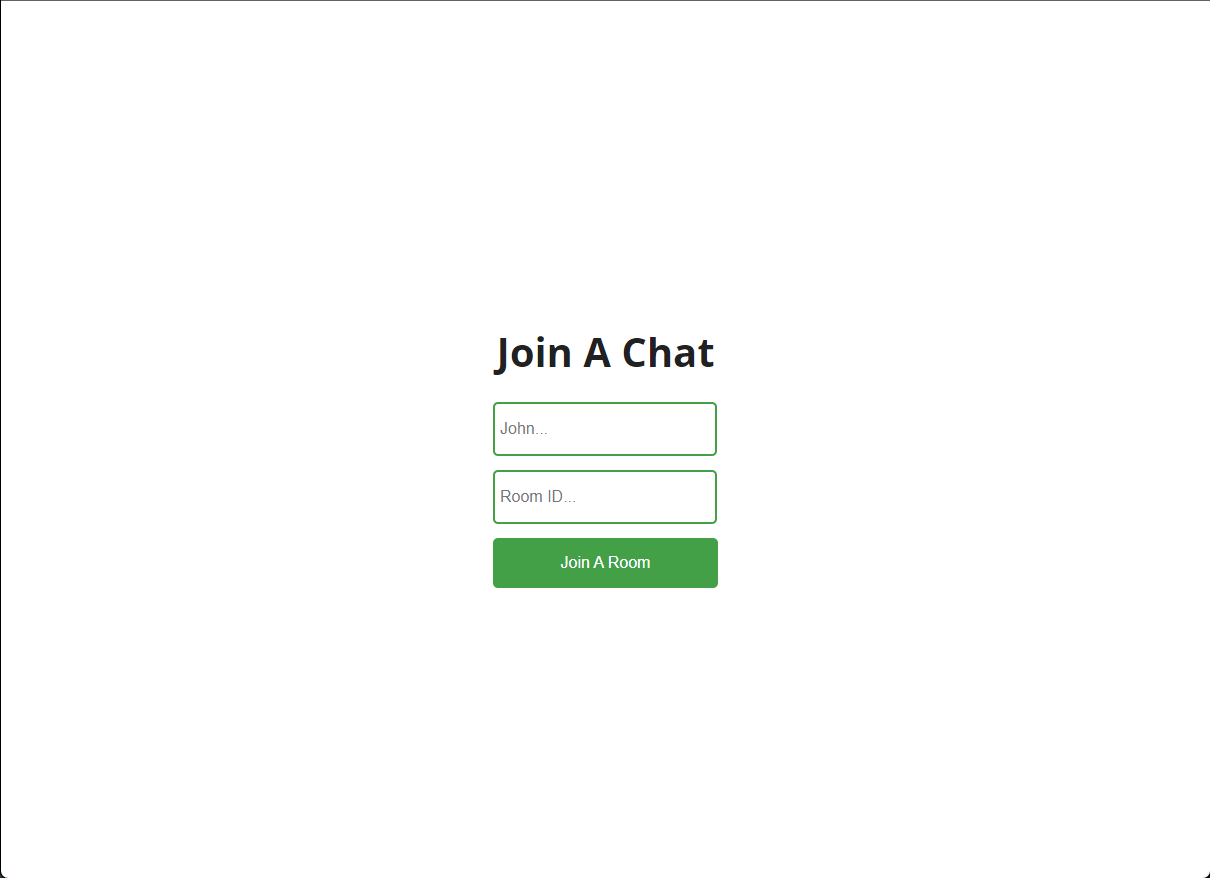
}

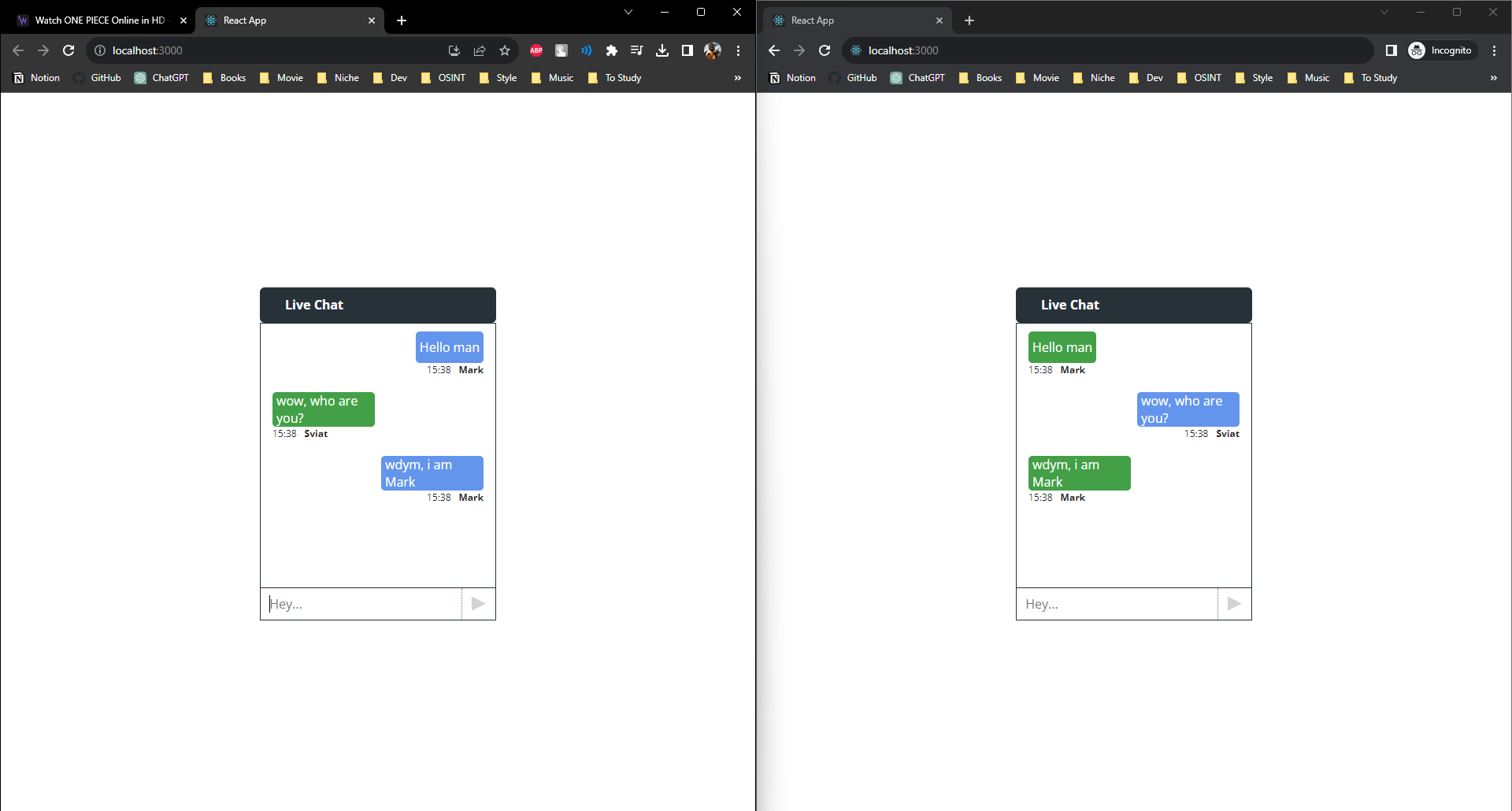
.hide {

  opacity: 0 !important;

}

**3. Отримані результати**





**Висновок**

Протягом лабораторної роботи було досліджено і розроблено додаток для обміну повідомленнями між учасниками в режимі реального часу за допомогою бібліотеки Socket.IO. Завдяки цій бібліотеці, цей додаток став дієвим засобом комунікації для користувачів у режимі реального часу. Він може бути використаний у різних сферах, таких як онлайн-чати, спільна робота над проектами або будь-яка інша ситуація, де важлива швидка та ефективна комунікація.

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