

Лабораторна робота 06.

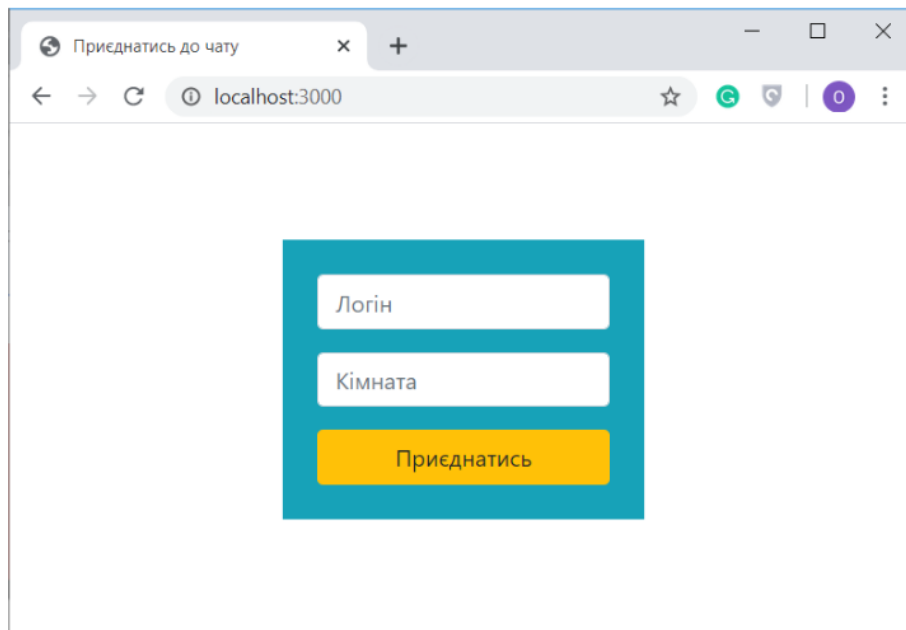
Протокол WebSocket. Використання Socket.io для розробки чат-додатків. Project03 “Chat Application”

Мета: ознайомитись з протоколом WebSocket, використанням Socket.io для розробки чат-додатків. Project03 “Chat Application”

Завдання до проекту (Лабораторна №6)

- Розробити додаток для обміну повідомленнями між учасниками в режимі реального часу
- Спілкування проводиться в кімнатах користувачами, що здійснили вхід в кімнату
- Про вхід чи вихід користувачів з кімнати необхідно надсилати повідомлення та оновлені дані кімнати
- Використати зразки сторінок на наступних слайдах

Сторінка для приєднання до чат-кімнати



					ДУ «Житомирська політехніка».24.121.13.000 - Лр6			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Клосович І.А.			Звіт з лабораторної роботи		Літ.	Арк.
Перевір.		Сидорчук В.О.						Аркушів
Керівник							1	7
Н. контр.							ФІКТ Гр. ІІЗ-22-1[1]	
Зав. каф.								

Зразок сторінки для спілкування в чат-кімнаті

Події, що емітуються клієнтом:

join – приєднання нового користувача до чат-кімнати

sendMessage – для надсилання повідомлення користувачем

Події, що емітуються сервером:

message – для відправлення повідомлення сервером

roomData – для відправлення даних про користувачів у кімнаті

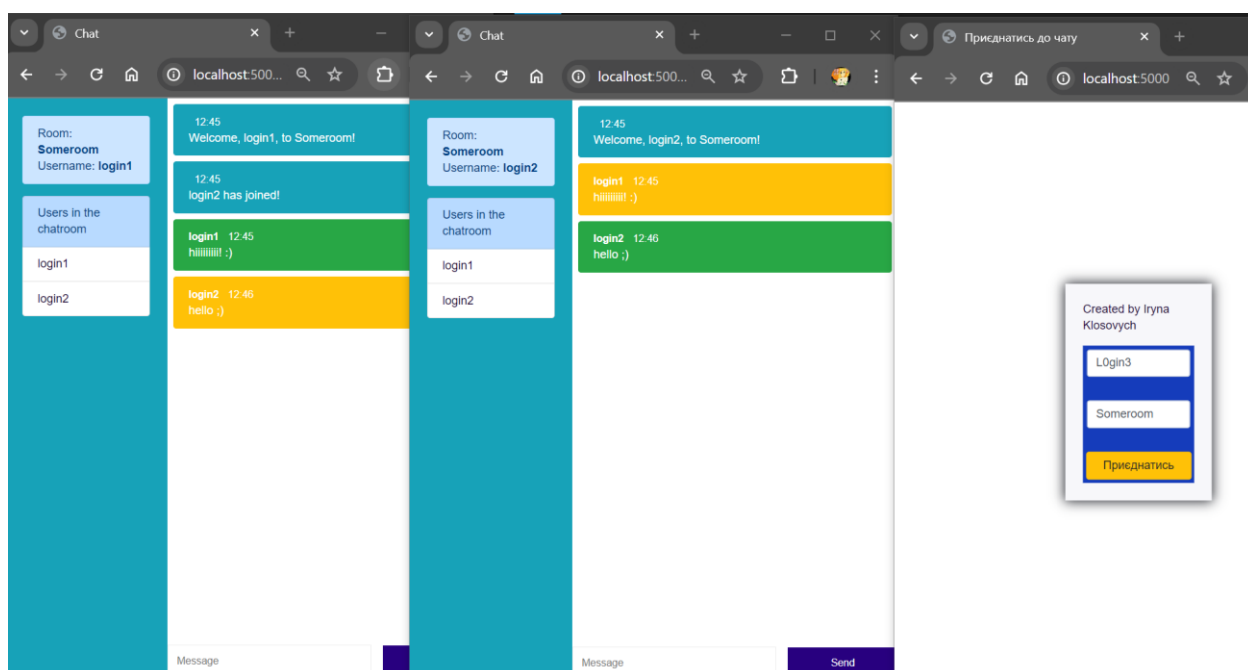
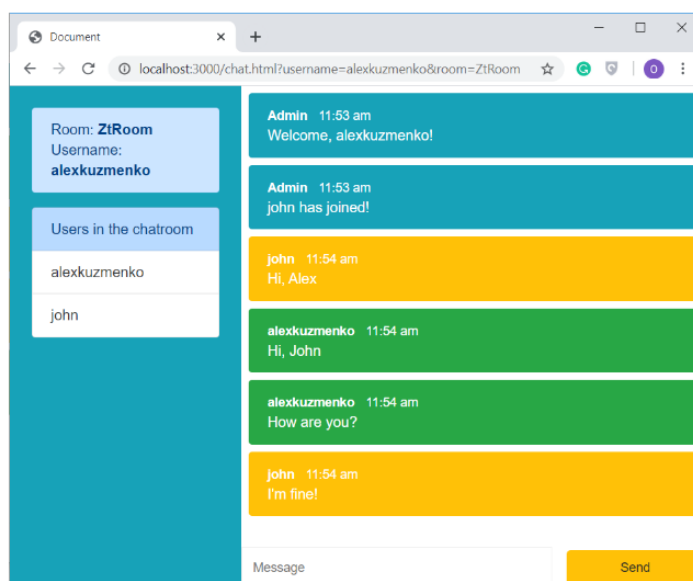


Рис. 1. Результат: реєстрація, вхід в кімнату, обмін повідомлень

		Клосович І.А.			ДУ «Житомирська політехніка».24.121.13.000 - Лр6	Арк.
		Сидорчук В.О.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

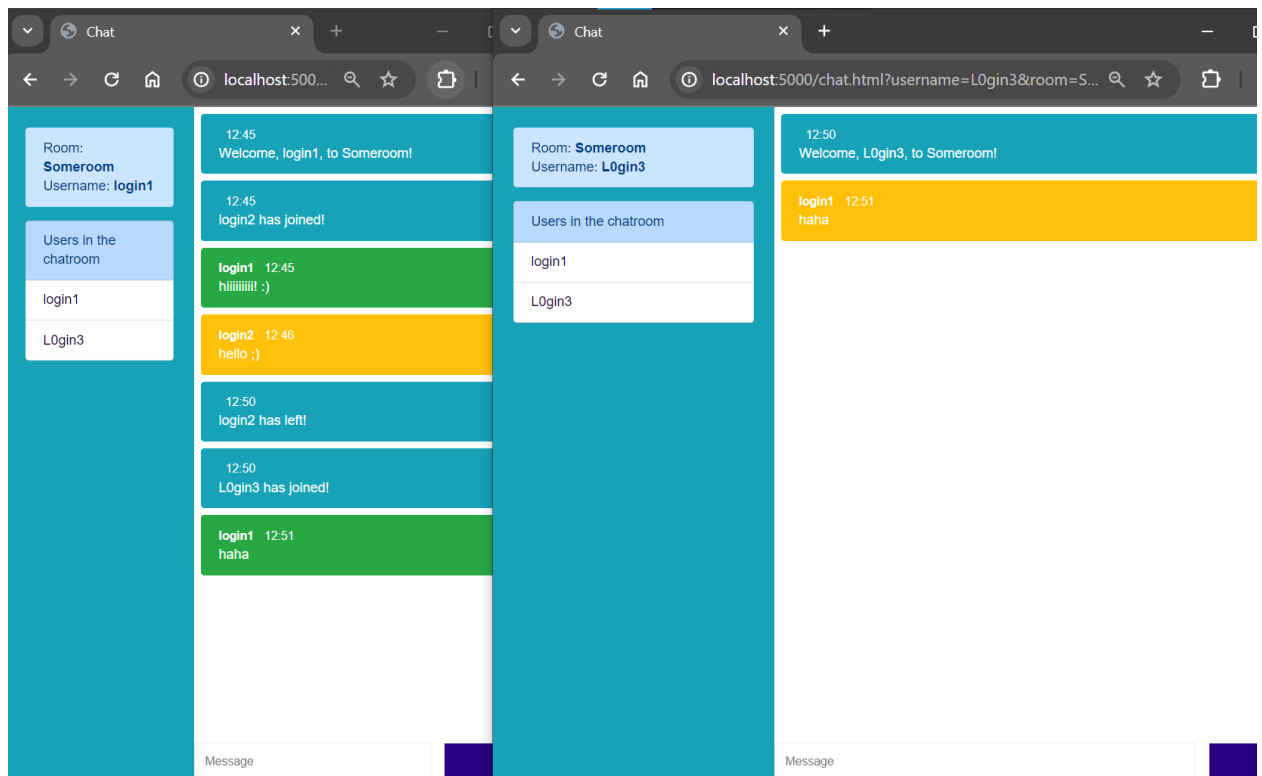


Рис. 2. Результат: вихід одного користувача. вхід іншого, обмін повідомлень

Завдання до проекту (самостійно)

- Забезпечити внутрішній скролінг для контенту з повідомленнями та даними про кімнати (авто прокручування до останнього повідомлення)
- Додати функціональність “{user} is typing” (тривалістю 2 секунди): всі користувачі в межах кімнати отримують відповідну подію з повідомленням.
 - Крок1. Клієнт відправляє на сервер подію “userIsTyping”
 - Крок2. Сервер слухає подію “userIsTyping” та ретранслює її на всі інші клієнти в межах даної кімнати
 - Крок3. Клієнт слухає подію “userIsTyping” та відображає повідомлення в інформаційному блоці
- Додати функціональність для приватних повідомлень: в повідомленні якщо написати “@username: text”

		Клосович І.А.			ДУ «Житомирська політехніка».24.121.13.000 - Лр6	Арк.
		Сидорчук В.О.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<> chat.html U X  <> index.html U  JS app.js U  JS users.js U  JS messag
Lab6_ChatApp > public > <> chat.html > html > body > script > socket.on('messag
  2  <html lang="en">
 11  <body>
 65  <script>
 84      socket.on('message', (message) => {
 94          const html = Mustache.render(messageTemplate, {
          createdAt: moment(message.createdAt).format('HH:mm')
 98      })
 99      $messages.insertAdjacentHTML('beforeend', html)
100      $messages.scrollTop = $messages.scrollHeight;
101  })
102  })
103

# style.css U X
Lab6_ChatApp > public > styles > # style.css > ...
180  .users {
184  }
185
186  #messages {
187      max-height: 100%;
188      overflow-y: auto;
189      border: 1px solid #ccc;
190      padding: 10px;
191  }

```

Рис. 3. Результат: код автоскролінгу

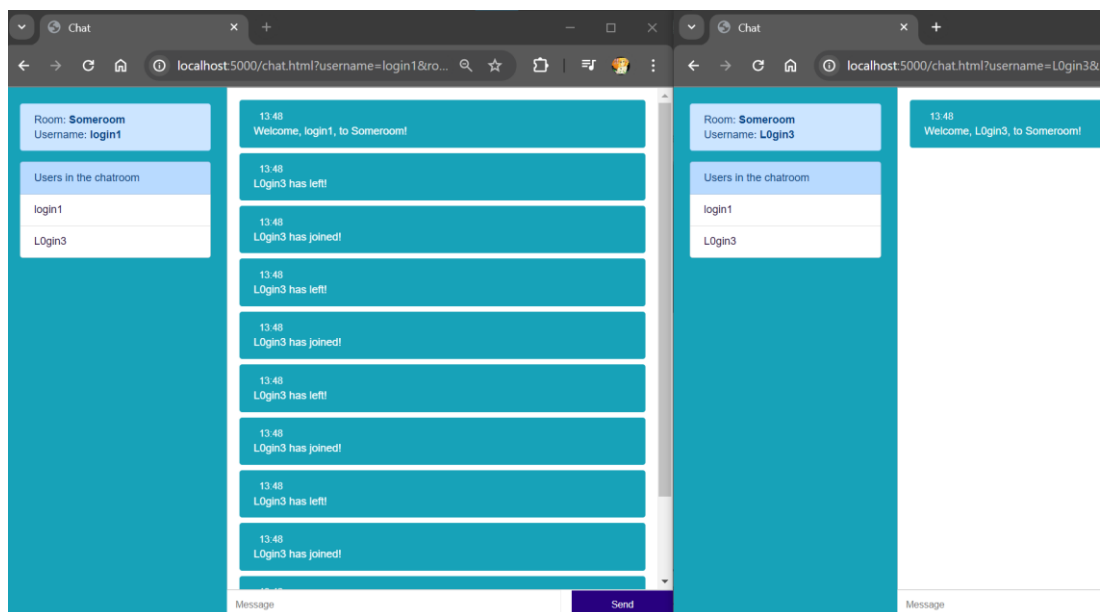


Рис. 4. Результат

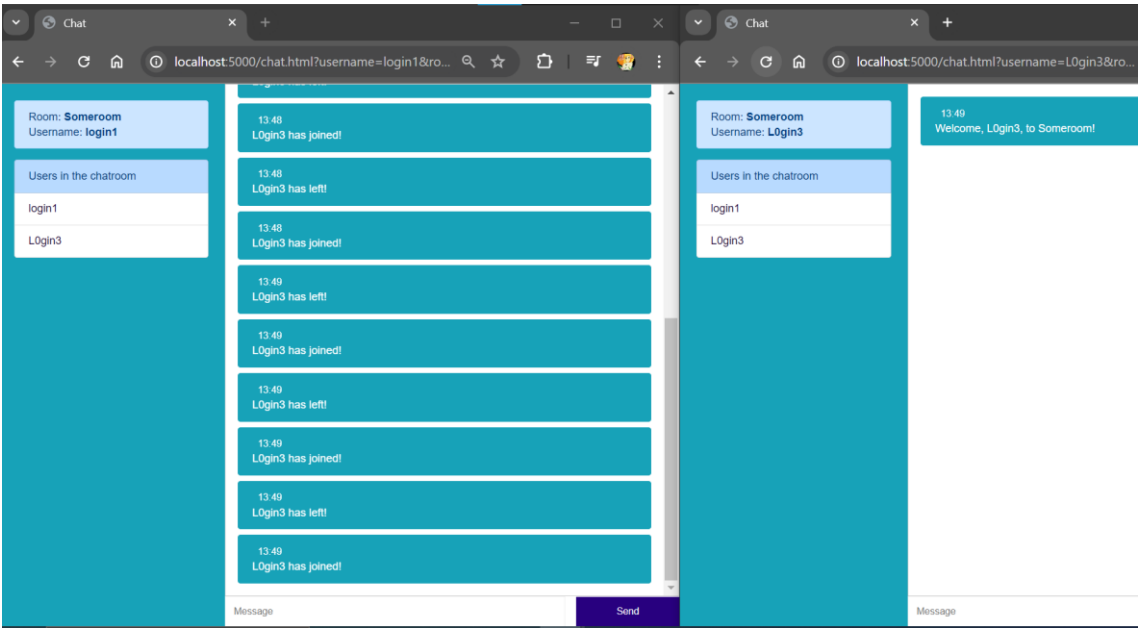


Рис. 5. Результат

```
140 function sendTypingMessage() {
141   socket.emit('userIsTyping');
142 }
143 document.querySelector("input[name='message']").addEventListener('input', () => {
144   clearTimeout(typingTimer);
145   typingTimer = setTimeout(sendTypingMessage, doneTypingInterval);
146 });
147
148 socket.on('messageTyping', (message) => {
149   console.log(message);
150   if(document.querySelectorAll('.typing').length==0){
151     let bgClass = 'text-black typing';
152     const html = Mustache.render(messageTemplate, {
153       message: `${message} is typing...`,
154       bgClass: bgClass,
155     });
156     $messages.insertAdjacentHTML('beforeend', html);
157     $messages.scrollTop = $messages.scrollHeight;
158     setupClearTypingTimer();
159   }
160 })
161
162 function clearTypingMessage() {
163   // ...
164 }
165
166 socket.on('sendMessage', (message, callback) => {
167   console.log(socket);
168   const user = getUser(socket.id);
169   io.to(user.room).emit('message', generateMessage(user.username,
170     callback));
171 });
172
173 socket.on('userIsTyping', () => {
174   const user = getUser(socket.id);
175   console.log(user.username);
176   socket.broadcast.to(user.room).emit('messageTyping', user.username);
177 });
```

Рис. 6. Результат: код userIsTyping

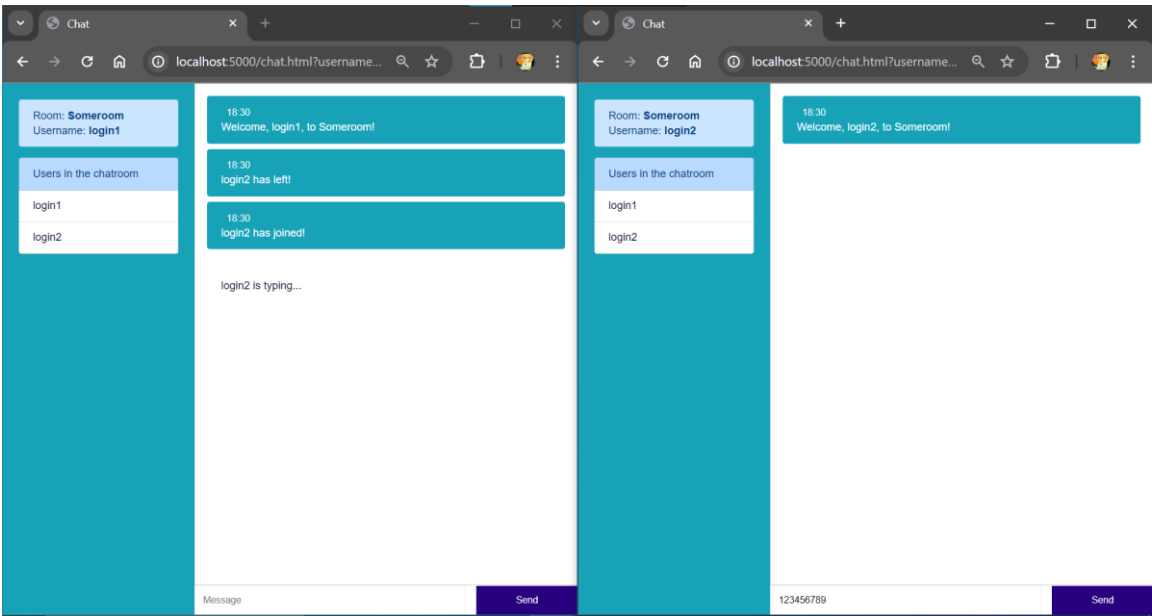


Рис. 7. Результат userIsTyping

```

111 $messageForm.addEventListener('submit', (e) => {
112
113   const message = e.target.elements.message.value
114   if (!message.startsWith('@')) {
115     socket.emit('sendMessage', message, (error) => {
116       $messageFormButton.removeAttribute('disabled')
117       $messageFormInput.value = ''
118       $messageFormInput.focus()
119
120       if (error) {
121         return console.log(error)
122       }
123
124       console.log('Message delivered!')
125     })
126   }
127   else {
128     const atIndex = message.indexOf(':');
129     if (atIndex !== -1) {
130       const username = message.slice(1, atIndex).trim();
131       const newMessage = message.slice(atIndex + 1).trim();
132       socket.emit('privateMessage', { to: username, text: newMessage });
133       $messageFormButton.removeAttribute('disabled')
134       $messageFormInput.value = ''
135       $messageFormInput.focus()
136     }
137     if (error) {
138       return console.log(error)
139     }
140   }
141 }
142
143
144
42
43
44
45 socket.on('userIsTyping', () => {
46   const user = getUser(socket.id)
47   console.log(user.username)
48   socket.broadcast.to(user.room).emit('messageTyping', user.u
49 })
50
51 socket.on('privateMessage', (data, callback) => {
52   console.log(data)
53   const recipientSocket = io.sockets.sockets.get(getUserByName(data.to))
54   console.log(recipientSocket)
55   const user = getUser(socket.id)
56   if (recipientSocket) {
57     // Відправити приватне повідомлення до отримувача
58     recipientSocket.emit('message', generateMessage(user.username, data.text))
59     callback()
60   }
61 })
62
63 //Відключення
64 socket.on('disconnect', () => {
65   console.log('user is disconnected')
66   const user = removeUser(socket.id)
67
68   if (user) {
69     io.to(user.room).emit('message', generateMessage('', 'user disconnected'))
70     io.to(user.room).emit('roomData', {
71       room: user.room,
72       users: getRoomUsers(user.room)
73     })
74   }
75 })

```

Рис. 8. Основний код до завдання з надси­лан­ням по­відом­лен­ня кон­крет­но­му ко­ри­сту­вачу

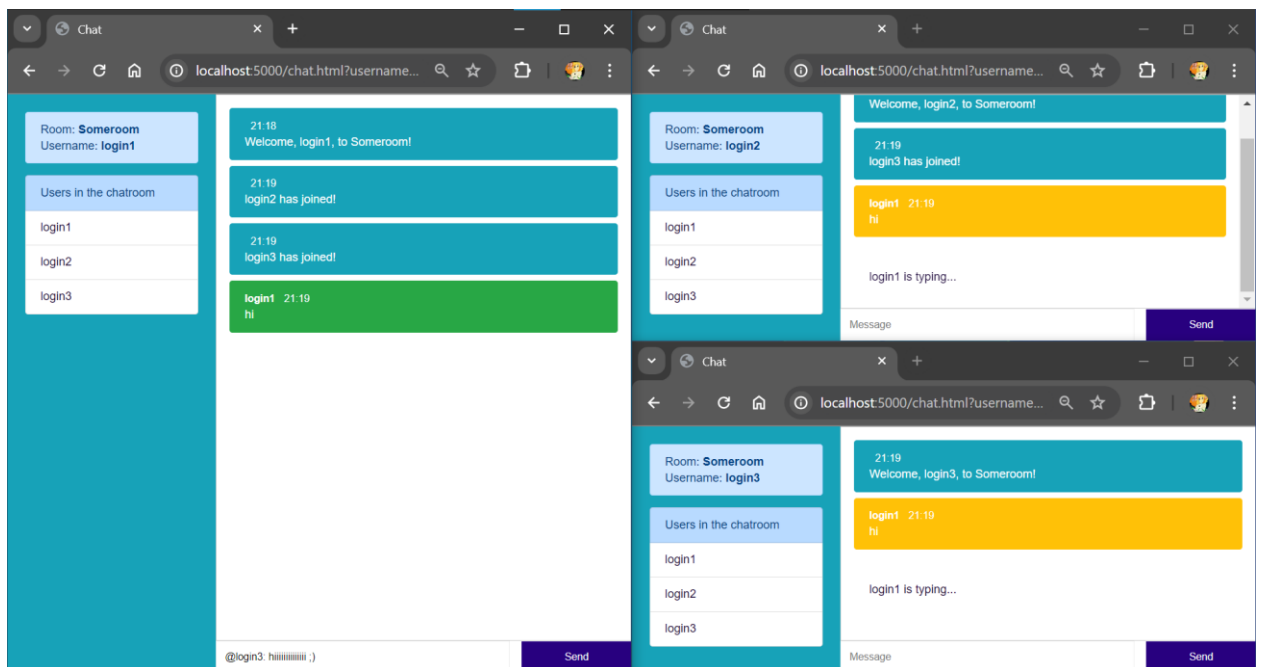


Рис. 9. Результат до надси­лан­ня по­відом­лен­ня пев­но­му ко­ри­сту­вачу

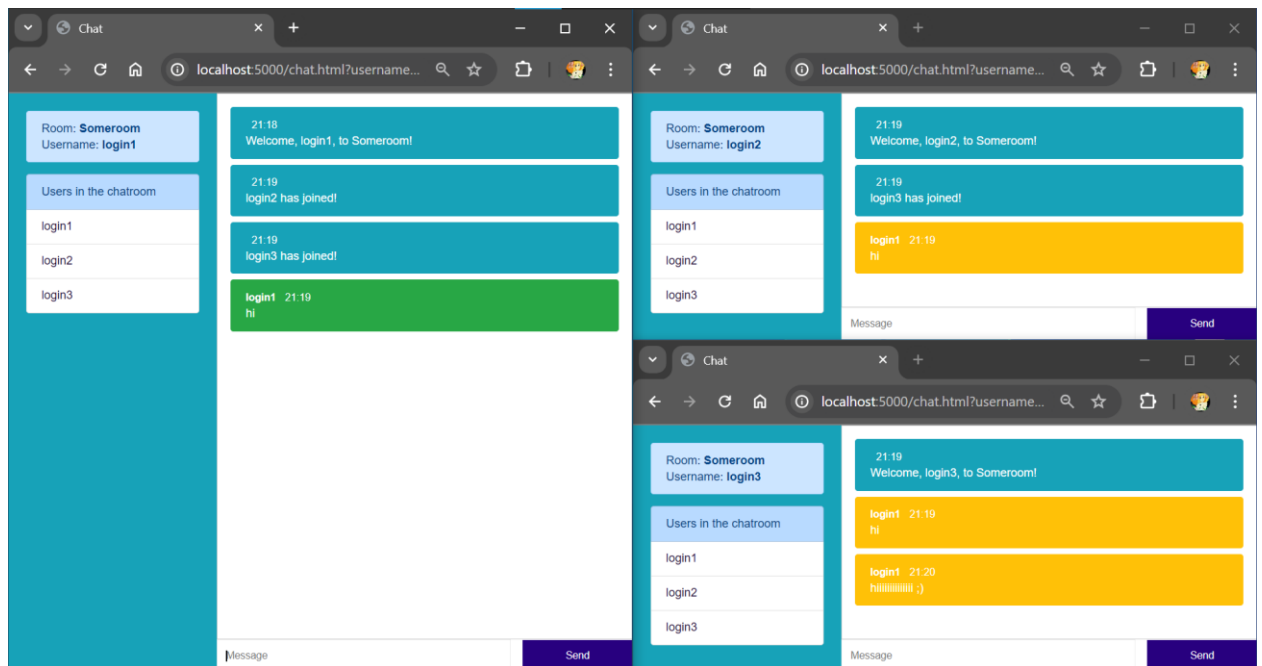


Рис. 10. Результат після

Висновок: на лабораторному занятті ми ознайомились з протоколом WebSocket, використанням Socket.io для розробки чат-додатків. Розробили Project03 “Chat Application”

		Клосович І.А.			ДУ «Житомирська політехніка».24.121.13.000 - Лр6	Арк.
		Сидорчук В.О.				7
Змн.	Арк.	№ докум.	Підпис	Дата		