МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА АНОНІМНОГУ ЧАТУ З ВИКОРИСТАННЯМ SOCKET.IO»**

Виконав: студент 4 курсу, групи 6.1219-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
І.В. Ткаченко
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Горбенко В.І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ 07 ” 02 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Ткаченку Івану Володимировичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка анонімного чату з використанням Socket.IO

керівник роботи Горбенко Віталій Іванович, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 26 » січня 2023 року № 102-с

2. Строк подання студентом роботи 07.06.2023 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка анонімного чату з використанням Socket.IO.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 07.02.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.02.2023	
2.	Збір вихідних даних.	17.02.2023	
3.	Обробка методичних та теоретичних	10.03.2023	
	джерел.		
4.	Розробка першого та другого розділу.	14.04.2023	
5.	Розробка третього розділу.	17.05.2023	
6.	Оформлення та нормоконтроль	01.06.2023	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	23.06.2023	

Студент _____ І.В. Ткаченко
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.І. Горбенко
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка анонімного чату з використанням Socket.IO»: 38 с., 25 рис., 9 джерел.

АНОНІМНІСТЬ, ВЕБ-ДОДАТОК, ВЗАЄМОДІЯ, ИНТЕРНЕТ, РОЗМОВА, САЙТ, ЧАТ.

Об’єкт дослідження даної роботи – розробка анонімного чату з використанням socket.io.

Мета роботи – розробити функціональний анонімний чат з використанням socket.io та навчитись використовувати його на практиці.

Метод дослідження – практичний, порівняльний.

Socket.io – це бібліотека JavaScript, яка дозволяє створювати реальнічасові додатки, зокрема анонімні чати. Вона базується на протоколі WebSocket та дозволяє створювати зручні та ефективні додатки з реальнічасовим спілкуванням.

У даній роботі будуть розглянуті основні можливості бібліотеки, такі як створення кімнат для спілкування, відправка повідомлень, обробка подій та інші. Буде розроблено анонімний чат, який дозволить користувачам спілкуватись між собою за допомогою веб-браузера.

Варто відзначити, що застосування socket.io в розробці анонімних чатів може бути досить ефективним. Використання даної бібліотеки дозволяє створювати додатки з реальночасовим спілкуванням, що є дуже зручним та популярним серед користувачів.

SUMMARY

Bachelor’s qualifying paper «Development of Anonymous Chat using Socket.IO»: 38 p., 25 figures, 9 references.

ANONYMITY, WEB APPLICATION, INTERACTION, INTERNET, CONVERSATION, WEBSITE, CHAT.

The object of this study is the development of an anonymous chat using Socket.io

The aim of this work is to develop a functional anonymous chat using Socket.io and to learn how to use it in practice.

The research method used is practical and comparative.

Socket.io is a JavaScript library that allows the creation of real-time applications, including anonymous chats. It is based on the WebSocket protocol and enables the creation of convenient and efficient applications with real-time communication.

This work will consider the basic capabilities of the library, such as creating chat rooms, sending messages, event handling, and more. An anonymous chat will be developed that will allow users to communicate with each other through a web browser.

It is worth noting that the use of Socket.io in the development of anonymous chats can be quite effective. The use of this library allows the creation of applications with real-time communication, which is very convenient and popular among users.

ЗМІСТ

ВСТУП

В сучасному світі інформаційні технології постійно розвиваються і знаходять все більше і більше застосувань у нашому житті. Одним з таких застосувань є чат, який має велике значення для спілкування в режимі реального часу. Але водночас існує проблема безпеки, оскільки в інтернеті можна зустріти багато шахраїв та зловмисників, які можуть шкодити користувачеві.

Тому дедалі більше користувачів шукають захист своєї особистості від інших. Розробники створюють різноманітні інструменти для захисту особистої інформації, одним з яких є анонімний чат. Як правило, він забезпечує конфіденційність спілкування від третіх осіб, тож користувач може бути впевнений в безпеці своїх повідомлень.

У даному дипломному проєкті буде розглянуто процес розробки анонімного чату, який буде створено з використанням бібліотеки JavaScript socket.io. Ця бібліотека дозволяє забезпечити двосторонній зв'язок між клієнтом та сервером, тому її вважають одним з найкращих інструментів для розробки різноманітних додатків.

Основною метою даного дипломного проєкту є розробка функціонального анонімного чату з можливістю забезпечення безпеки його користувачів. Для досягнення такої мети буде розглянуто основні можливості бібліотеки socket.io, розроблено серверну та клієнтську частини додатку, проведені тести та проведений аналіз отриманих результатів.

Крім того, в рамках проєкту буде розглянуто найбільш поширені проблеми при розробці анонімного чату та методи їх вирішення. Також буде розроблено інтерфейс користувача, що дозволить зручно користуватися чатом.

1 РОБОТА З SOCKET.IO

1.1 Аналіз та опис сутностей

Розробка веб-сайтів на JavaScript є досить поширеною та важливою сферою в IT-галузі. Цей мова програмування використовується для створення динамічних інтерактивних веб-сайтів, які можуть працювати на будь-якому пристрої. Для того, щоб розпочати розробку сайту на JavaScript, необхідно мати знання HTML та CSS, оскільки вона використовується для створення структури та оформлення сайту відповідно. Потім можна розпочати розробку на JavaScript, додавши інтерактивність та функціональність до сайту.

Основна перевага розробки на JavaScript полягає в тому, що ця мова програмування є дуже гнучкою та може використовуватися для різних цілей, що дозволяє розробникам створювати різноманітні веб-додатки, ігри, мобільні додатки та багато іншого. Крім того, JavaScript - це мова програмування, яка має широку спільноту розробників та велику кількість ресурсів, що дозволяє швидко вирішувати проблеми та знайти відповіді на питання.

Ще однією важливою перевагою розробки на JavaScript є наявність великої кількості бібліотек та фреймворків, що значно спрощують розробку та зменшують час, необхідний для створення веб-сайту. Наприклад, такі фреймворки, як React, Angular та Vue, роблять розробку веб-сайтів на JavaScript набагато простішою та ефективнішою.

Мої колеги Українські програмісти є дуже компетентними у розробці веб-сайтів на JavaScript. Україна має велику кількість талановитих IT-фахівців, які здатні створювати дуже складні та інноваційні проєкти. Багато українських IT-компаній пропонують послуги з розробки веб-сайтів на JavaScript та забезпечують високу якість роботи. Крім того, українські програмісти активно беруть участь у міжнародних проєктах та співпрацюють з колегами з усього

світу.

Розробка веб-сайтів на JavaScript є важливою та цікавою галуззю, яка має великий потенціал для подальшого розвитку. За допомогою JavaScript можна створювати різноманітні веб-додатки, ігри, мобільні додатки та багато іншого. Ця мова програмування продовжує розвиватися та знаходити нові застосування, тому розробникам на JavaScript є чим зайнятися та розвиватися у своїй професійній діяльності.

Найбільш поширеним фреймворком для розробки на JavaScript є React. Він дозволяє створювати високоефективні та динамічні веб-сайти з використанням компонентів. Angular є ще одним популярним фреймворком для розробки веб-додатків на JavaScript. Він дозволяє швидко створювати веб-сайти з використанням шаблонів та компонентів. Vue – це ще один фреймворк, який швидко набирає популярності. Він дозволяє створювати високоефективні та легкі веб-сайти з використанням компонентів та роутінгу.

Отже, розробка веб-сайтів на JavaScript є важливою складовою IT-галузі, яка продовжує розвиватися та знаходити нові застосування.

1.2 Socket.io бібліотека

Socket.io – це бібліотека JavaScript для реалізації двостороннього зв'язку між клієнтом та сервером на основі протоколу WebSocket. Вона дозволяє створювати інтерактивні веб-додатки, які можуть передавати дані в режимі реального часу, що дозволяє користувачам отримувати оновлення без необхідності оновлювати сторінку [1].

Socket.io використовується для створення різних типів додатків, включаючи чати, онлайн ігри, спільну роботу над документами та інші типи додатків, де важливо, щоб всі користувачі бачили оновлення в режимі реального часу.

Для початку використання Socket.io необхідно встановити його на сервері та клієнті. Для цього можна використовувати менеджер пакетів npm. Далі, ви можете створити сервер та підключити Socket.io до нього (рис. 1.1) [1].

```
const app = require('express')();
const http = require('http').createServer(app);
const io = require('socket.io')(http);
```

Рисунок 1.1 – Приклад підключення

Після цього ви можете встановити з'єднання з клієнтом та передавати дані між сервером та клієнтом з використанням Socket.io. Для створення з'єднання з клієнтом можна використати подію connection, яка зберігає об'єкт socket, який використовується для взаємодії з клієнтом (рис. 1.2) [1].

```
io.on('connection', (socket) => {
  console.log('a user connected');
  socket.on('message', (msg) => {
    console.log('message: ' + msg);
    io.emit('message', msg);
  });
});
```

Рисунок 1.2 – Приклад створення з'єднання

Код вище (рис. 1.2) встановлює з'єднання з клієнтом та слухає подію `message`, коли користувач надсилає повідомлення. Після цього він передає повідомлення всім підключеним клієнтам з використанням методу `io.emit()`.

Socket.io також дозволяє використовувати кімнати для групування клієнтів та передачі повідомлень тільки в певні кімнати. Це корисно, коли потрібно передавати повідомлення тільки до певної групи користувачів (рис. 1.3) [1].

```
io.on('connection', (socket) => {
  socket.join('room1');
  socket.on('message', (msg) => {
    io.to('room1').emit('message', msg);
  });
});
```

Рисунок 1.3 – Приклад створення кімнати

Код на рисунку 1.3 створює кімнату з назвою `room1` та приєднує користувача до неї. Після цього, якщо користувач надсилає повідомлення, він передає його всім користувачам, які приєдналися до кімнати `room1`.

Одним з інших корисних функцій є можливість обробки помилок. Якщо під час передачі даних виникає помилка, Socket.io може повернути її на сервер або клієнт для подальшої обробки (рис 1.4) [1].

```
io.on('connection', (socket) => {
  socket.on('error', (err) => {
    console.log('received error from client:', err);
  });
});
```

Рисунок 1.4 – Приклад обробки помилок

Socket.io також підтримує передачу файлів між клієнтом та сервером. Для цього можна використовувати метод socket.emit() на клієнті та метод socket.on() на сервері (рис. 1.5) [1].

Socket.io також дозволяє використовувати middleware для обробки запитів до серверу. Middleware можна використовувати для перевірки авторизації користувача, обробки запиту та інших операцій (рис. 1.6) [1].

Таким чином використання Socket.io може значно полегшити розробку веб-додатків, які потребують двостороннього зв'язку між клієнтом та сервером.

```
// на клієнті
const fileInput = document.querySelector('input[type="file"]');
fileInput.addEventListener('change', (event) => {
  const file = event.target.files[0];
  const reader = new FileReader();
  reader.onload = () => {
    socket.emit('file', {
      filename: file.name,
      data: reader.result
    });
  };
  reader.readAsBinaryString(file);
});

// на сервері
io.on('connection', (socket) => {
  socket.on('file', (data) => {
    console.log(`received file ${data.filename}`);
    // збереження файлу на сервері
  });
});
```

Рисунок 1.5 – Приклад передачі файлів

```
io.use((socket, next) => {
  if (socket.request.headers.cookie) {
    // перевірка авторизації користувача
    return next();
  }
  next(new Error('Authentication error'));
});

io.on('connection', (socket) => {
  // обробка запиту
});
```

Рисунок 1.6 – Приклад middleware

1.3 React бібліотека

Я вирішив використовувати Socket.io та React разом для створення чату. Основна причина – це можливість створення інтерактивних та динамічних чатів з реальним часом оновлення.

Поєднання Socket.io та React дозволяє нам створити інтерактивний та динамічний чат з реальним часом оновлення. Ми можемо надсилати повідомлення з клієнта на сервер за допомогою Socket.io, а потім оновлювати відповідні компоненти React на клієнтській стороні. Це дозволяє нам побудувати чат, який оновлюється в реальному часі та показує повідомлення користувачів без перезавантаження сторінки.

Таким чином, використання Socket.io та React дозволяє нам створювати інтерактивні та динамічні чати з реальним часом оновлення, що забезпечує високу якість користування додатком.

React – це бібліотека для створення інтерфейсів користувача, яка була розроблена компанією Facebook і з'явилася в 2013 році. Основною метою React є полегшення розробки веб-додатків з великою кількістю динамічного контенту [2].

React на сьогоднішній день є однією з найпопулярніших бібліотек для створення інтерфейсів користувача, завдяки своїм унікальним можливостям та безлічі корисних функцій. Багато великих компаній, таких як Netflix, Instagram, WhatsApp та багато інших, використовують React для розробки своїх веб-додатків.

Однією з найбільших переваг React є його можливість створювати компоненти. Компонент – це невеликий елемент інтерфейсу, який може відрендерити свій вміст та має свій власний стан. Компоненти можуть бути вкладені один в одного, що дозволяє створювати складні інтерфейси з невеликої кількості коду.

Одна з основних переваг React – це використання віртуального DOM. Він дозволяє створювати віртуальне представлення сторінки, яке рендериться

швидше, ніж звичайний DOM. Крім того, React автоматично оновлює тільки ті елементи, які змінилися, що забезпечує покращення продуктивності додатка [2].

Крім того, React має дуже активну спільноту розробників, що забезпечує швидкий розвиток технології та наявність великої кількості корисних модулів та бібліотек. Завдяки цьому, розробники можуть швидко знаходити та використовувати інструменти, що дозволяють їм ефективно працювати з React. Наприклад, Redux є однією з найбільш поширених бібліотек для керування станом додатка в React.

Ще однією з переваг React є можливість використання JSX – мови програмування, яка дозволяє описувати інтерфейс користувача за допомогою HTML-подібного синтаксису. Це дозволяє розробникам швидко створювати складні інтерфейси без необхідності вручну вписувати HTML-код [2].

В цілому, React – це потужна технологія для створення інтерактивних веб-додатків з великою кількістю динамічного контенту. Він дозволяє розробникам швидко створювати складні інтерфейси, зменшуючи кількість коду, що потрібно написати. Якщо ви шукаєте бібліотеку, яка дозволить вам ефективно розробляти веб-додатки, варто розглянути можливості React [2].

1.4 Висновки до розділу 1

Розробка анонімного чату – це відповідальний процес, який потребує розуміння сучасних технологій. У даному випадку, для створення анонімного чату будуть використані такі технології, як socket.io та react. Ці інструменти є найбільш популярними та функціональними у своїй сфері, так як вони дуже прості у використанні та забезпечують швидку та якісну розробку.

Socket.io – це відкрита бібліотека для розробки додатків, яка дозволяє створювати з'єднання між клієнтом та сервером, щоб передавати дані у режимі реального часу. Це забезпечує швидку і безперебійну роботу додатка, що особливо важливо в анонімному чаті [1].

React – це бібліотека JavaScript, яка використовується для створення інтерактивних інтерфейсів користувача. Вона дозволяє створювати складні інтерфейси за допомогою простих компонентів, що забезпечує зручний та ефективний процес розробки [2].

Таким чином, використання socket.io та react для розробки анонімного чату – це найбільш оптимальний вибір, який дозволяє створити якісний та швидкий додаток. На додаток до цих інструментів, будуть використані інші технології, які допоможуть зробити чат ще більш функціональним та зручним для користувачів.

2 ПРОЄКТУВАННЯ АНОНІМНОГО ЧАТУ

2.1 Вимоги до проєкту

Для дослідження предметної області з використанням socket.io було обрана тема розробки анонімного чату. Проєкт передбачає створення веб-додатку, де користувачі можуть обговорювати теми, що їх цікавлять без необхідності реєстрації. Основна мета проєкту – створити простий та зручний веб-додаток для спілкування користувачів. Проєкт передбачає розробку функцій, таких як створення кімнат для спілкування, можливість відправляти повідомлення та переглядати історію розмов. Розробка цих функцій дозволить зробити проєкт максимально зручним та привабливим для користувачів, що дозволить привернути більше уваги до проєкту та збільшити його популярність.

Основні функціональні вимоги до проєкту:

– можливість анонімного входу в чат без реєстрації з використанням лише нікнейму;
– можливість обміну повідомленнями в режимі реального часу без перезавантаження сторінки;
– підтримка мобільних пристроїв та різних браузерів для зручного використання чату;
– доступність чату через HTTPS-протокол для захисту від зловмисників та забезпечення безпеки взаємодії користувачів;
– можливість додавання нових функцій та розширення можливостей чату в майбутньому;
– зручний та зрозумілий інтерфейс.

2.2 Додатки, використані при розробці

Visual Studio Code – це безкоштовний текстовий редактор, що розробляється компанією Microsoft. Його можна встановити на операційні системи Windows, macOS та Linux. Visual Studio Code має великий список функцій, що робить його одним з найбільш популярних текстових редакторів серед розробників [3].

Переваги Visual Studio Code:

– безкоштовний: Visual Studio Code можна використовувати безкоштовно, що є великою перевагою для початківців та студентів;
– кроссплатформовий: редактор підтримує операційні системи Windows, macOS та Linux;
– багата функціональність: Visual Studio Code має великий список функцій, таких як автодоповнення, налаштування тем та шрифтів, підтримка Git, настройка лінтерів та багато іншого;
– розширюваність: редактор можна легко розширювати за допомогою розширень, які можна знайти в маркетплейсі Visual Studio Code;
– легкий у використанні: Visual Studio Code має інтуїтивний і легкий у використанні інтерфейс, що дозволяє зосередитися на розробці програмного забезпечення.

Узагальнюючи, Visual Studio Code – це безкоштовний, кроссплатформовий текстовий редактор з великим списком функцій та легким у використанні інтерфейсом. Його можна рекомендувати для початківців та досвідчених розробників програмного забезпечення [3].

Node.js – це відкрите програмне забезпечення, яке використовується для розробки серверних додатків на JavaScript. Node.js використовується на багатьох веб-сайтах, таких як LinkedIn, Yahoo та PayPal, оскільки він дозволяє розробникам створювати швидкі, масштабовані та ефективні додатки для обробки великої кількості запитів [4].

Розглянемо основні переваги Node.js.

Висока швидкість та продуктивність: Node.js використовує двигун V8, який розроблений Google для виконання JavaScript. Це дозволяє Node.js оброблювати більшу кількість запитів за один раз порівняно з традиційними серверами. Крім того, Node.js дозволяє виконувати багато задач одночасно, що поліпшує продуктивність додатків.

Легкий та простий у використанні: Node.js є легким та простим у використанні засобом розробки серверів. Він використовує JavaScript, який є однією з найпопулярніших мов програмування, тому багато розробників вже знають його. Крім того, Node.js має велику кількість модулів та пакетів, що дозволяє розробникам швидко створювати додатки без необхідності писати багато коду з нуля.

Масштабованість: Node.js дозволяє розробникам створювати масштабовані додатки, які можуть оброблювати велику кількість запитів одночасно. Node.js має подієвий цикл, який дозволяє обробляти багато запитів одночасно без блокування виконання інших запитів.

Відкрите програмне забезпечення: Node.js – це відкрите програмне забезпечення, що означає, що ви можете користуватися ним безкоштовно та змінювати його за своїми потребами. Крім того, Node.js має велику спільноту розробників, яка постійно розвиває його та додає нові функції та можливості.

Node.js – це потужний засіб для розробки серверних додатків на JavaScript. Він дозволяє розробникам створювати швидкі, масштабовані та ефективні додатки, які можуть оброблювати велику кількість запитів одночасно [4].

2.3 Додаткові бібліотеки, фреймворки та плагіни

Redux – це бібліотека управління станом додатків, яка дозволяє легко керувати даними і подіями в додатку. Він дозволяє зберігати стан додатку в одному місці, що полегшує його керування та розробку. Redux має три головні

принципи: єдиний джерело істини, стан є тільки для читання та зміни стану за допомогою чистих функцій [5].

Основними елементами Redux є стор (store), дії (actions) та зменшувачі (reducers). Стор зберігає стан додатку, дії описують зміни стану, а зменшувачі змінюють стан додатку на основі дій [5].

Одним з прикладів використання Redux в React може бути додаток зі списком завдань, де користувач може додавати та видаляти завдання. У цьому випадку, стор міститиме список завдань, дії будуть описувати додавання та видалення завдань, а зменшувачі будуть змінювати стан стору, коли відбувається дія.

Щоб використовувати Redux в React, спочатку необхідно ініціалізувати стор у файлі index.js, де ми будемо рендерити наш додаток. Після цього необхідно створити дії та зменшувачі для кожної операції в додатку. Далі, необхідно створити контейнерні компоненти, які будуть зв'язувати наші компоненти зі стором. За допомогою connect() методу з бібліотеки react-redux, ми можемо отримати доступ до стору та дій в наших компонентах [5].

Отже, Redux – це потужна бібліотека, яка значно полегшує керування станом додатків в React. Використовуючи Redux, ми можемо зберігати стан додатку в одному місці та зменшити кількість передачі пропсів між компонентами [5].

React Hooks – це функції, що дозволяють використовувати функціональні компоненти React зі станом, без використання класів. Hooks дозволяють нам використовувати стейт і інші функції React, не використовуючи класовий компонент [2].

JSHint – це інструмент статичного аналізу JavaScript-коду, який допомагає виявляти помилки та невідповідності в коді, що може призвести до проблем у роботі програми. Плагін JSHint для різних редакторів коду дозволяє використовувати цей інструмент безпосередньо в робочому процесі, при цьому відразу ж отримуючи зворотний зв'язок про можливі проблеми [6].

JSHint перевіряє код на відповідність певним правилам (rules), які можна

налаштувати відповідно до потреб проєкту. Ці правила визначають, які частини коду мають бути написані, які конструкції використовувати та які не використовувати, а також які змінні мають бути оголошені та як [6].

Крім цього, JSHint може використовуватися для перевірки коду на відповідність стандартам кодування, таким як CommonJS та AMD [6].

Плагін JSHint дозволяє налаштовувати правила перевірки, вибирати рівень строгості перевірки, а також виводити результати аналізу у різних форматах.

Використання плагіна JSHint значно підвищує якість коду та спрощує процес дебагу, що робить його необхідною складовою будь-якого проєкту на JavaScript.

Reactjs – це бібліотека для створення інтерфейсів для веб-додатків. Reactjs code snippets – це короткий фрагмент коду, який можна використовувати для створення компонентів в Reactjs.

Reactjs code snippets дозволяють значно скоротити час на написання коду, зберігаючи при цьому точність і правильність написання коду. Вони дуже корисні для початківців, які тільки навчаються Reactjs, а також для досвідчених розробників, які хочуть зберегти час на написання коду.

Reactjs code snippets можуть бути знайдені в різних інтернет-ресурсах, таких як Github, NPM, а також можуть бути встановлені в популярних текстових редакторах, таких як Visual Studio Code, Atom, Sublime і т.д.

Різні Reactjs code snippets можуть включати в себе функції компонентів, імпорти, експорти, обробники подій, роутери та багато іншого. Користувачі можуть вибирати з багатьох різних варіантів, що найкраще відповідає їхнім потребам.

Узагалі, Reactjs code snippets є потужним інструментом для розробки веб-додатків, який дозволяє розробникам швидко та ефективно створювати високоякісний код.

Debugger for Chrome – це інструмент, який дозволяє налагоджувати JavaScript-код безпосередньо в браузері Google Chrome. Він надає різноманітні функції для аналізу та виправлення помилок у коді, а також допомагає в

розумінні, як працює код [7].

Основні функції Debugger for Chrome включають:

– встановлення точок зупинки: розробник може встановити точки зупинки в коді, щоб зупинити виконання коду в певній точці та проаналізувати поточний стан додатка;
– інспектування змінних: Debugger for Chrome дозволяє розробникам переглядати значення змінних в будь-якій точці виконання коду, що допомагає виявляти помилки та проблеми в коді;
– покрокове виконання коду: інструмент надає можливість поетапного виконання коду, щоб краще зрозуміти, як працює додаток;
– налагодження подій: Debugger for Chrome дозволяє налагоджувати події, такі як кліки миші, завантаження сторінок та інші, що допомагає виявляти проблеми з користувацьким інтерфейсом.

Debugger for Chrome є потужним інструментом для налагодження JavaScript-коду в браузері Google Chrome. Він може значно скоротити час, витрачений на налагодження коду, та допомогти розробникам створювати більш якісні та надійні додатки [7].

Express.js - це фреймворк для розробки веб-додатків на мові JavaScript, що працює на платформі Node.js. Express.js дозволяє зручно створювати серверну частину додатків шляхом забезпечення набору утиліт та функцій, що дозволяють більш просто та ефективно взаємодіяти з HTTP-протоколом [8].

Розглянемо основні переваги Express.js.

Простота використання та швидкість розробки. Express.js надає базовий набір функцій та утиліт, що дозволяє швидко розпочати розробку веб-додатку.

Розширюваність. Express.js дозволяє використовувати додаткові модулі та бібліотеки, що дозволяє розширити функціональність додатку.

Маршрутизація. Express.js надає зручний механізм маршрутизації запитів, що дозволяє зручно обробляти HTTP-запити.

Розглянемо ключові концепції Express.js.

Маршрутизація. Маршрутизація в Express.js передбачає зв'язок між URL-

адресою запиту та функцією, яка повинна бути викликана для обробки запиту.

Middleware. Middleware – це функції, які виконують певні дії з запитом перед тим, як запит буде оброблений. Middleware може бути викликаний для кожного запиту або тільки для певних запитів.

Шаблонізація. Express.js надає можливість використовувати різні шаблонізатори для генерації HTML-сторінок.

2.4 Підсумкова покрокова схема проєкту

Налаштування середовища розробки:

- встановити Node.js;
- завантажити та встановіти Node.js з офіційного веб-сайту;
- вибрати потрібну версію Node.js для своєї операційної системи та запустити установник;
- перевірити, чи Node.js успішно встановлено;
- встановлення React та створення нового проєкту;
- перейдіти до бажаного каталогу, де я хочу створити проєкт;
- відкрити термінал та ввести наступну команду для створення нового проєкту React: npx create-react-app chat-app (ця команда створить нову папку з назвою "chat-app" із початковими файлами та налаштуваннями проєкту React);
- після завершення команди перейти в папку проєкту, ввести команду cd chat-app, а потім запустіть проєкт, ввести команду npm start (це запустить сервер React та відкрити його у браузері за замовчуванням).

Створення серверної частини:

- створення сервера за допомогою Node.js та Express.js;
- відкрити новий термінал та перейти до папки проєкту;
- ініціалізувати новий проєкт Node.js, ввести команду npm init;
- встановити фреймворк Express.js, ввести команду npm install express;

– створіти новий файл з назвою server.js кореневій папці проєкту;
– відкрийте цей файл у текстовому редакторі та підключіть модуль Express.js;
– встановлення та налаштування бібліотеки socket.io;
– встановіти бібліотеку socket.io, ввести команду npm install socket.io;
– додати код для налаштування socket.io;
– налаштування маршрутизації сервера;
– визначити основний маршрут сервера, який буде обробляти запити клієнта;
– додати код для обробки основного маршруту;
– додати код для обробки події з'єднання клієнта з сервером;
– додати код для запуску сервера на визначеному порту 3000.

Створення клієнтської частини:

– створення базового шаблону React-компонентів;
– перейти до папки проєкту React;
– видалити лишні файли;
– створити нові файли;
– відкрити файли та додати базовий код;
– підключення бібліотеки socket.io-client;
– відкрити App.js та очистити наявний код;
– додати свій код;
– відкрийте index.js та оновити код.

Обмін повідомленнями:

– реалізувати функціонал для надсилання повідомлень з клієнта на сервер і навпаки;
– налаштуйти обробку подій socket.io для прийому та відправлення повідомлень між клієнтом та сервером;
– відображати отримані повідомлення на клієнтській стороні.

Тестування та налагодження:

– перевірити роботу чату, відправляючи повідомлення та перевіряючи, чи

вони правильно приходять на сервер і відображаються на клієнті;

– виправити помилки та оптимізувати код для забезпечення більшої стабільності та продуктивності.

Схема ілюстрація представлена на рисунку 2.1.

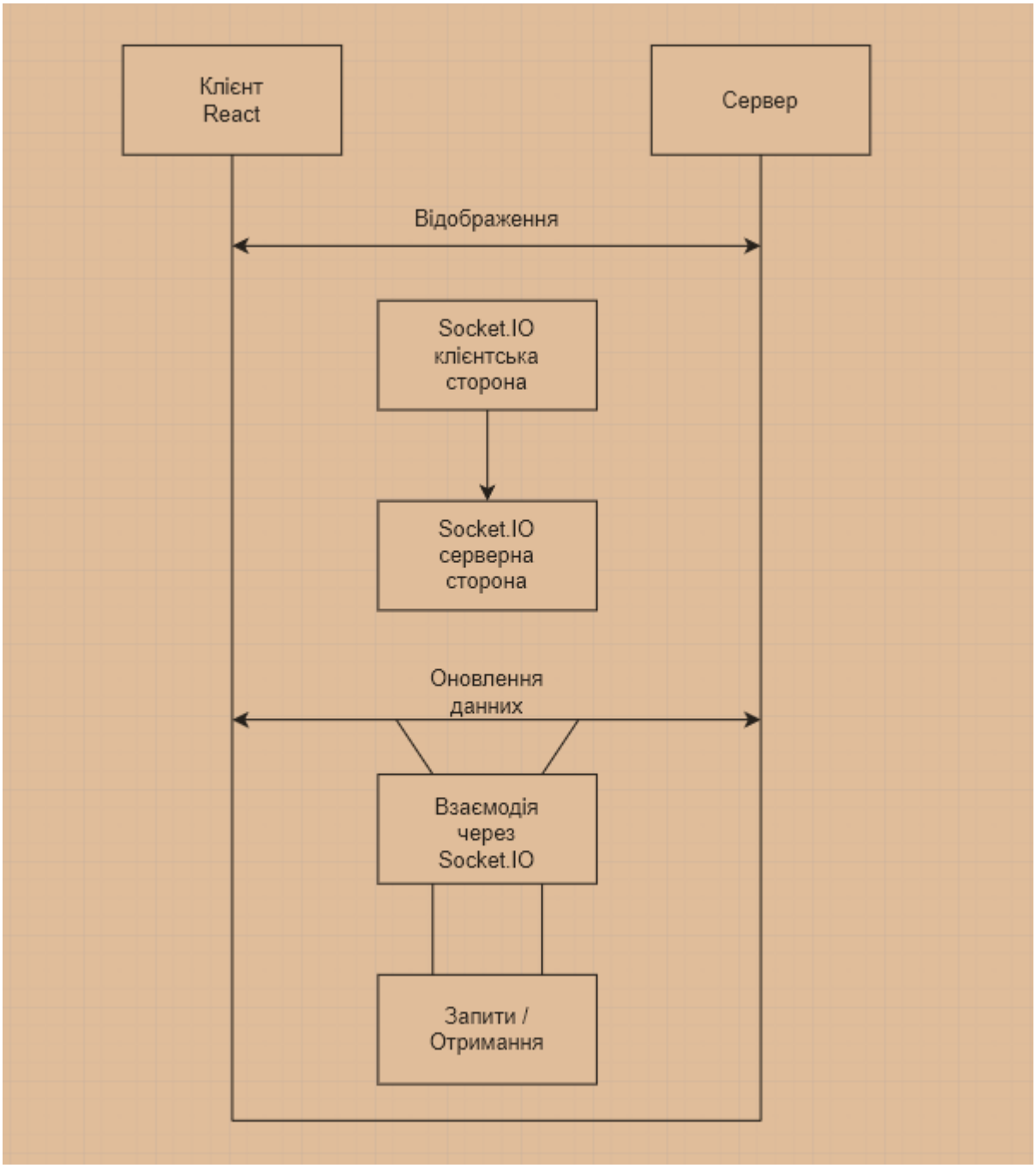

Рисунок 2.1 – Приклад схема

3 РОЗРОБКА АНОНІМНОГО ЧАТУ

3.1 Початкові налаштування проєкту та розробка client

Спочатку проєкту потрібно встановити React і всі залежності та плагіни для нього які я використовуватиму. Після чого я отримаю файл package.json де зберігатимуться всі залежності та їх версії. Далі в цьому файлі я пропишу scripts. Вони мені знадобляться для швидкого запуску проєкту в різних режимах і етапах розробки. У кінцевому етозі на виході я отримаю файл (рис. 3.1).

```
{
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@material-ui/core": "^4.11.2",
    "@material-ui/icons": "^4.11.2",
    "@testing-library/jest-dom": "^5.11.4",
    "@testing-library/react": "^11.1.0",
    "@testing-library/user-event": "^12.1.10",
    "formik": "^2.2.6",
    "react": "^17.0.1",
    "react-dom": "^17.0.1",
    "react-redux": "^7.2.2",
    "react-router-dom": "^5.2.0",
    "react-scripts": "4.0.1",
    "redux": "^4.0.5",
    "redux-saga": "^1.1.3",
    "socket.io-client": "^3.0.5",
    "web-vitals": "^0.2.4"
  },
  "scripts": {
    "start": "react-scripts --openssl-legacy-provider start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

Рисунок 3.1 – Файл package.json

Як я вже зазначив, також у цьому файлі є чотири скрипти:

- "start": "react-scripts --openssl-legacy-provider start": відповідальний за запуск проєкту;
- "build": "react-scripts build": відповідальний за складання проєкту;
- "test": "react-scripts test": відповідальний за тестування проєкту;
- "eject": "react-scripts eject": команда "eject" витягує налаштування react-scripts та видаляє їх із проєкту, які більше не пов'язані з react-scripts. Це означає, що ви повністю контролюєте налаштування вашої програми і можете налаштувати її, як вам завгодно.

Далі я почав написання інтерфейсу. Я наперед визначився з дизайном і почав писати компонети. У React, компоненти – це незалежні блоки коду, які можуть бути використані повторно для створення користувацького інтерфейсу. Компоненти можуть бути створені як функції або класи, які приймають вхідні параметри, називаються `props`, і повертають React-елементи, які потім відображаються на сторінці.

Компоненти дозволяють розділяти користувацький інтерфейс на окремі частини, що спрощує їх розробку та підтримку. Вони можуть бути використані для створення простих елементів, таких як кнопки та форми, і для більш складних інтерфейсів, таких як таблиці даних та списки.

У моєму додатку три основні компоненти (рис. 3.2):

- ChatPage - Сторінка самого чату;
- NotFoundPage - Сторінка, яка не була знайдена;
- UserForm - Форма входу в сам чат.

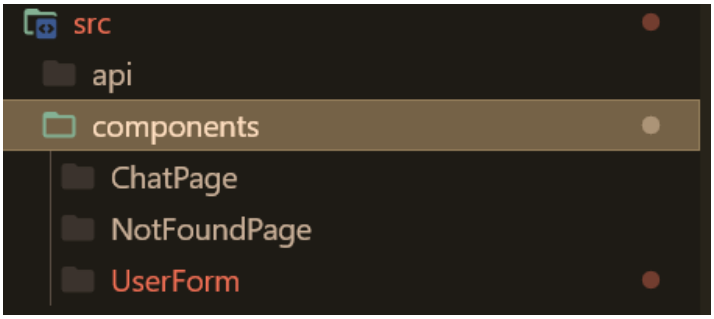

Рисунок 3.2 – Копоненти у додатку

Після опрацювання всіх функцій та магії коду. Сторінка форми входу виглядає так (рис. 3.3) та (рис. 3.4).

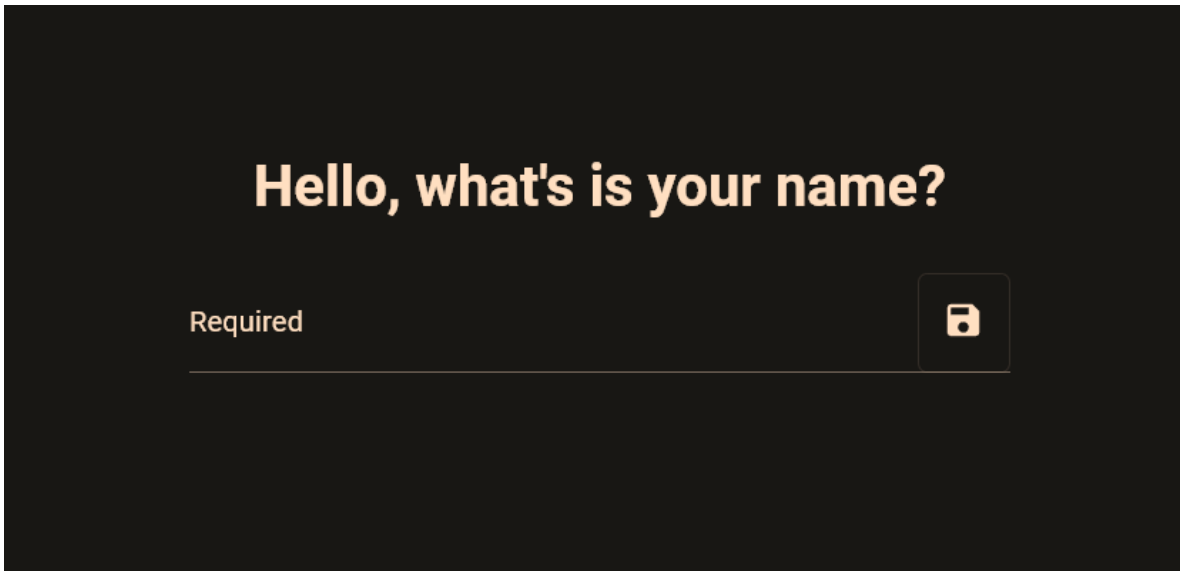

Рисунок 3.3 – Форма входу до чату

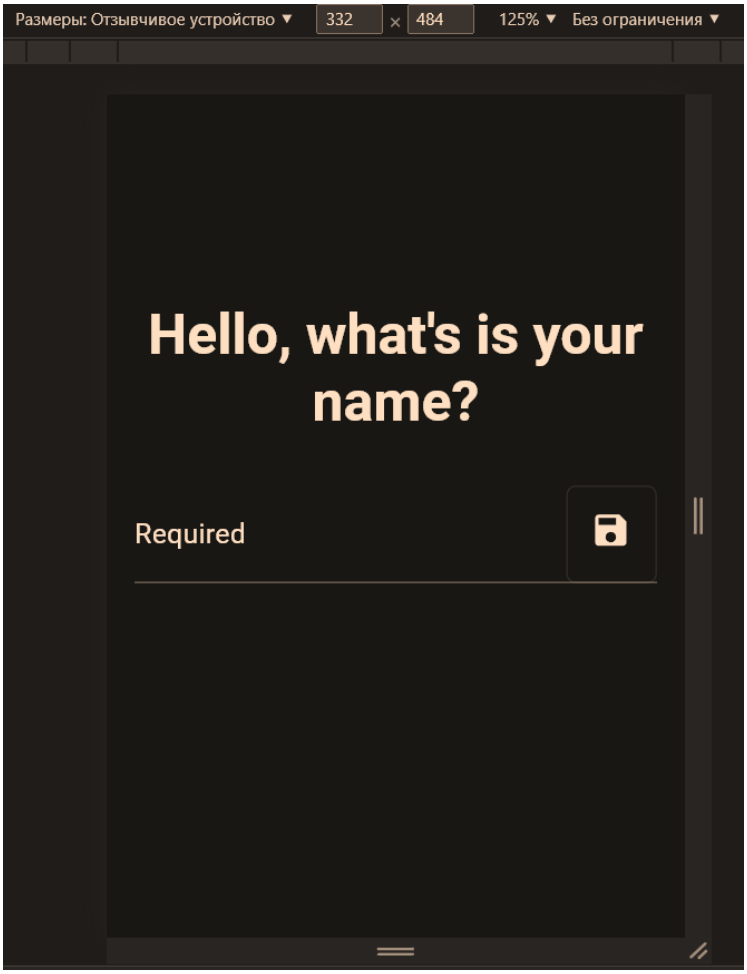

Рисунок 3.4 – Форма входу до чату на мобільному пристрої

Передбачається, що користувач повинен ввести свій логін або будь-який інший текст так як це анонімний чат і натиснути на дискету для збереження та переходу в сам чат.

Після опрацювання всіх функцій та магії коду. Головна сторінка чату виглядає так (рис. 3.5).

Рисунок 3.5 – Головна сторінка чату

Як видно на (рис. 3.5) та (рис. 3.6) у верхівці написано кількість учасників чату. Знизу форма для надсилання повідомлень.

Після написання інших множинних функцій у клієнта мені залишається не забути написати простенький api для підключення майбутнього сервера.

API (або інтерфейс програмування додатків) – це набір інструментів, правил та протоколів, які дозволяють різним програмам взаємодіяти між собою. API може бути порівняно зі службою доставки, яка переносить інформацію з одного місця в інше. За допомогою API, програми можуть обмінюватись даними між собою, що дозволяє їм працювати разом та забезпечувати користувача потрібними функціями та сервісами [9].

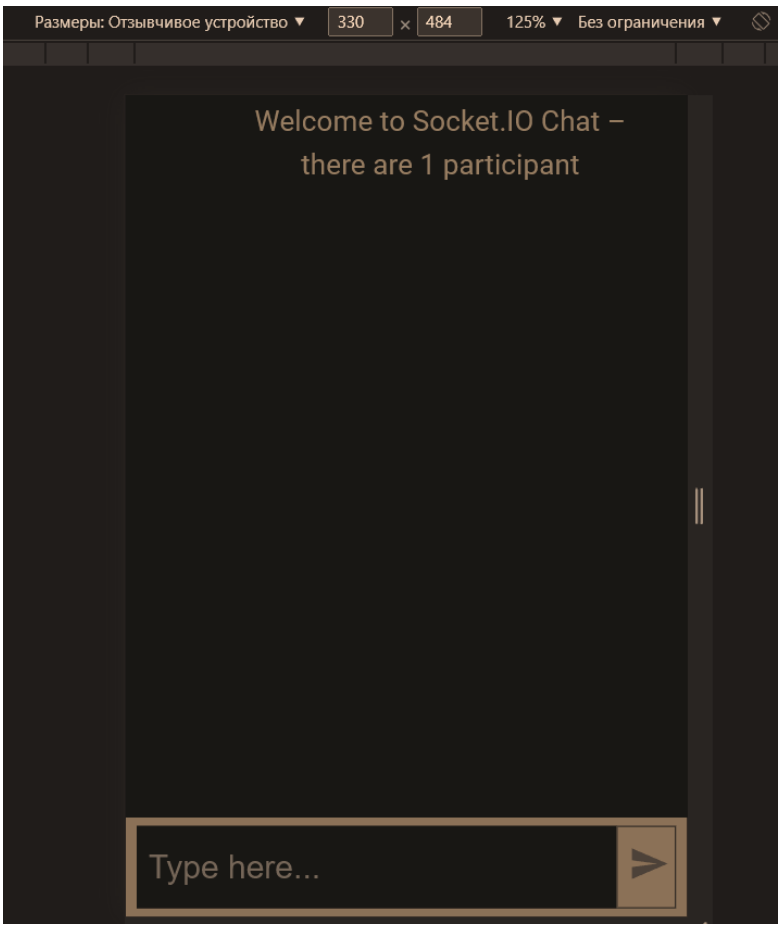

Рисунок 3.6 – Головна сторінка чату на мобільному пристрої

Так що тут я підключаю клієнт до майбутнього сервера, який буде працювати на порту 3001 (рис. 3.7).

```
import io from 'socket.io-client';

export const socket = io.connect('http://localhost:3001/')
```

Рисунок 3.7 – Підключення клієнта

Після доопрацювання та тестування клієнта його зміст виглядає так, як показано на рисунку 3.8.

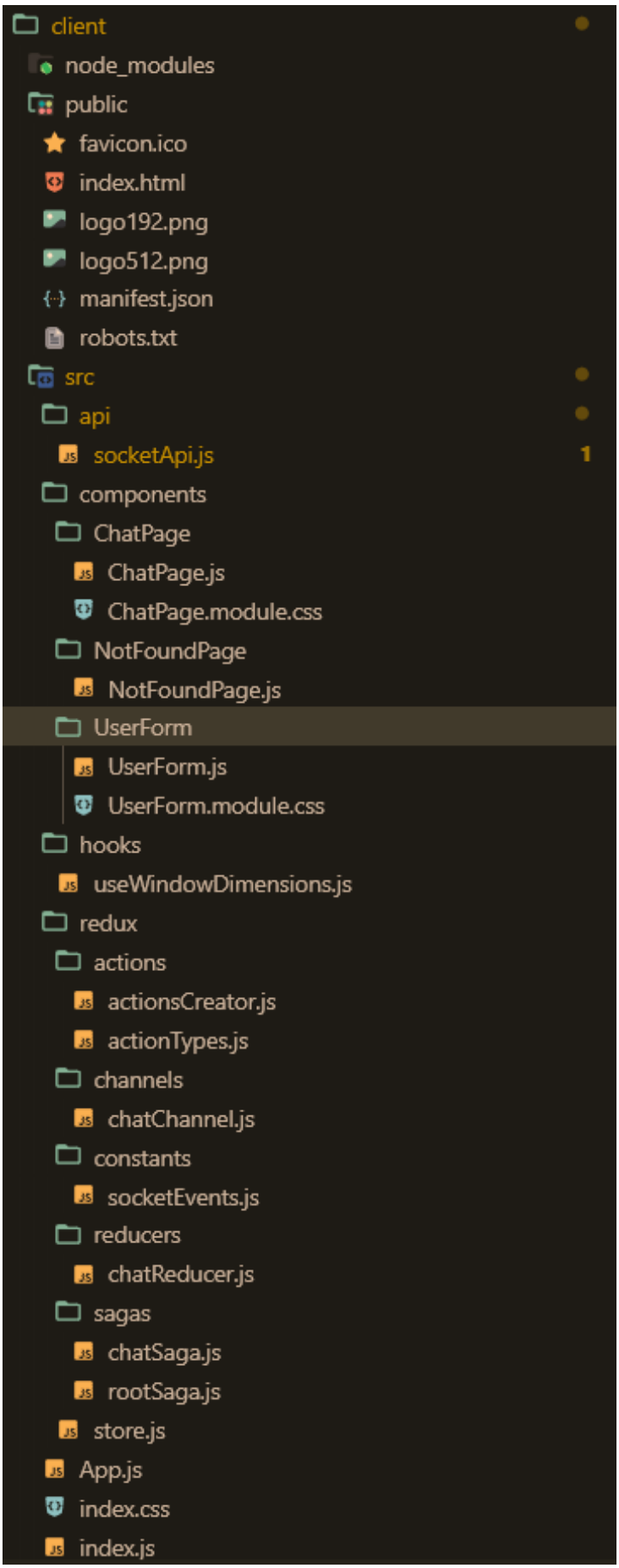

Рисунок 3.8 – Зміст клієнта

3.2 Розробка server для анонімного чату

Для успішного створення та запуску серверу також необхідно виконати кілька підготовчих кроків. Спочатку, необхідно встановити всі необхідні залежності та плагіни, які будуть використовуватись під час роботи серверу. Це дозволить забезпечити оптимальну роботу системи.

Далі, необхідно мати файл package.json зі списком всіх залежностей та їх версій. Цей файл допоможе забезпечити надійність та стабільність роботи серверу, оскільки він містить інформацію про всі використовувані компоненти та їх взаємозв'язки.

Файл package.json для сервера виглядає таким чином (рис. 3.9).

Рисунок 3.9 – Файл package.json для сервера

Тут ми можемо бачити 4 залежності та один скрипт:

– Express – що дозволить значно прискорити та спростити процес розробки;
– Http – щоб отримувати та надсилати дані між клієнтом та сервером;
– Path – може бути корисним при роботі з файлами та директоріями у проєкті на JavaScript. Наприклад, створення шляхів до файлів чи

директоріям, конкатенації шляхів, визначення імені файлу з шляху і т.д. При розробці проєктів на JavaScript, в яких використовуються модулі, Path може знадобитися залежно від інших модулів. Це відбувається, коли модуль використовує функціонал Path для роботи з файлами та директоріями у проєкті. Таким чином, Path може бути корисним модулем для проєктів JavaScript, які вимагають роботи з файловою системою;

– Socket.io – про який я вже згадував раніше;
– Dev – скрипт, який запустить сервер;

Сам сервер уже не такий величезний, як React. Так що я зміг вкластися в один головний файл indexjs. Весь каталог server виглядає так, як на рисунку 3.10.

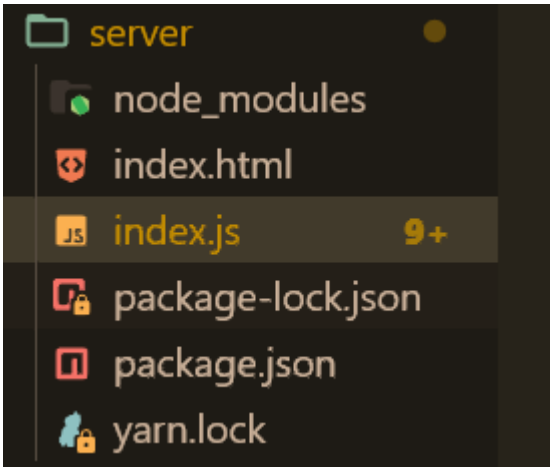

Рисунок 3.10 – Каталог server

Сам код сервера виглядає так, як на рисунку 3.11.

Коли клієнт хоче підключитись до сервера, він створює екземпляр клієнта socket.io у своєму браузері за допомогою JavaScript API io(). Клієнт потім надсилає запит на сервер, щоб встановити з'єднання.

Після встановлення з'єднання клієнт та сервер можуть обмінюватися повідомленнями через веб-сокети. Клієнт може надсилати повідомлення на сервер, використовуючи метод socket.emit(), а сервер може відправляти повідомлення на клієнт, використовуючи метод socket.emit() або socket.broadcast.emit() [1].

```
const express = require('express');
const app = express();
const path = require('path');
const server = require('http').createServer(app);

app.use(express.static(path.join(__dirname, 'build')));

app.get('/', function (req, res) {
  res.sendFile(path.join(__dirname, 'build', 'index.html'));
});

const io = require('socket.io')(server, {
   cors: {
      origin: "http://localhost:3000",
   }
});
const port = process.env.PORT || 3001;

server.listen(port, () => {
   console.log('Server listening at port %d', port);
});

let numUsers = 0;

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

   console.log('connected');
   let addedUser = false;

   socket.on('add user', (username) => {
      if (addedUser) return;

      socket.username = username;
      ++numUsers;
      addedUser = true;
      socket.emit('login', {
         numUsers: numUsers
      });

      socket.broadcast.emit('user joined', {
         username: socket.username,
         numUsers: numUsers
      });
   });

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

      if (!data.trim()) return;
      let messageData = {
         username: socket.username,
         message: data
}

   });

   socket.on('disconnect', (reason) => {
      console.log('disconnect, reason:', reason);
      if (addedUser) {
         --numUsers;

         socket.broadcast.emit('user left', {
            username: socket.username,
            numUsers: numUsers
         });
      }
   });
});
```

Рисунок 3.11 – Код server

На стороні сервера можна визначити обробники подій, які будуть викликатись при отриманні певних повідомлень від клієнта. Наприклад, можна визначити обробник події message, який буде викликатись при отриманні нового повідомлення від клієнта. Обробник може обробляти повідомлення і відправляти його назад на клієнт або широкомовно всіх підключених клієнтів.

Сервер також може керувати підключеннями клієнтів, наприклад, відстежувати підключення та відключення клієнтів за допомогою обробників подій connect та disconnect.

У результаті сервер для чату на socket.io забезпечує надійну та ефективну двосторонню взаємодію між клієнтом та сервером, що дозволяє створювати потужні та інтерактивні програми в режимі реального часу.

3.3 Приклад роботи готового проєкту

Спочатку запускаємо сервер (рис. 3.12) та (рис. 3.13).

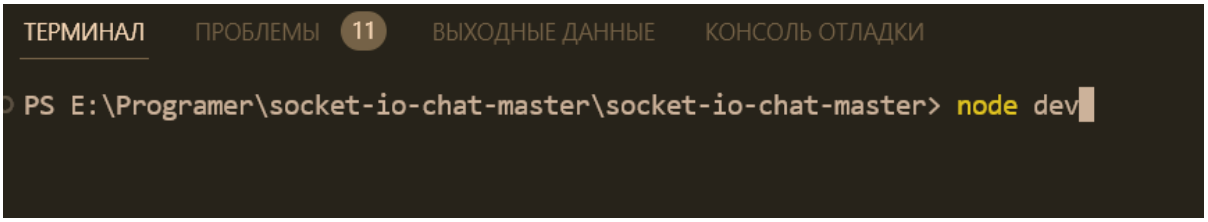

Рисунок 3.12 – Запуск сервера

Рисунок 3.13 – Успішний запуск сервера

Бачимо, що сервер слухає порт 3001. Значить, сервер запустився і працює (рис. 3.12).

Далі запускаємо клієнтську частину. Тобто сам React (рис. 3.14) та (рис. 3.15).

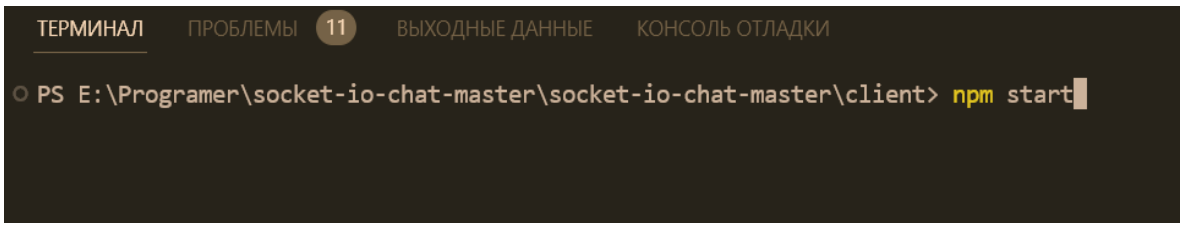

Рисунок 3.14 – Запуск клієнтську частину

Рисунок 3.15 – Успішний запуск клієнт частини

Бачимо, що все успішно запустилося і потрібно перейти на порт 3000 (рис. 3.15).

Переходимо в сам чат і вводимо нікнейм (рис. 3.16).

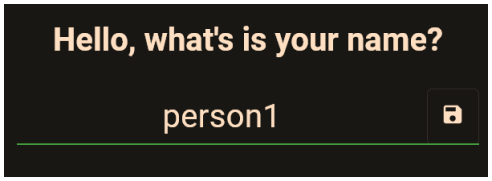

Рисунок 3.16 – Приклад вводу нікнейма

Нас перекидає в чат (рис. 3.17).

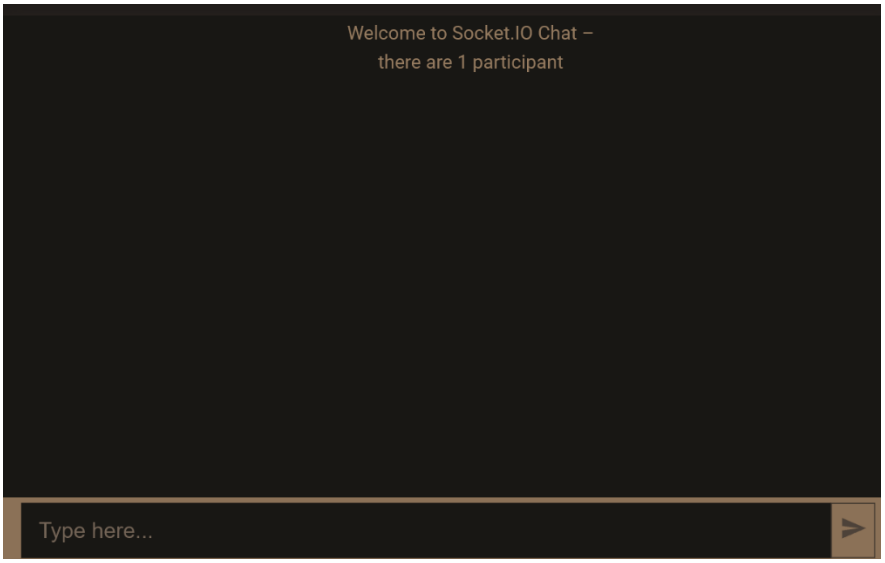

Рисунок 3.17 – Успішно перекинуло до чату

Потім для демонстрації я відкриваю інше вікно браузера і проробляю те саме. І бачимо підсумковий результат (рис. 3.18).

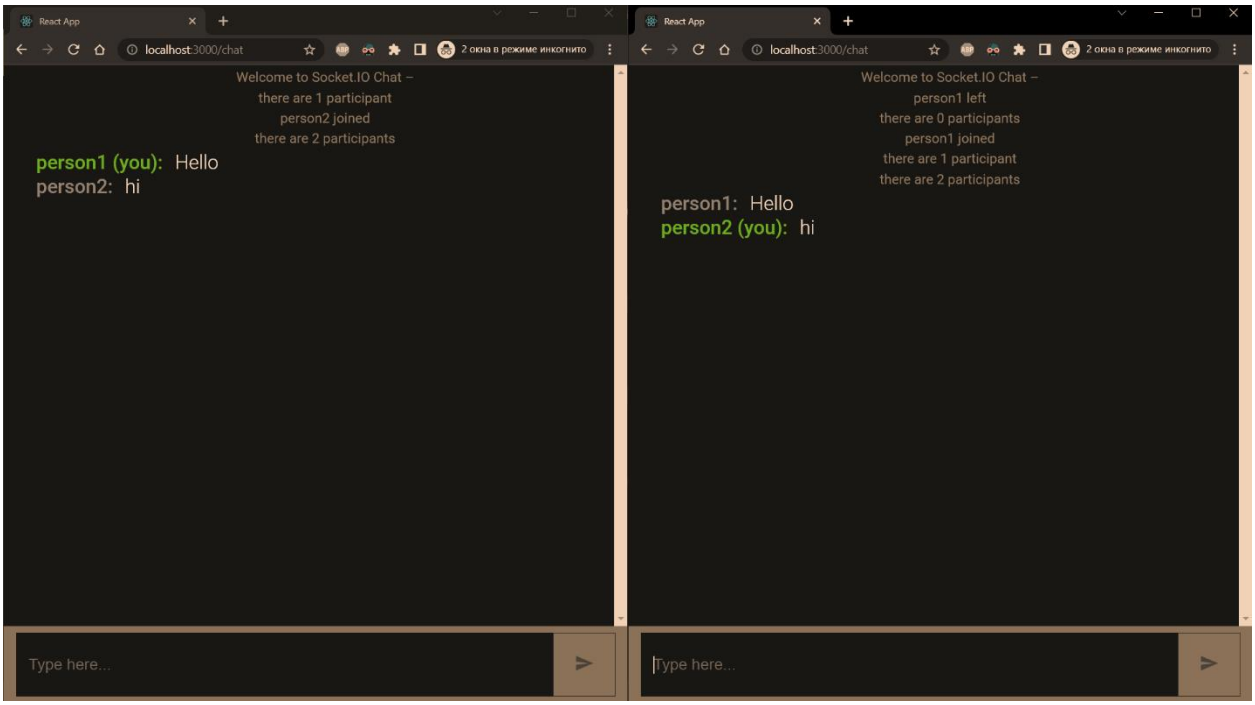

Рисунок 3.18 – Успішно працюючий чат

ВИСНОВКИ

Успішно була розроблена анонімна чат-платформа з використанням технології socket.io. Цей фреймворк дозволяє побудувати ефективну та масштабовану систему обміну повідомленнями в режимі реального часу.

Результати виконаної роботи демонструють успішну реалізацію основних функціональних вимог до анонімного чату, таких як надійність, швидкість передачі повідомлень, а також підтримка анонімного користування.

У роботі вдалося впровадити основні механізми для забезпечення анонімності, включаючи генерацію тимчасових ідентифікаторів користувачів та використання криптографічних алгоритмів для шифрування повідомлень.

Отримані результати підтверджують, що розроблений анонімний чат з використанням socket.io є ефективним і гнучким рішенням для створення платформи комунікації в режимі реального часу з високою ступенем анонімності.

Додатковими перевагами розробленого анонімного чату є його простота в розгортанні та використанні, а також можливість масштабування системи для великої кількості користувачів.

Дипломна робота відкриває шлях до подальших досліджень і вдосконалення анонімного чату з використанням socket.io, включаючи розширення функціональності, вдосконалення алгоритмів шифрування та анонімізації, а також покращення користувацького інтерфейсу.

Результати дипл омної роботи можуть бути корисні для розробників, які прагнуть створити безпечну та анонімну платформу комунікації з використанням socket.io для різних цілей, включаючи ділову, освітню або розважальну галузі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Cadenhead T. Getting Started with Socket.IO. Birmingham : Packt Publishing, 2015. 184 p.
2. Freeman A. Pro React 16 1st ed. New York : Apress, 2019. 768 p.
3. Getting Started Visual Studio Code. Visual Studio Code Documentation. URL: https://code.visualstudio.com/docs (дата звернення: 20.03.2023).
4. Colin J. I. Pro Node.js for Developers 1st ed. New York : Apress, 2013. 328 p.
5. Getting Started with Redux. Redux Documentation. URL: https://redux.js.org/introduction/getting-started (дата звернення: 20.03.2023).
6. JSHint. Wikipedia. URL: https://en.wikipedia.org/wiki/JSHint (дата звернення: 20.03.2023).
7. Debugging in the browser. JavascriptInfo. URL: https://javascript.info/debugging-chrome (дата звернення: 20.03.2023).
8. Mardan A. Express.js Deep API Reference 1st ed. New York : Apress, 2014. 175 p.
9. API. Wikipedia. URL: https://en.wikipedia.org/wiki/API (дата звернення: 20.03.2023).