

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

**Звіт**  
**з технологічної практики**

студента II курсу групи ПЗ-20-1  
спеціальності 121 «Інженерія програмного  
забезпечення»

Вербовського Олександра Юрійовича  
(прізвище, ім'я та по-батькові)

Керівник ст. викл.каф. ПЗ Чижмотря О.Г.

Дата захисту: " \_\_ " \_\_\_\_\_ 2022 р.

Національна шкала \_\_\_\_\_

Кількість балів: \_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

Члени комісії

_____	_____
(підпис)	(прізвище та ініціали)
_____	_____
(підпис)	(прізвище та ініціали)
_____	_____
(підпис)	(прізвище та ініціали)

Житомир – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп'ютерних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень: бакалавр

Спеціальність 121 «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»

В.о.зав. кафедри ІПЗ

\_\_\_\_\_ А.В.Морозов

“ \_\_\_\_ ” \_\_\_\_\_ 2022 р.

ЗАВДАННЯ  
НА ПРАКТИЧНУ РОБОТУ СТУДЕНТУ  
Вербовському Олександру Юрійовичу

1. Тема роботи: розробка чату з оновленням даних, режимі реального часу, на основі вивчення нового матеріалу,  
керівник роботи: ст. в. Морозов А.В.
2. Строк подання студентом: “ 04 ” лютого 2022р.
3. Вихідні дані до роботи: розробка чату з оновленням даних, в режимі реального часу, на основі вивчення нового матеріалу
4. Зміст розрахунково-пояснювальної записки(перелік питань. Які підлягають розробці)
  - Аналіз проблематики, методів та засобів вирішення задач
  - Проектування та розробка програмного забезпечення
  - Опис роботи з програмним додатком та його тестування
5. Перелік графічного матеріалу(з точним зазначенням обов'язкових креслень)
  1. Посилання на репозиторій: [https://gitlab.com/ipz201\\_voyu/chat](https://gitlab.com/ipz201_voyu/chat)
6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посади консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1,2,3	Чижмотря О.Г., ст. викладач каф. ІПЗ		

7. Дата видачі завдання: “ 24 ” січня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів курсового проекту	Строк виконання етапів проекту	Примітки
1	Постановка задачі	24.01	Виконано
2	Пошук, огляд та аналіз аналогічних розробок	25.01	Виконано
3	Формулювання технічного завдання	25.01	Виконано
4	Опрацювання літературних джерел	28.01	Виконано
5	Проектування структури	02.02	Виконано
6	Написання програмного коду	03.02	Виконано
7	Відлагодження	03.02	Виконано
8	Написання пояснювальної записки	03.02	Виконано
9	Захист	04.02	

Студент \_\_\_\_\_  
(підпис)

Вербовський О.Ю.  
(прізвище та ініціали)

Керівник проекту \_\_\_\_\_  
(підпис)

Морозов А.В.  
(прізвище та ініціали)

## РЕФЕРАТ

Завданням на практичну роботу було розробити інтернет-магазин електронної техніки.

Пояснювальна записка до практичної роботи на тему «Розробка чату» складається з вступу, трьох розділів, висновків, списку використаної літератури та додатку.

Текстова частина викладена на 43 сторінках друкованого тексту.

Пояснювальна записка має 5 сторінку додатків. Список використаних джерел містить 9 найменувань і займає 1 сторінку. В роботі наведено 34 рисунків. Загальний обсяг роботи – 47 сторінок.

Ключові слова: NODE.JS, REACT, REDUX, WEB-САЙТ, MONGODB, СЕРВЕР, ЧАТ, СОКЕТИ, ІНТЕРНЕТ, ДАНІ, КЛІЄНТ.

					ДУ «Житомирська політехніка». 21.121.05.000 - ІПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Вербовський О.Ю.			Розробка чату з оновленням даних в режимі реального часу	Літ.	Арк.	Аркушів
Перевір.		Морозов А.В.					4	47
Керівник						ФІКТ Гр. ІПЗ-20-1[1]		
Н. контр.								
Зав. каф.								

## Перелік умовних скорочень

Рис – рисунок

ООП – Об'єктно орієнтоване програмування

БД – база даних

ТП – технологічна практика

CRUD – create, read, update, delete

РРЧ – режим реального часу

VSCODE – Visual Studio Code

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

## Зміст

ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	8
1.1 Аналіз задачі, засобів та методів її рішення.....	8
1.2 Аналіз існуючого програмного забезпечення за тематикою курсового проекта.....	8
1.3 Технічне завдання на курсову роботу.....	12
Висновки до першого розділу.....	13
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ .....	14
2.1 Проектування загального алгоритму роботи програми.....	14
2.2 Розробка функціональних алгоритмів роботи програми.....	17
2.3 Розробка програмного забезпечення.....	19
Висновки до другого розділу.....	26
РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ.....	27
3.1 Опис роботи з програмним додатком.....	27
3.2. Тестування роботи програмного забезпечення.....	32
Висновок до третього розділу.....	40
ВИСНОВКИ .....	41
СПИСОК ЛІТЕРАТУРИ.....	43
ДОДАТКИ .....	44

## ВСТУП

У цьому звіті буде наведено процес розробки програми "Онлайн чат з оновленням даних в режимі реального часу".

**Актуальність теми.** Чат (англ. chat — «розмова», «невимушена бесіда») — мережевий засіб для швидкого обміну текстовими повідомленнями між користувачами інтернету в системі реального часу.

Отже чат дозволяє людям спілкуватись з різних точок світу у РРЧ, що є корисним, адже справді зручно, коли можеш поговорити з людиною, відправити їй фотографію або відеозапис тим самим розділивши якийсь приємний момент з іншою людиною, яка може знаходитись на іншому кінці світу.

**Метою роботи** є розробка чату за допомогою Node.js для серверної частини, React і Redux для клієнтської та сокетів для взаємодії між клієнтом та сервером у РРЧ.

### **Завданням на практичну роботу:**

- аналіз теоретичних задач проектування та реалізації чату;
- аналіз існуючих чатів, визначення унікальності;
- розробка адаптивного інтерфейсу веб-сайту за допомогою HTML5, CSS3, JavaScript(React та Redux);
- написання серверної частини;
- реалізація взаємодії між клієнтом та сервером як за допомогою сокетів так і за допомогою звичайного експресу.

**Об'єктом дослідження** є методи та засоби розробки чату за визначеними предметними областями.

**Предметом дослідження** є використання веб-технологій для забезпечення інформаційних потреб предметної області. В процесі виконання практичної практики студент може використовувати монографічні, аналітичні, математичні, графічні методи, методи об'єктно-орієнтованого проектування та програмування та інші методи дослідження.

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

# РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

## 1.1 Аналіз задачі, засобів та методів її рішення

Задача полягає у тому, що потрібно:

- Обрати потрібну інформацію
- Визначити потреби можливого замовника
- Створити зрозумілий для звичайного користувача інтерфейс програми

В процесі аналізування задачі було вирішено розробити додаток у середовищі VSCODE за допомогою мови node.js(для серверної частини сайту), js(бібліотеки react та redux)(для клієнтської), оскільки вони дозволяють швидко пере обробляти, відображати та передавати дані, що є необхідним для чату.

## 1.2 Аналіз існуючого програмного забезпечення за тематикою курсового проекту

Сьогодні існує не мало різних чатів спрямованих на різну аудиторію та для різних цілей(внутрішнє користування, всередині якоїсь компанії для комунікації між працівниками виключно продуктами, які може контролювати компанія, зовнішнього, наприклад телеграм – будь-хто будь-де може використовувати продукт) і визначити серед них найпопулярніші та найкращі досить складно, тому я обрав три найкращі чати, на мою думку. Ними є: “Telegram”, “Discord”, “Facebook messenger”.

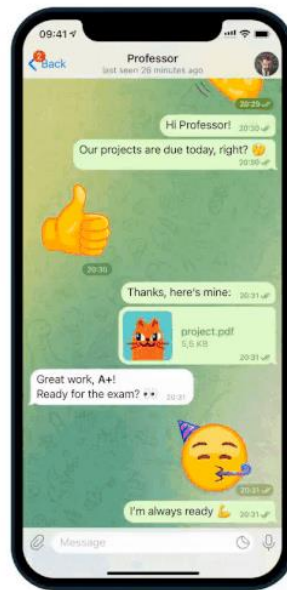
		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		8



Назва: “Telegram”

Посилання на сайт: <https://web.telegram.org/k/>

Вигляд чату зображено на Рис.1.1:



**Chat Themes**

Рис.1.1 – Чат “Telegram”,

Переваги:

- 1) Можна створити нові чати;
- 2) Знайти користувача;
- 3) Відправити не лише текстові повідомлення, а й аудіо, відео та інші файли;
- 4) Можна бачити статус співрозмовника(онлайн/офлайн/щось робить);
- 5) Можна задекорувати текст(в якийсь колір, або щ=щадати стиль тексту);
- 6) Зручний, інтуїтивно зрозумілий інтерфейс;
- 7) Можливість вибору поточної теми дизайну;

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

Назва: “Discord”,

Посилання на програму: <https://discord.com/>

Вигляд головної сторінки сайту можна побачити на Рис.1.2:



Рис.1.2 – Вигляд основної частини “Discord”

Переваги:

- 1) Можна виконувати пошук товару за категоріями, цінами та іншими певними характеристиками певних категорій;
- 2) Можливість пошуку користувача за його унікальним id;
- 3) Відправити не лише текстові повідомлення, а й аудіо, відео та інші файли;
- 4) Можливість проведення онлайн трансляцій;

Недоліки:

Зручний, однак не завжди зрозумілий інтерфейс для нових користувачів;

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Назва: “Facebook messenger”

Посилання на програму: <https://www.messenger.com/>

Вигляд головної сторінки сайту зображено на Рис.1.3:

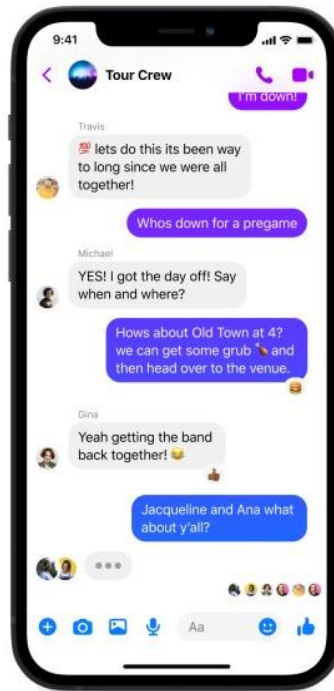


Рис.1.3 – Головна сторінка сайту “COMFY”

Переваги:

- 1) Можливість відправки аудіо повідомлень;
- 2) Можливість оперування грошима;
- 3) Можливість відправки смайликів знаходиться на головній відправній панелі;

Недоліки:

При тестуванні додатку було знайдено некоректне відображення сторінки при адаптивності;

В результаті аналізу цих програм я дійшов висновку, що чат має мати привабливий, адаптивний та інтуїтивно-зрозумілий дизайн. Крім цього чат повинен мати можливість відправки відео та аудіо повідомлень, їх запис, відправку фотографій та шукати нових користувачів для листування.

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

### 1.3 Технічне завдання на курсову роботу

є розробка, здійснення тестування і розгортання веб-орієнтованої програми для листування людей у РРЧ.

#### Загальні функції системи:

- Пошук та створення нових чатів
- Відправка нових повідомлень(текстових/відео/аудіо)
- Перемикання між чатами
- Перегляд повідомлень
- Реєстрація та логінінг

#### Функції серверної частини:

- Робота з користувачами: створення акаунту та вхід в нього
- Створення чатів
- Пошук користувачів для створення нових чатів
- Повертання а оновлення чатів
- Створення та повернення повідомлень

#### Функції клієнтської частини:

- Стандартні функції користувачів: реєстрація, вхід
- Редагування повідомлень, які потрібно відправити(зміна кольору тексту, виділення і т.д.)
- Відправка відео та аудіо повідомлень
- Пошук користувачів
- Створення нових чатів

#### Загальні функціональні вимоги щодо системи:

- Розробка фронтенду за допомоги React;
- Розробка бекенду за допомоги Node.js;
- Взаємозв'язок сервер-клієнт за допомогою сокетів

#### Функціональні вимоги клієнтської частини:

- Компоненти чату:
  - Список існуючих чатів
  - Пошук користувачів для взаємодії між собою
  - Блок зі створенням повідомлень
  - Запис аудіо та відео
  - Зміна теми

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

- Вимоги до роботи з профілем користувача:
  - Сторінка з формою для реєстрації користувача, сторінка з формою логінінгу та можливість виходу з акаунту
- Вимоги до роботи з чатами:
  - Відображення чатів
  - Можливість пошуку користувачів для нових чатів
  - Створення нових чатів
  - Оновлення чатів в режимі реального часу
- Вимоги до роботи з повідомленнями:
  - Створення нових текстових повідомлень
  - Створення нових аудіо/відео
  - Декорація тексту
  - Відображення нових повідомлень в режимі реального часу

#### **Функціональні вимоги адміністративної частини:**

- Оскільки відбуватиметься розробка чату, де буде звичайна розмова між двома людьми, то редагування та вплив адміністратора на чати не буде доцільним.

#### **Висновки**

У ході виконання першого розділу було проаналізовано три чати у результаті чого було визначено головний функціонал програми та допоміжний, крім цього, було визначено інтерфейс користувача, який необхідно створити. Також, під час виконання розділу 1, було вирішено виконувати курсовий проект у VSCODE за допомогою Node.js, React та Redux.

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

## РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 2.1 Проектування загального алгоритму роботи програми

На даному етапі необхідно створити загальний алгоритм роботи програмного додатку.

Після запуску програми відкриється головне вікно. Після чого сайт буде чекати на дії користувача.

Можуть бути виконані такі дії:

#### 1. Авторизація та реєстрація:

- Після входу на сторінку користувач буде бачити форму з пропозицією входу, після заповнення та відправки якої сервер поверне деякі дані клієнту, або інформацію про помилку, залежно від чого буде перекинуто користувача на головну сторінку, або повідомлено про помилку;
- Також на цій формі буде можливість реєстрації, при натисканні на кнопку “Sign up” буде змінено тип форми на відповідну. Після заповнення цієї форми та відправки її на сторону сервера, користувача буде повідомлено про успішну реєстрацію на електронній пошті, однак, поки користувач не перейде за посиланням, що прийде, його акант буде “заморожений”, якщо при валідації реєстраційних даних виникне помилка, користувача повідомлять про неї;
- Також з головної сторінки можна буде вийти з аканту, в бюргер меню буде спеціальна кнопка для цієї дії.
- Також у користувача буде можливість міняти темну тему на світлу і навпаки, для цього буде спеціальний повзунок на вікні з логінінгом та спеціальна кнопка в бюргері на основній сторінці чату.

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

## 2. Пошук та вибір чатів:

- Для пошуку чатів буде спеціальне поле, при зміні якого з сервера буде приходити інформація про зареєстрованих користувачів, імена, електронні пошти або номери яких будуть містити считані символи.
- При кліку на новий чат, усі інші знайдені чати будуть зникати, а при відправці повідомлення вибраний новий чат переміститься в блок з усіма чатами користувачів.
- Також при вході/виході в акант, усім користувачам буде відправлено повідомлення з цією інформацією, що дозволить відображати та змінювати новий статус співрозмовника у всіх чатах, де він є.

## 3. Перегляд та відправка повідомлень в чатах:

- При натисканні на чат на сервер буде відправлятися запит на отримання повідомлень цього чату, після отримання яких буде їх промальовка.
- У блоці, де буде відображатись список повідомлень, у користувача буде відображена панель з полем для вводу повідомлення, кнопкою для зчитування емоцій, відправкою файлів та записом аудіо або відео меседжа(для зміни режиму потрібно буде обробити подію при виклику контекстного меню).
- Також, повідомлення, що будуть відправлятися можна буде редагувати, для цього також потрібно буде клікнути правою клавішею миші, по виділеному тексту, в результаті чого над текстом з'явиться блок з кількома кнопками, які будуть замінювати виділений текст новим відредагованим.
- При отриманні нового повідомлення, чат, повідомлення до якого прийшло, буде оновлюватись у списку чатів(останнє

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

повідомлення), якщо цей чат є активним в момент приходу повідомлення, то воно буде відображено ще й у блоці з повідомленнями.

Отже функціональна схема інтернет-додатку має наступний вигляд:

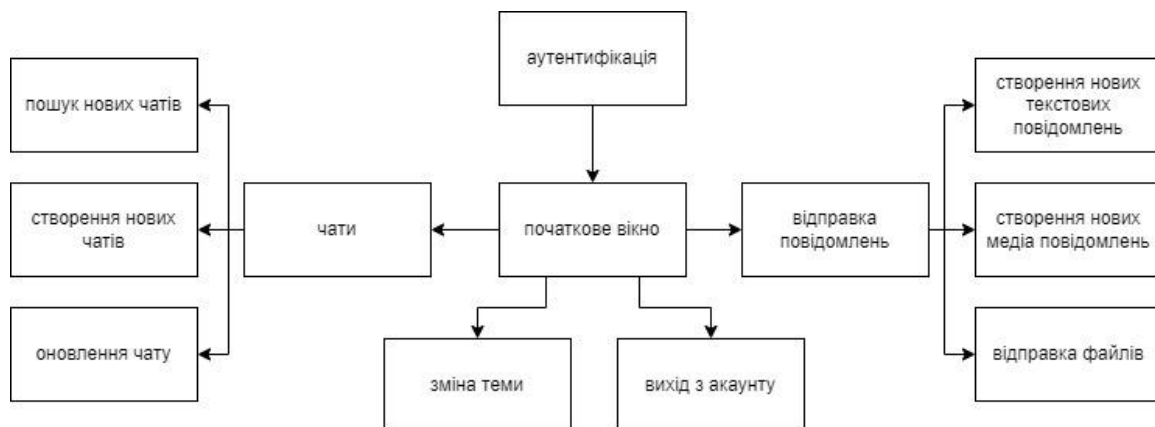


Рис.2.1 – Функціональна схема додатку

На наступному зображенні показано схему сутність-зв'язок:

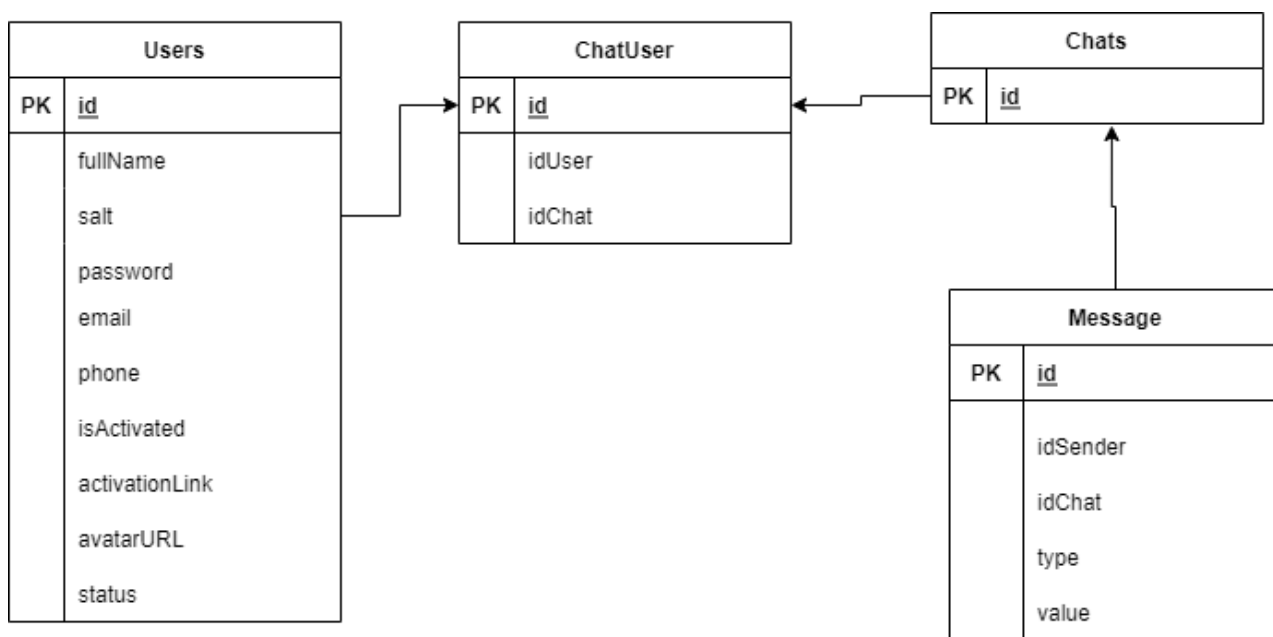


Рис.2.2 – Діаграма „сутність-зв'язок” чату



## 2.2 Розробка функціональних алгоритмів роботи програми

Безсумнівно одним з головних функціоналів будь-якого веб-додатку є можливість авторизації, адже цей функціонал дозволяє зберігати певну інформацію з одного пристрою та відображати її повторно не лише на ньому, а й на інших, тому з цього функціоналу варто почати написання програми.

Отже, для данного модулю потрібно зробити сервер, для цього ідеально підійде бібліотека `express`, так як у ній реалізовано легке прийняття запитів зі сторонни клієнта та зручно запускати та налаштовувати сервер. Після реалізації створення та налаштування сервера потрібно створити роутер, що буде реагувати на `Post` та `Get` запити зі сторонни клієнта. У цьому роутері при певних запитах будуть викликатись відповідні методи, наприклад на логін запит буде викликатись метод логін з контроллера, що відповідає за роботу з користувацьким модулем, а на реєстрацію – `registration`, це є зручним, так як до цього запиту можна навісити ще один обробник, який перевірить чи всі дані коректні та продовжить запит з результатом перевірки. Дана перевірка потрібна при використанні токенів, на жаль, у роботі не вдалось реалізувати до кінця логіку `Access` та `Refresh` токенів, тому було вирішено повністю прибрати її з проекту.

У методах контролера будуть викликатись методи з сервісів, які дозволять зберігати дані в базу даних за схемами, збереженими в моделях.

Після вдалої авторизації на сторону клієнта буде повернено інформацію про нього, після чого користувача буде перекинуто на сторінку чату, звідки буде підключено сокет, що дозволить серверу почати прослуховування вхідних сокетів, метод, що буде прослуховувати сокети теж винесено до роутерів.

Після того, як сокети будуть підключені, користувачу будуть навішені сокети, що будуть спрацьовувати при приходженні повідомлення про те, що співрозмовник почав писати повідомлення, користувач вийшов або увійшов у мережу, прийшло повідомлення та інші події.

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

Також будуть створені та експортовані методи для того, щоб надсилати сокети на сторону сервера, а саме метод коли користувач почне писати повідомлення, відправить його, відправить медіа-повідомлення, або файл, вибере інший чат.

Крім взаємодії з сервером у користувача буде декілька корисних функцій. Наприклад, буде можливість перемкнути поточну тему, що дозволить комфортніше переглядати сайт у темну пору доби, при натисканні на певну кнопку, буде змінюватись поточне значення теми, Залежно від якого буде навішено інші стилі на певні елементи.

Також буде можливість для декорування повідомлень, для цього потрібно буде виділити текст, клікнути по ньому лівою кнопкою миші, після цього буде змінено стан панелі з відповідними кнопками, на активний, після чого в одному з хуків буде викликано метод для задання позиції цієї панелі, щоб вона з'являлась над виділеним текстом, а не в якомусь нелогічному місці. Після того, як панель буде відображено, по її кнопках можна буде натиснути, після чого, виділений текст буде замінено текстом, що буде обернений у блок, який вибрав користувач(або ні, якщо вибір буде повторний).

Також буде реалізовано запис аудіо та відео, при натисканні на праву кнопку миші по кнопці з записом буде відбуватись зміна режиму запису(аудіо/медіа), при натисканні на ліву зміниться стан кнопки на активну, що викличе метод для запису повідомлення. У функції буде відбуватись запит до початку стріма, якщо користувач надає доступ, до починається запис. При натисканні на кнопку "Stop" відбудеться переведення записаного інформації в певний тип даних, який буде повернено до компоненти, де зберігатимуться повідомлення до відправки, після чого цей тип даних буде переведено або в аудіо або в відео та відображено цей блок користувачу для підтвердження відправки.

Дане відображення буде працювати і для відображення звичайних файлів. Детальніше розглянемо цю логіку. Функція приймає считаний файл,

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

читає його, і считані дані повертає, наступна функція отримані дані передає стан, як поточне медіа-повідомлення, після чого відбувається відображення контексту цього повідомлення. При натисканні поза блоком з підтвердженням повідомлення стан поточного медіа-меседжу очищується, що призводить до перерендерингу цього блоку, а відповідно до його видалення. При натисканні ж на кнопку підтвердити відбудеться виклик іншого методу, що відправить считані дані на сервер функцією, отриманої з користувацького хука.

## 2.3 Розробка програмного забезпечення

Структуру програми зображено на Рис.2.3:

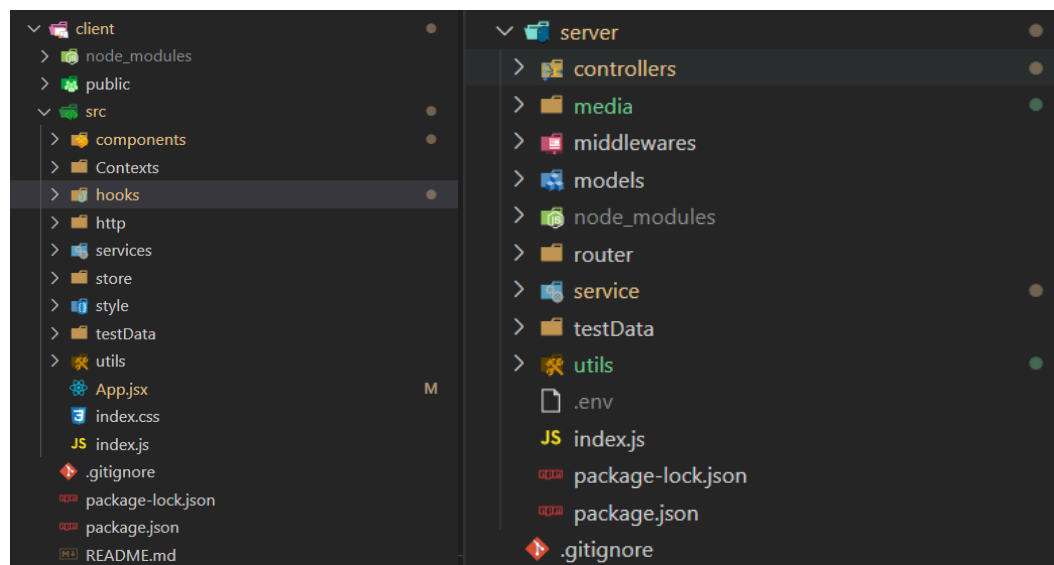


Рис.2.3 – Структури рішення серверної та клієнтської частини програми

Розглянемо для початку папки та файли в яких знаходиться серверна частина:

Index.js – це основний файл сервера в якому відбувається підключення та налаштування серверу. Для того, щоб створити сервер потрібно використати імпортований метод express, після чого, результат зберегти в константу, навісити роутери на певні маршрути, зробити налаштування сервера та навісити прослуховувач(певного порту). Лістинг коду:

```
require("dotenv").config()
const express = require('express');
const app = express();
const PORT = process.env.PORT || 5000;
const logText = `http://localhost:${PORT}`;
const urlencodedParser = express.urlencoded({
```

```

        extended: true
    });

    app.use(express.json());
    app.use(urlencodedParser);
    app.use(upload.array());
    app.use(cors({
        credentials: true,
        origin: process.env.CLIENT_URL
    }));
    app.use("/api", router)
    app.use(errorMiddleware);

    const start = async () => {
        try {
            const server = app.listen(PORT, () => {
                console.log(`Сервер начал прослушивание запросов: ${chalk.blue(logText)}`);
            })
        }

    }

    start();

```

Також у данному файлі відбувається підключення сокетів, для цього була використана бібліотека socketIO та підключення бази даних mongodb:

```

await mongoose.connect(process.env.DB_URL, {
    useNewUrlParser: true,
    useUnifiedTopology: true
})
const socketIO = require('socket.io');
const io = socketIO(server, {
    cors: {
        credentials: true
    }
})
io.on('connect',(socket)=> onConnection(socket, io));

```

Крім цього, варто зазначити, що рядок `require("dotenv").config()` дозволяє считувати дані з файлу `.env` що дає доступ до даних з цього файлу у будь-якій точці проекту серверної частини. Це зручно, адже там можна зберігати якусь приватну інформацію, яку, можливо, доведеться змінювати в майбутньому, наприклад параметри для підключення до бази даних, або посилання на клієнтську частину.

Далі розглянемо папку `router`, так як в ній знаходяться обробники запитів, саме в них викликаються метод из контролерів, які виконують якісь дії(виконують вхід, реєстрацію користувача і т.д.). Для прикладу розглянемо методи декілька рядків з роутеру, що реагує на звичайні запити та з роутеру, що реагує на сокети. Перший роутер:

```

const {Router} = require('express');
const router = new Router();
router.post("/login", userController.login);

```

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

```
router.get("/userAvatar/:link", chatController.getAvatar);
router.get("/messageMedia/:type/:link", chatController.getMediaMessage);
module.exports = router;
```

Метод `UserController.login` зреагує при логінінгу користувача, а методи `chatController.getAvatar` та `chatController.getMediaMessage` будуть спрацьовувати коли користувачем будуть запитані малюнки, відео та аудіо зі сторонни клієнта(здебільшого потрібно для відображення даних у `img`, `video` та `audio` блоках).

Сокет-роутер:

```
const chatSocketController = require('../controllers/chatSockets');
const onConnection = async (socket, inputIo) => {
  const chatSocketObj = new chatSocketController(socket, inputIo);
  socket.on("createChat", async (mess) => await chatSocketObj.createChat(mess));
  socket.on("createMessage", async (mess) => await chatSocketObj.createMessage(mess));
  socket.on("setChat", async (mess) => await chatSocketObj.setChat(mess));
  socket.on('disconnect', (mess) => chatSocketObj.disconnect(mess))
}
module.exports = onConnection;
```

Методи `createChat`, `createMessage`, `setChat`, `disconnect` спрацюють при виклику відповідних сокетів зі сторони клієнта.

Далі розглянемо папку `controllers`, адже в файлах цієї папки знаходяться методи які виконують основну логіку роботи. Наприклад у файлі `chat.js` є метод `getMediaMessage`, який приймає `request`, `response` та `next`, перший параметр – це параметр, який містить інформацію, що прийшла зі сторони клієнту. У полі `params` збережено асоціативний масив вхідних параметрів, считаних з `get`-запиту. Для того, щоб зчитати тип даних, який потрібно повернути користувачу використано рядок `req.params["type"]`, після чого відбувається його аналіз та звернення до папки, що відповідає типу та знаходить посилання файлу, який потрібно повернути, за ще одним считаним параметром `link`, після чого рядком `res.sendFile(fullPath)` відбувається повертання даних, считаних з файлу, що дозволяє в місці, де відбувся запит відобразити медіа-дані.

Розглянемо ще один метод але вже з сокет-контролера – `createChat`. Лістинг повного коду наведено в Додатку А. Метод є асинхронним, приймає параметр `mess`, він містить відправлені зі сторонни клієнта дані. У самому методі відбувається виклик методу `createChat` з сервісу `chat`. Даний метод

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

приймає id користувачів, які потрібно додати до створеного чату. Після цього відбувається повертання id нового чату, та збереження його у відповідну змінну, щоб легше було виконувати певні операції з іншими сокетами. Після ще одного збереження користувацьких id користувачу, що створив чат відправляється сокет з інформацією про створений чат, після чого відбувається відправка зі сторони клієнта повідомлення про створення нового повідомлення, але це вже інший метод.

Також варто розглянути папки models та service, так як в них містяться моделі даних, що зберігаються в базі даних та функції для їх обробки. Наведу приклад на основі моделі message:

```
const {Schema, model} = require('mongoose');
const MessageSchema = new Schema({
  idSender:{type: String, required: true},
  idChat:{type: String, required: true},
  type:{type: String, default: "text"},
  value:{type: String},
  sendAt:{ type: Date, required: true}
})
module.exports = model("Message", MessageSchema);
```

У цій моделі є п'ять полів: id відправника, id чату, дата відправки, тип повідомлення та контент. З них обов'язковими є перші три поля також усі вони мають тип String, за винятком sendAt, дане поле має тип Date.

Далі розглянемо один з методів у сервісі для створення повідомлення. Даний метод є асинхронним, так як усі методи для роботи з базою даних є асинхронними. Також метод приймає чотири параметри, а саме – id відправника, id чату, контент та тип. У середині методу відбувається створення об'єкта цієї моделі за допомогою методу const newMess = await MessageModel.create({idSender, idChat, type, value, sendAt}). Після чого відбувається повернення створеного повідомлення. Детальніше моделі зображені у Додатку Б.

Перейдемо до клієнтської частини. У ній цікава лише папка src.

Для початку розглянемо папку store, адже в ній буде зберігатись інформація про поточний стан програми. Далі розглянемо папку actions, в ній

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

знаходяться всі константи з можливими типами дій, які може викликати користувач(зроблено для того, щоб зменшити кількість помилок).

Далі в папці actionCreators знаходяться функції, які приймають якесь значення, та на основі цього створюють об'єкт з полем тип, що знаходиться в папці actions та полем payload з отриманим значенням.

Також потрібно розглянути файли-редьюсери. Вони експортують функції, які приймають параметрами state та action, у перший параметр по замовчуванню передаємо якийсь початковий state, усередині функції відбувається перевірка того, який тип action містить, залежно від цього відбувається зміна вхідного state(по замовчуванню поєртається початкове вхідне значення).

У файлі index відбувається створення store на основі існуючих редьюсерів. Лістинг коду:

```
export const store = createStore(combineReducers({: authReducer, main: mainReducer, chatList: chatListReducer, chat: chatReducer}));
```

Отже тепер є сховище даних та методи, які можуть змінювати ці сховища, тому варто розглянути місця де відбуваються їх виклики та ініціалізація. Остання дія відбувається у файлі index.js. Лістинг:

```
import {store} from './store/index'
import {Provider} from 'react-redux'

ReactDOM.render(
  <React.StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>,
  document.getElementById('root')
);
```

Розглянемо тепер виклик зміни та отримання інформації сховища. Для цього найкраще підійде повзунок, що змінює тему на сторінці з формою логінінгу. Даний файл імпортує метод connect, та об'єкт mainAction з папки actionCreators(який містить усі action creators). Далі створюється сама функція ChangerTheme. Після неї потрібно створити дві функції. Перша приймає state та повертає об'єкт, що складається з даних, які потрібно повернути зі сховища, а другий – приймає функцію dispatch та повертає

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

об'єкт з функцій, які приймають параметри та dispatch actionCreators на основі вхідних даних. Приклад:

```
const mapStateToProps = state => {
  return {
    theme: state.main.theme
  }
}

const matchDispatchToProps = dispatch => {
  return {
    ChangeTheme: newTheme=>dispatch(mainAction.mainActionCreator.changeThemeAction(newTheme))
  }
}
```

Однак при експорті функції, що створює цей компонент його потрібно передати параметром у функцію, що повертається connect-ом, яка приймає дві попередньо-створені функції: export default connect(mapStateToProps, matchDispatchToProps)(ChangerTheme);

Тепер варто розглянути як саме передаються ці параметри у компонент. Сам компонент приймає лише props, тому для того, щоб отримати певні дані чи методи теж варто звернутись до props. На основі функції onChangeCheckboxChange з методу, що створює компонент, можна детальніше розібрати як це виглядає:

```
const onChangeCheckboxChange = (e)=>{
  const newTheme = e.target.checked?"dark":"light";
  localStorage.setItem("actualTheme", newTheme);
  props.ChangeTheme(newTheme);
}
```

Далі варто розглянути папку hooks, у ній є папка useChat – це кастомний хук, який приймає певні функції які потрібно викликати при певних змінах, та повертає набір функцій для того, щоб відправляти якусь інформацію на сервер. Також при виклику даного хука відпрацьовується хук useEffect, який приймає функцію, усередині якої відбувається під'єднання сокету до сервера та навішення прослуховувачів подій та повертає функцію, яка відключає юзера від сервера(дана функція спрацьовує при видаленні компонента зі сторінки). Розглянемо ініціалізацію сокету навішення однієї події та повертання функції, Ініціалізація та навішення знаходиться в хуку useEffect. Лістинг коду:

```
const socketRef = useRef(null);
useEffect(() => {
```

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		24



```

socketRef.current = io(API_URL, {
  query: { userId, hashedPassword },
  transports: ["websocket"]
});
socketRef.current.on("startTyping", mess => {
  if (mess.idChat === actualChat.actualChatId && mess.idUser !== userId) {
    changeTyping(true);
  }
});
}

```

У строчці з ініціалізацією socketRef, що теж є результатом застосування хука useRef, відбувається звертання до сервера з передачею деяких даних користувача, а саме – id та хешований пароль, у результаті чого на сервері спрацює певний метод.

І за аналогією до метода, що спрацьовує коли сервер повідомляє юзера про те, що хтось пише повідомлення в чаті, який містить поточного користувача, буде відпрацьований метод, який збереже у сховище усі отримані чати.

Далі розглянемо метод, який відправляє повідомлення на сервер, а саме - createMessage. Лістинг:

```

const createMessage = async (idChat, type, message) => {
  setLoading(true);
  await socketRef.current.emit("createMessage", {idSender: userId, idChat, type, message});
};

```

Даний метод приймає тип повідомлення, id чату в який відправляється повідомлення та контекст повідомлення. Функція є асинхронною, в ній відбувається відправка сокету на сервер з повідомленням createMessage та вхідними даними та id поточного користувача, так як лише він може відправити повідомлення зі свого акаунту. Після цього сервер виконає якісь дії та вернее на сторону клієнта відповідний сокет і спрацює подія, що вже є навішана у хуці useEffect.

Оскільки даний хук вертає набір функцій, який потрібно буде викликати в якихось дочірніх компонентах, то варто використати хук useContext для раціональнішої передачі даних дочірнім компонентам.

Лістинг створення контексту:

```

import React from 'react';

export const CreateChatContext = React.createContext();

```

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

Лістинг запису даних в контекст:

```
<CreateChatContext.Provider value={socketFunctions}>
  <ChatListParent/>
  <ChatBlockParent/>
</CreateChatContext.Provider>
```

Лістинг отримання даних:

```
const { findUsersForChatting } = useContext(CreateChatContext);
```

### Висновки

У результаті виконання даного розділу було спроектовано загальну схему програми, описано алгоритми більшості функцій програми, як серверних так і клієнтських. Також було описано логіку обробки запитів сервером та взаємодію клієнт-сервер за допомогою сокетів. Крім цього було наведено логіку збереження тимчасових даних за допомогою Redux. На даному етапі було завершено написання основного програмного коду.

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

## РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ

### 3.1 Опис роботи з програмним додатком

Після запуску програми ми бачимо форму з пропозицією залогінитись або відразу бачимо чат, залежно від того, чи виконував користувач вхід до цього. Форму входу зображено на Рис.3.1:

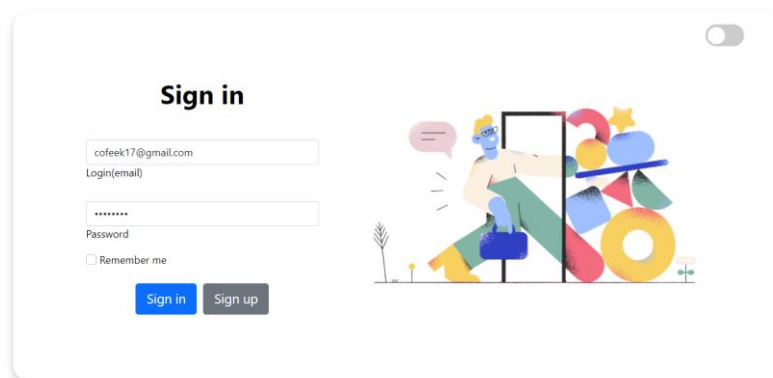
The image shows a light-themed login form titled "Sign in". It features two input fields: the first is labeled "Login(email)" and contains the text "cofeek17@gmail.com"; the second is labeled "Password" and contains a masked password "\*\*\*\*\*". Below the password field is a checkbox labeled "Remember me". At the bottom are two buttons: "Sign in" (blue) and "Sign up" (grey). To the right of the form is a colorful illustration of a person with a blue face and yellow hair, wearing a green shirt and blue pants, carrying a blue bag and holding a blue stick. The person is standing next to a large, colorful, abstract shape that resembles a stylized letter 'E' or a door frame. The background of the illustration is white with some small green plants.

Рис.3.1. Форма входу в акаунт

На цій формі ще можна побачити кнопку “sign up” для зміни типу форми на реєстраційну, checkbox “remember me” зі збереженням паролю та логіну, дана функція потрібна для подальших входів на сторінку, якщо юзер вирішить вийти і не захоче наступного разу вводити дані вручну. Також на сторінці зображено повзунок, що змінює тему зі світлої на темну. Приклад темної теми зображено на Рис.3.2:

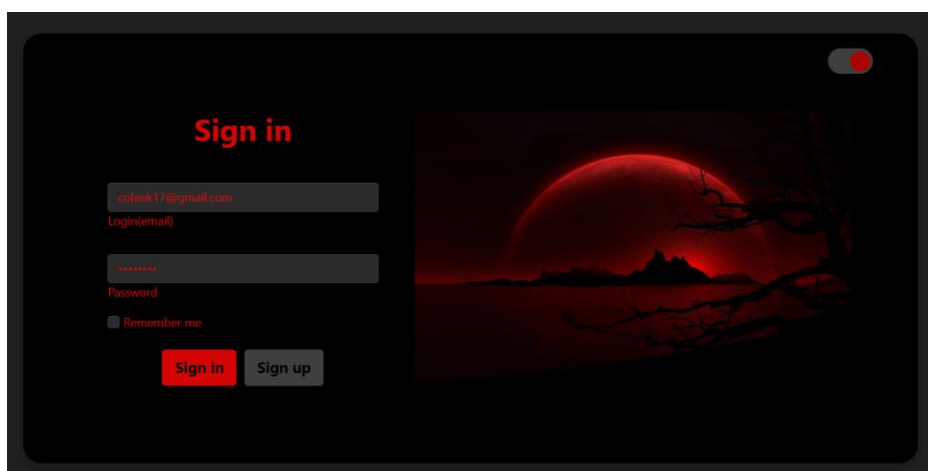
The image shows a dark-themed login form titled "Sign in" in red text. It features two input fields: the first is labeled "Login(email)" and contains the text "cofeek17@gmail.com"; the second is labeled "Password" and contains a masked password "\*\*\*\*\*". Below the password field is a checkbox labeled "Remember me". At the bottom are two buttons: "Sign in" (red) and "Sign up" (grey). To the right of the form is a dark illustration of a person with a blue face and yellow hair, wearing a green shirt and blue pants, carrying a blue bag and holding a blue stick. The person is standing next to a large, dark, abstract shape that resembles a stylized letter 'E' or a door frame. The background of the illustration is dark with some small green plants.

Рис.3.2. Темна тема форми з входом в акаунт

## Sign up

Login(email)

Password

Full name

Phone



Рис.3.3. Форма реєстрації

Після вдалого входу на сторінку користувачу буде відображено список його чатів з можливістю пошуку користувачів для нового чату, біля цього пошукового блоку буде ще кнопка(бургер-меню), при натисканні на яку буде випадати список з можливістю виходу з акаунту та зміною теми. Сторінку зображено на Рис.3.4:

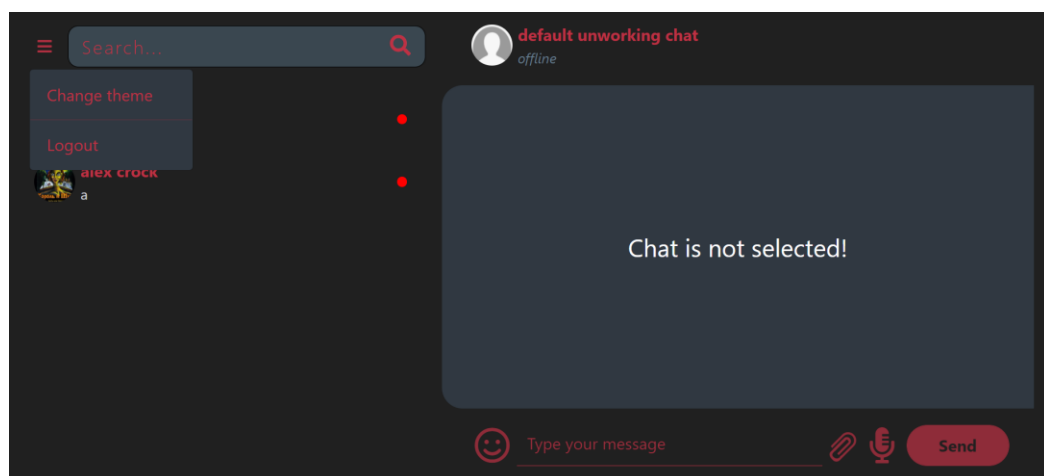


Рис.3.4. Початкове вікно чату з активованим бургер-меню

При натисканні на чат, блок з повідомленнями змінюється. У ньому з'являються повідомлення, якщо цей чат не новий, або просто зникає напис, що сповіщає про невибраний чат. Після чого користувач може записати повідомлення та відправити його іншому користувачу. Результат вибирання чату можна побачити на Рис.3.5:

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

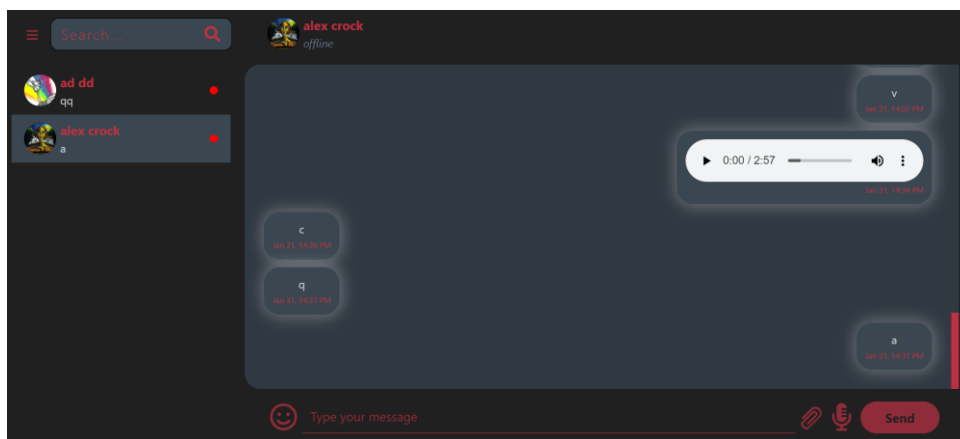


Рис.3.5. Результат обрання існуючого чату

При натисканні на смайлик буде відображено блок зі смайликами, при натисканні на яких повідомлення, що вводив користувач буде доповнене обраним смайликом вкінці рядка. Результат активації попапу смайликів та натиску на одного з них зображено на Рис.3.6:

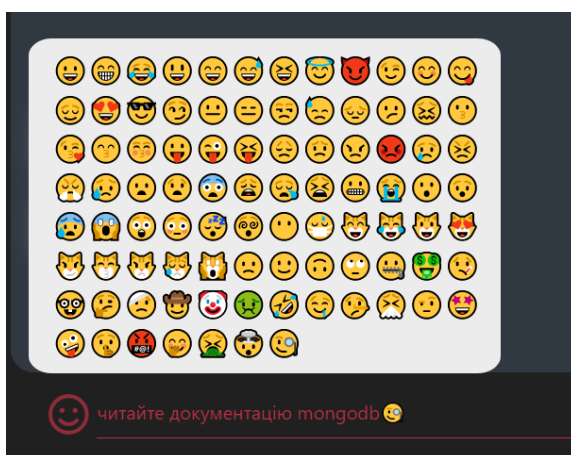


Рис.3.6. Результат натискання на смайлики

Також повідомлення можна відправити та побачити що користувач, який у мережі пише повідомлення. Результат цих дій зображено на Рис.3.7:

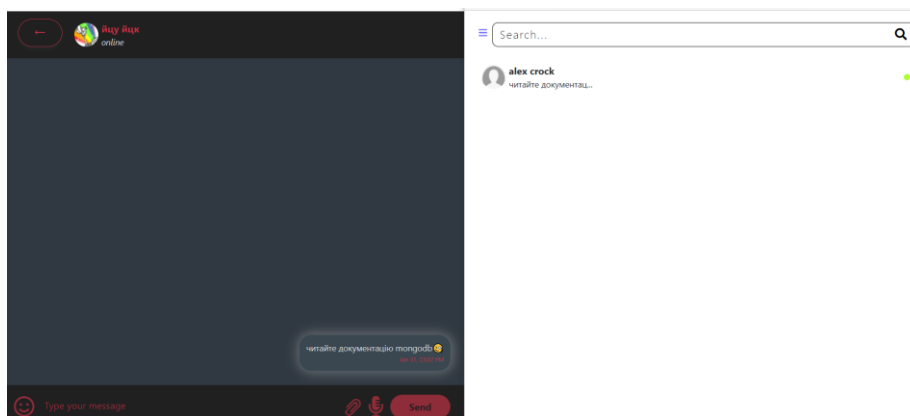


Рис.3.7. Результат відправки повідомлення

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

Результат пошуку користувача(другий знайдений користувач виділений через те, що було на нього наведено):

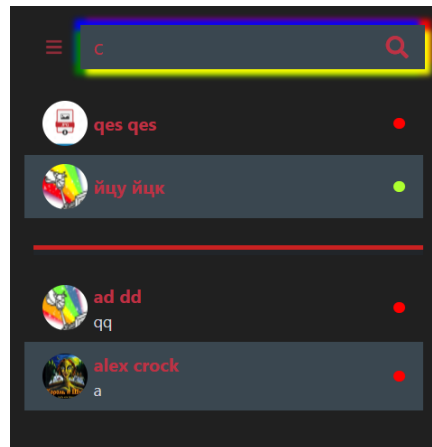


Рис.3.8. Результат пошуку користувача

Також користувач може записати аудіо, відео або відправити збережений файл. Після чого користувачу буде задано запит на підтвердження відправки. Результат можна побачити на Рис.3.9:

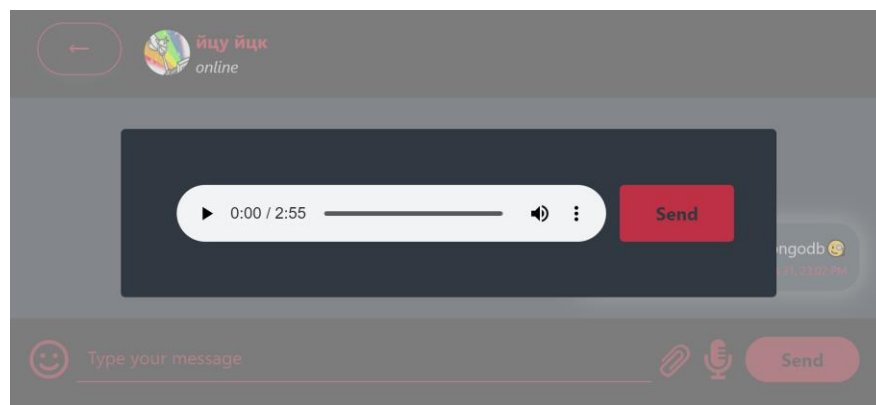


Рис.3.9. Запит на підтвердження відправки аудіо

При натисканні поза блоком повідомлення не буде відправлено, а при натисканні на кнопку “send”, дія буде виконана:

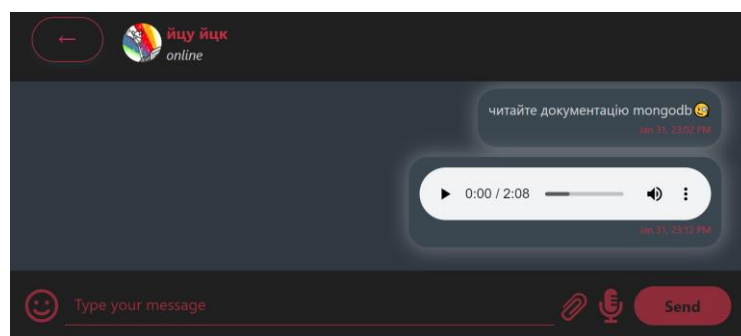


Рис.3.10. Результат відправки аудіо-повідомлення

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

Також тип запису можна змінити. Для цього потрібно натиснути праву клавішу миші на мікрофон, або відеокамеру на панелі, що відповідає за запис повідомлень:

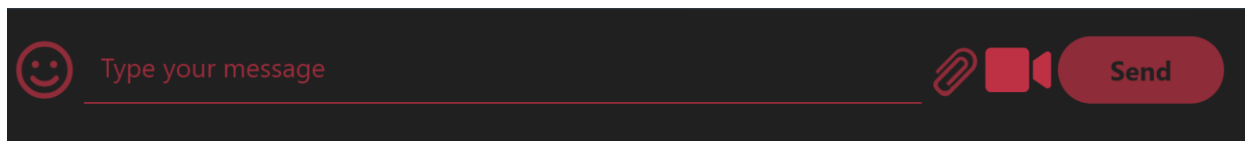


Рис.3.11. Режим відео

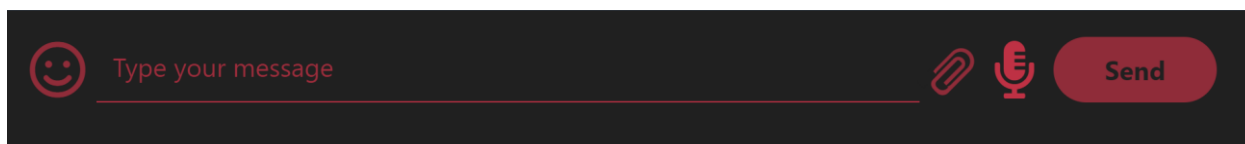


Рис.3.12. Режим аудіо

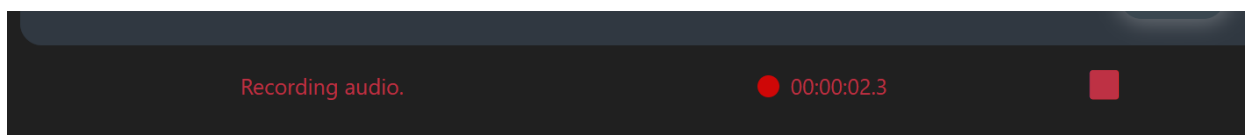


Рис.3.13. Режим запису

Також при написанні повідомлення його можна декорувати, для цього потрібно виділити текст та натиснути правою клавішею миші по ньому, після чого з'явиться панель з кнопками при натисканні на які буде відбуватись редагування тексту. Приклад панелі та її робот показано на Рис.3.14:

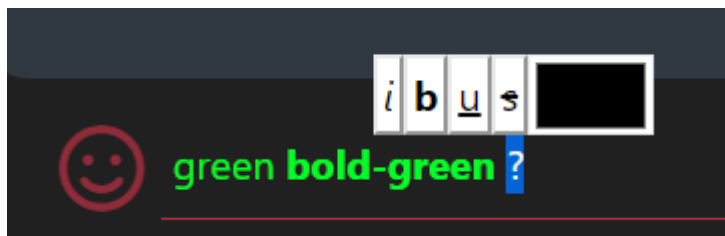


Рис.3.14. Приклад роботи з панеллю для декорування тексту

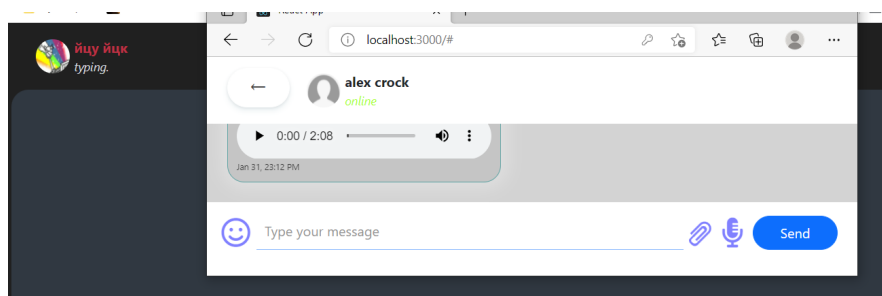


Рис.3.15. Приклад відображення як відбувається відображення друкування повідомлення у РРЧ

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

### 3.2. Тестування роботи програмного забезпечення

Під час виконання даного розділу буде показано те, як проводилось функціональне ручне тестування.

Таблиця 1 – Тестування входу в акаунт

Номер	1
Заголовок	Вхід в акаунт за умови введення коректних даних
Передумова	Відкрите початкове вікно сайту <a href="http://localhost:3000/">http://localhost:3000/</a> з формою для авторизації
Кроки	Очікуваний результат
Вводиться значення в поле “Login”, що складається з латинських літер, кирилиці, чисел та собачки	У полі “ Login ” відображається считане значення
Вводиться значення в поле “Password”, що складається з латинських літер, кирилиці та чисел	У полі “ Password ” відображається считане значення, результат цього кроку та попереднього зображено на Рис.3.16
Натискаємо під заповненою формою на кнопку “Sign in”	З’являється лоадер, на деякий час, після чого він зникає та відбувається оновлення вікна. У ньому відображається вікно з поточним чатами та вікно, що вказує на відсутність вибраного чату, результат зображено на Рис. 3.17



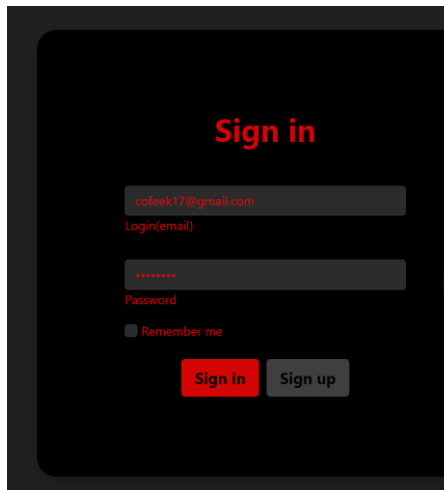


Рис.3.16. Зображено результат считування даних у поля

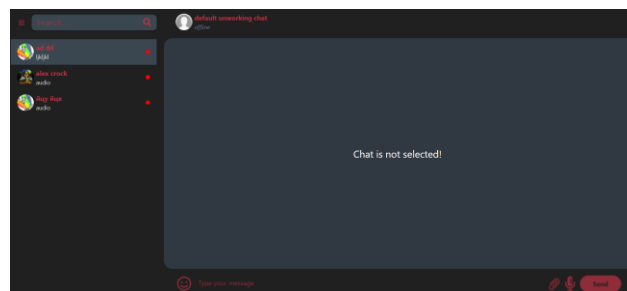


Рис.3.17. Зображено результат входу в акаунт

Таблиця 2 – Тестування відображення статусу

Номер	2
Заголовок	Відображення статусу користувача іншому користувачу
Передумова	Виконано вхід в один акаунт на сайті <a href="http://localhost:3000/">http://localhost:3000/</a> з одного браузера та в інший з іншого. Ці акаунти мають спільний чат. В обох акаунтах вибрано цей чат.
Кроки	Очікуваний результат
Натискаємо на бургер-меню та клікаємо на кнопку “Logout” в одному з акаунтів та переходимо в браузер з акаунтом, з якого не виходили	У заголовку чату змінюється статус співрозмовника з “online” на “offline”, результат зображено на Рис.3.18

Повертаємось до браузера з акаунтом, з якого вийшли. Заповнюємо поля у формі, натискаємо на кнопку “Sign in” під формою та переходимо в браузер з акаунтом, з якого не виходили	У заголовку чату змінюється статус співрозмовника з “offline” на “online”, результат зображено на Рис.3.19
В акаунті з якого було виконано вхід та вихід натискаємо на чат в якому написано ім’я акаунту з якого не було виконано ні вхід, ні вихід. Клікаємо на поле для вводу повідомлення та переходимо в інше вікно	У заголовку чату змінюється статус співрозмовника з “online” на “typing...”, результат зображено на Рис.3.20

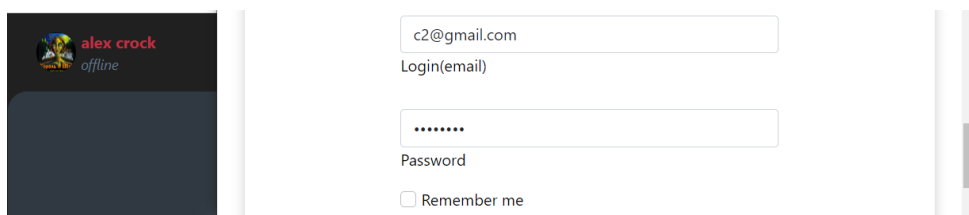


Рис.3.18. Результат відображення офлайн статусу

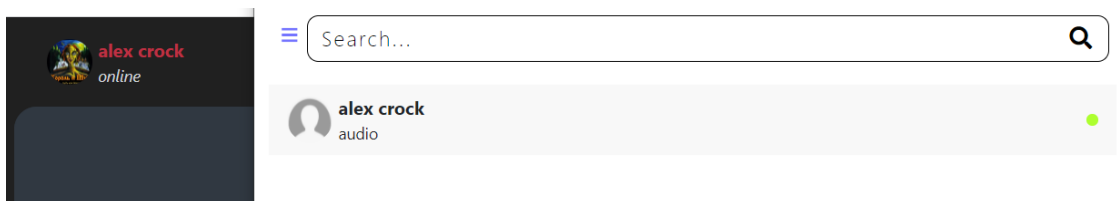


Рис.3.19. Результат відображення онлайн статусу

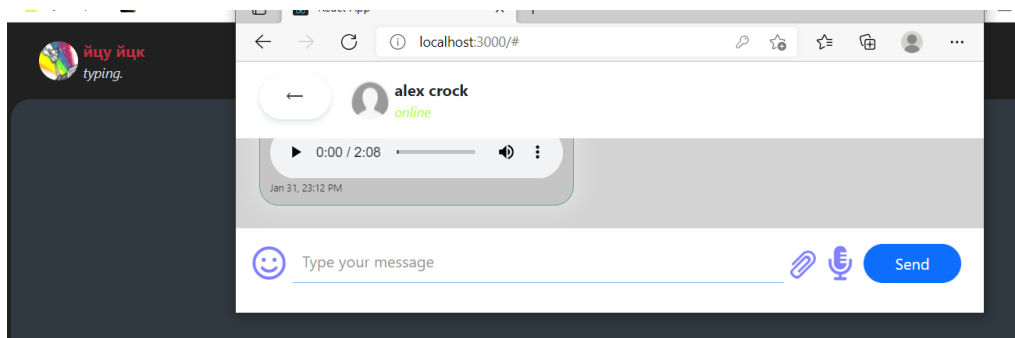


Рис.3.20. Результат початку написання повідомлення

Таблиця 3 - Тестування зміна теми на сторінці

Номер	3
Заголовок	Зміна теми на сторінці
Передумова	Виконано вхід в акаунт на сайті <a href="http://localhost:3000/">http://localhost:3000/</a>
Кроки	Очікуваний результат
Клікаємо по бургер-меню	Відображається блок з двома кнопками. На одній написано “Logout”, а на іншій - “Change theme”
Клікаємо на кнопку “Change theme”	Відбувається зміна кольорів на сайті, результат зображено на Рис.3.21.
Клікаємо на кнопку “Change theme” повторно	Усе повертається до початкового стану(Рис.3.22.)
Клікаємо по бургер-меню та натискаємо на відображену кнопку “Logout”	Відбувається перехід на вікно з формою для входу в акаунт
Натискаємо на повзунок, що знаходиться над малюнком у формі	Відбувається зміна кольорів на сайті, результат зображено на Рис. 3.23.
Повторно натискаємо на повзунок, що знаходиться над малюнком у формі	Усе повертається до початкового стану

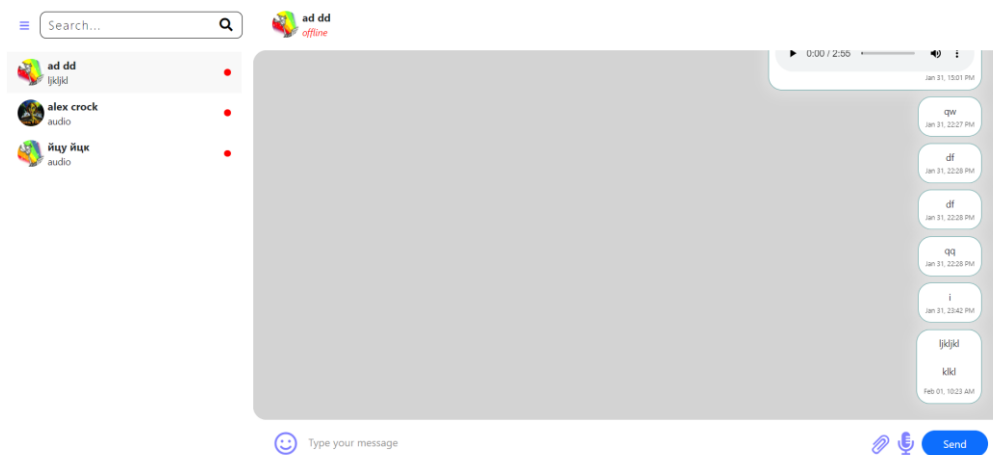


Рис.3.21. Результат перемикання теми

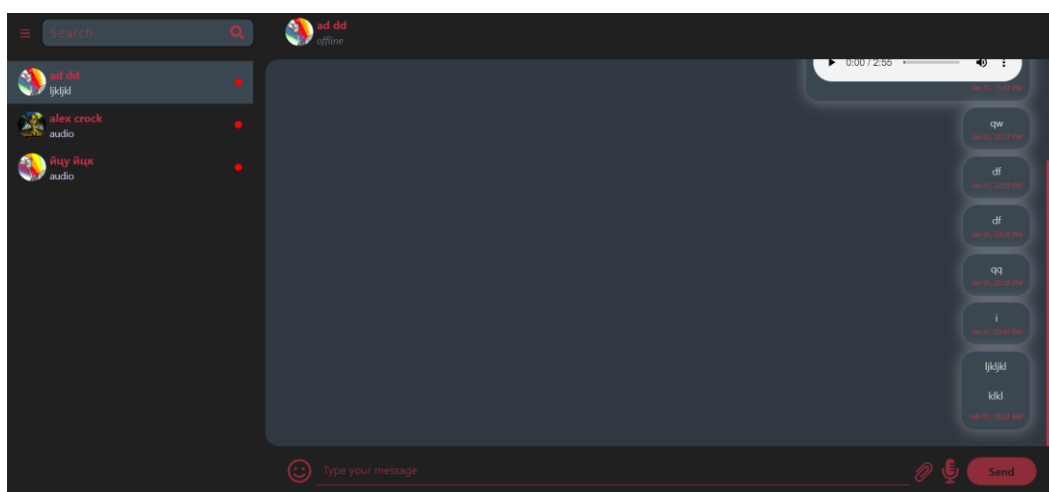


Рис.3.22. Результат повторного перемикання теми

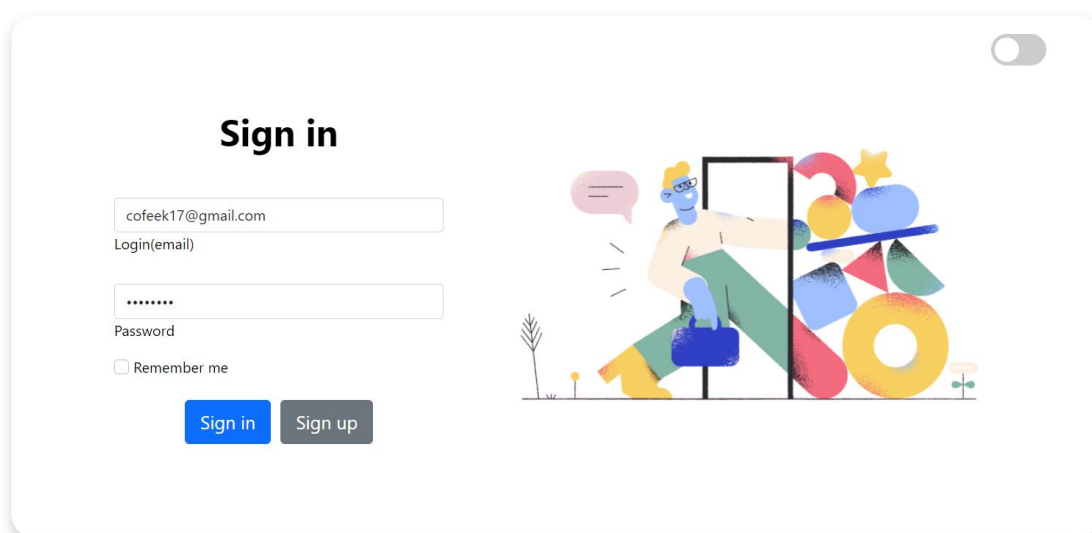


Рис.3.23. Результат перемикання теми на сторінці для входу в акаунт

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				36
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 4 – Тестування запис та відправка аудіо-повідомлення

Номер	4
Заголовок	Запис та відправка аудіо-повідомлення
Передумова	Виконано вхід в акаунт на сайті <a href="http://localhost:3000/">http://localhost:3000/</a> та обрано чат результатом натискання на один з блоків з відповідного списку
Кроки	Очікуваний результат
Клікаємо лівою клавішею миші по мікрофону	Панель, що відповідає за запис та відправку повідомлень змінюється, результат зображено на Рис.3.24.
Чекаємо декілька секунд та натискаємо на відображену кнопку “Зупинити запис”	Панель, що відповідає за запис повідомлень повертається до початкового стану та відображається блок, де можна прослухати записане аудіо та кнопка “Send” для відправки, результат зображено на Рис.3.25.
Клікаємо на утворену кнопку “Send”	Повідомлення відображається у чаті



Рис.3.24. Результат початку запису

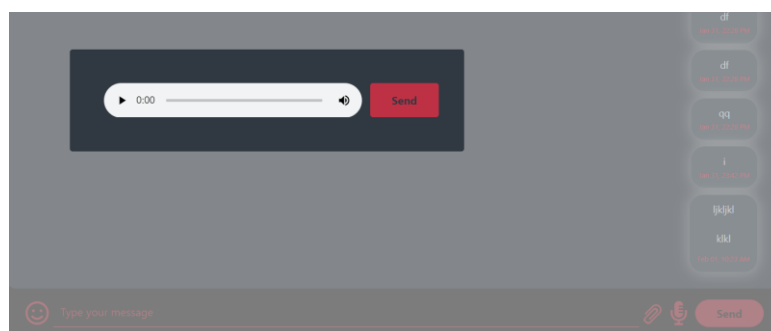


Рис.3.25. Результат зупинки запису

Таблиця 5 - Тестування емоцій

Номер	5
Заголовок	Доповнення поля, що відповідає за запис текстового повідомлення емоцією
Передумова	Виконано вхід в акаунт на сайті <a href="http://localhost:3000/">http://localhost:3000/</a> та обрано чат результатом натискання на один з блоків з відповідного списку
Кроки	Очікуваний результат
Клікаємо лівою клавішею миші по смайлику, що знаходить ліворуч від блоку з записом текстового повідомлення	З'являється блок з емоціями які можна дописати в кінець повідомлення, результат зображено на Рис.3.26
Клікаємо на один зі смайликів, відображених у створеному блоці	Поле, що відповідає за запис текстових повідомлення доповнюється вибраним смайликом, доказ зображено на Рис.3.27
Клікаємо поза блоком з емоціями	Блок з емоціями зникає

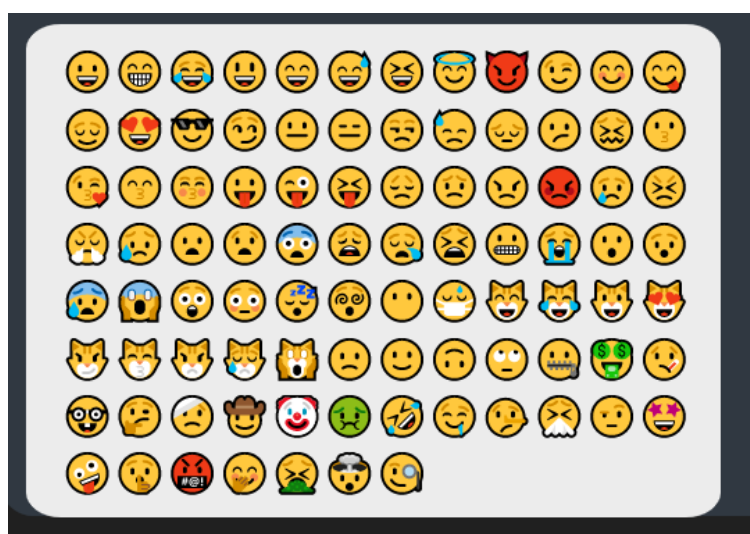


Рис.3.26. Результат кліку по смайлику з панелі для запису



Рис.3.27. Результат кліку по смайлику з блоку зі смайликами

Таблиця 6 – Bug reporter

Короткий опис	При пошуку користувачів для створення нового чату відображаються користувачі, ат из якими вже є
Проект	Назва: “Чат з оновленням даних у режимі реального часу” Посилання: <a href="http://localhost:3000/">http://localhost:3000/</a>
Компонент програми	Пошук користувачів для створення нових чатів
Номер версії	1.0.0
Серйозність	Major(значна)
Пріоритет	P2 середній
Статус	Призначений
Автор	Вербовський Олександр
Призначений на	Вербовського Олександра
Оточення ОС/Браузер	Windows 10/Chrome(Версія 97.0.4692)
Кроки до відтворення	<ol style="list-style-type: none"> <li>1. Шукаємо користувача ad dd за допомогою заповнення поля у лівому верхньому куті</li> <li>2. Натискаємо на нього</li> <li>3. Заповнюємо поле з повідомленням</li> </ol>

	4. Натискаємо кнопку Send 5. Шукаємо знову користувача ad dd за допомогою заповнення поля у лівому верхньому куті
Очікуваний результат	Користувача ad dd не буде знайдено
Реальний результат	Користувача ad dd знайдено



Рис.3.28. Відображення багу

Після знаходження багу з пошуком користувачів для нових чатів, його було виправлено.

### Висновки

У цьому розділі була описана документація до чату, яка описує роботу з додатком для користувача.

У результаті тестування було виявлено та виправлено деякі помилки, після виправлення помилок, тестування було проведено повторно, у результаті чого, помилок не було знайдено. На даному етапі програма є стійкою до помилок, як зі сторони користувача, так і зі сторони програми.

Крім цього програма є адаптивною та стійкою для більшості гаджетів.

Також у цьому розділі додано відстеження помилок та повідомлення користувачу про них.



Загалом у процесі виконання цього розділу було завершено розробку та тестування програмного коду.

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

## ВИСНОВКИ

У результаті виконання даного проекту було спроектовано та розроблено чат з можливістю обробки даних у РРЧ.

Під час виконання курсового проекту було проаналізовано головні, можливі, конкуруючі чати, що допомогло визначити основний функціонал та інтерфейс додатку.

Після того, як було проаналізовано конкуруючі продукти та визначено їх головний функціонал, було розібрано та реалізовано можливість входу в акант, реєстрацію, зміну поточної теми зі збереженням інформації про поточну тему на пристрої, відправку повідомлень, пошук та створення нових чатів, запис медіа-меседжів, відправку файлів та декорування тексту.

Після проходження практики було детально розібрано Node.js, бібліотеки React та Redux та можливість відправки даних від клієнта до сервера у РРЧ за допомогою сокетів.

Також було здобуто базові практичні навички з роботою з не реляційною базою даних MongoDB, що дозволить у подальшому швидко створювати тимчасові бази даних.

Також під час розробки програми було поглиблено знання бібліотеки React та застосовано їх на практиці, а саме було розібрано функціональні компоненти, хуки, такі як useState, useContext, useEffect та інші і розібрано взаємозв'язок між react та redux у результаті чого було створено тимчасове сховище даних

Крім цього під час проходження було закріплено роботу з мовою програмування js, особливо корисним було навчитись працювати зі зчитуванням даних з вебкамери та мікрофону та збереженням їх у тип Blob.

Крім цього досить цікавим завданням була відправка файлів від клієнта до сервера, так як для файлів, розмір яких більше 1 мб, проста одноразова відправка файлів не підходить, тому було реалізовано “сокетову рекурсію”, детальний лістинг коду зображено у Додатку В.

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

По завершенню реалізації усіх функцій, було проведено функціональне тестування у ручному режимі, після виявлення помилок, їх було виправлено та протестовано додаток повторно. Після того, як усі помилки були виправлені та після повторного тестування не було знайдено жодних помилок, чат було остаточно завершено.

Таким чином була розроблена чат з використанням мови Node.js, React, Redux, MongoDB та сокетів. Також, було вдосконалено навички аналізування, проектування та реалізування функціонального коду. Крім цього було розібрано взаємодію мови Node.js та нереляційної бази даних MongoDB.

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

## СПИСОК ЛІТЕРАТУРИ

1. Зачем нужен Refresh Token, если есть Access Token? – 2017. Режим доступа: <https://habr.com/ru/company/Voximplant/blog/323160/>
2. Запись звука JS с микрофона или голосовые комментарии – 2020. Режим доступа: <https://habr.com/ru/post/484196/>
3. Онлайн-руководство по MongoDB – 2021. Режим доступа: <https://metanit.com/nosql/mongodb/>
4. Руководство по Node.js – 2021. Режим доступа: <https://metanit.com/web/nodejs/>
5. JS. Selection – 2019. Режим доступа до ресурсу: <https://developer.mozilla.org/ru/docs/Web/API/Selection>
6. MongoDB Documentation – 2021. Режим доступа: <https://docs.mongodb.com/>
7. Redux Fundamentals – 2021. Режим доступа: <https://redux.js.org/tutorials/fundamentals/part-1-overview>
8. Tutorial: Intro to React – 2022. Режим доступа: <https://reactjs.org/tutorial/tutorial.html>
9. WebSocket - 2021. Режим доступа: <https://nodejsdev.ru/doc/websocket/>

# ДОДАТКИ

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

```

createChat = async (mess) => {
  try {
    const idChat = await chatService.createChat(mess.usersId);
    this.actualChatInfo.idChat = idChat;
    this.actualChatInfo.users = mess.usersId.map(id => {
      return {
        id
      }
    })
    this.socket.emit("createChatDone", {
      idChat
    });
  } catch (e) {
    this.socket.emit("error", e);
  }
}

```

```

/*Модель user*/
const {Schema, model} = require('mongoose');

const UserSchema = new Schema({
  email:{type: String, unique: true, required: true},
  password:{type: String, required: true},
  fullName:{type: String, required: true},
  phone:{type: String, required: true},
  salt:{type:Number, required: true},
  isActiveted:{type: Boolean, default: false},
  activationLink:{type: String},
  avatarURL:{type: String, default: ''},
  online:{type:Boolean, default: false}
});

module.exports = model("User", UserSchema);

/*Модель chatUser*/
const {Schema, model} = require('mongoose');

const ChatUserSchema = new Schema({
  idUser: {type: String, required: true},
  idChat: {type: String, required: true}
});

module.exports = model("ChatUser", ChatUserSchema);

/*Модель chat*/
const {Schema, model} = require('mongoose');
const ChatSchema = new Schema({})

module.exports = model("Chat", ChatSchema);

```

```

/*сторона клієнта*/
const createFileMessage = async (blob, type, idChat) => {
  setLoading(true);
  const typeMedia = blob.type.split("/")[0];
  const blobsArr = [];
  for (let startSize = 0; startSize < blob.size; startSize += 100000) {
    if (startSize + 100000 > blob.size)
      blobsArr.push(blob.slice(startSize, blob.size, typeMedia + "/" + type));
    else

```

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        blobsArr.push(
            blob.slice(startSize, startSize + 100000, typeMedia + "/" + type)
        );
    }
    await socketRef.current.emit("createFileStream", {
        type,
        idChat
    });

    let i = 0;

    const sending = i => {
        socketRef.current.off("readyWrite");
        if (i >= blobsArr.length) {
            socketRef.current.emit("closeFileStream", {});
            return;
        }
        socketRef.current.emit("sendFileBlob", {
            blob: blobsArr[i]
        });
        socketRef.current.on("readyWrite", mess => {
            i++;
            sending(i);
        });
    };
    socketRef.current.on("fileCreated", mess => {
        sending(0);
    });
};

/*сторона сервера*/
createFileStream = async (mess) => {
    try {
        const videoTypes = ["mp4", "mov", "wmv", "avi", "avchd", "mkv", "webm", "mpeg-2"];
        const audioTypes = ["m4a", "mp3", "flac", "wav", "wma", "aac", "mpeg"];
        const imageTypes = ["apng", "avif", "gif", "jpeg", "png", "svg", "webp"];

        let mediaType = null;
        let folderHref = null;
        if (videoTypes.includes(mess.type)) {
            mediaType = "video";
            folderHref = "../media/messageVideos";
        }
        if (audioTypes.includes(mess.type)) {
            mediaType = "audio";
            folderHref = "../media/messageAudios";
        }
        if (imageTypes.includes(mess.type)) {
            mediaType = "image";
            folderHref = "../media/messagePhotos";
        }
        if (mediaType) {
            const createdMessage = await chatService.createMessage(this.user.id, mess.idChat,
mediaType);
            this.messageId = createdMessage._id.toString()
            this.actualWritingFile = this.messageId + "." + mess.type;
            this.absolutePathActualWritingFile = path.join(__dirname, folderHref,
this.actualWritingFile);
            fs.writeFile(this.absolutePathActualWritingFile, '', () =>
this.socket.emit("fileCreated", {}));
        }
    } catch (e) {
        this.socket.emit("error", e);
    }
};

```

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
}

sendFileBlob = (mess) => {
    fs.appendFile(this.absolutePathActualWritingFile, mess.blob, () => {
        this.socket.emit("readyWrite", {})
    });
}

closeFileStream = async (mess) => {
    try {
        const message = await chatService.updateMessage(this.messageId, this.actualWritingFile)
        this.absolutePathActualWritingFile = "";
        this.actualWritingFile = "";
        this.messageId = ""
        this.socket.emit("endWrited", {})
        this.sendMessageForUsersFromActualChat(message);
    } catch (e) {
        this.socket.emit("error", e);
    }
}

```

		Вербовський О.Ю.			ДУ «Житомирська політехніка».21.121.05.000 - ІПЗ	Арк.
		Морозов А.В.				48
Змн.	Арк.	№ докум.	Підпис	Дата		