

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Онлайн чат в реальному часі»

Виконав студент групи ЗПЗ-20-1
ІЩЕНКО Андрій Юрійович

Керівник роботи:
ВЛАСЕНКО Олег Васильович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2024

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«__» _____ 20__ р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ІЩЕНКО Андрій Юрійович**

Керівник роботи: **ВЛАСЕНКО Олег Васильович**

Тема роботи: **«Онлайн чат в реальному часі»,**

затверджена Наказом закладу вищої освіти від **«23» лютого 2024 р., №75с**

Вихідні дані для роботи: **розробка онлайн чату в реальному часі**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів,
що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Власенко О.В.	26.02.2024р.	26.02.2024р.
2	Власенко О.В.	26.02.2024р.	26.02.2024р.
3	Власенко О.В.	26.02.2024р.	26.02.2024р.

РЕФЕРАТ

Випускна робота бакалавра складається з комплексної програмної реалізації онлайн чату, а також пояснювальної записки. Пояснювальна записка до дипломної роботи містить 72 сторінок, 29 ілюстрацій, 17 таблиць і 30 літературних джерел.

Основною метою даної роботи є розробка та реалізація веб-додатку для онлайн спілкування з використанням технологій React та Socket.IO.

У роботі використано наступні методи: розробка фронтенду з використанням React, налаштування взаємодії з сервером за допомогою Socket.IO, тестування веб-додатку.

В результаті проведених досліджень було створено функціональний веб-додаток для обміну повідомленнями в реальному часі з можливістю реєстрації користувачів та відстеження їх онлайн статусу. Додаток забезпечує швидку та надійну передачу даних, що підвищує зручність користування. Також було проведено аналіз ефективності роботи системи при високих навантаженнях, що дозволило оптимізувати її продуктивність.

Додатково, веб-додаток має інтуїтивно зрозумілий інтерфейс користувача, що значно полегшує процес взаємодії з системою. Було впроваджено ряд функцій, таких як підтримка приватних та групових чатів, висилання мультимедійних файлів, а також система сповіщень, яка інформує користувачів про нові повідомлення.

Рекомендується використання результатів досліджень для подальшого розвитку та вдосконалення веб-додатків для онлайн спілкування. Результати цієї роботи можуть бути застосовані як в освітніх цілях, так і для комерційних проектів, що потребують ефективних рішень для онлайн комунікації.

КЛЮЧОВІ СЛОВА: ЧАТ, ВЕБ-ДОДАТОК, MERN, SOCKET.IO

					ІПЗ.КР.Б – 121 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Онлайн чат в реальному часі Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Іщенко А.Ю.					2	148
Керівник		Власенко О. В.				Житомирська політехніка, група ЗІПЗ-20-1		
н. контр		Тичина О. П						
Зав. каф.		Вакалюк Т. А.						

ABSTRACT

The bachelor's thesis consists of a comprehensive software implementation of an online chat room, as well as an explanatory note. The explanatory note to the thesis contains 72 pages, 29 illustrations, 17 tables and 30 references.

The main purpose of this work is to develop and implement a web application for online communication using React and Socket.IO technologies.

The following methods have been used in the work: frontend development using React, setting up interaction with the server using Socket.IO, testing the web application.

As a result of the research, a functional web application for real-time messaging with the ability to register users and track their online status was created. The application ensures fast and reliable data transfer, which improves user experience. We also analysed the system's performance under high loads, which allowed us to optimise its performance.

Additionally, the web application has an intuitive user interface, which greatly facilitates the process of interacting with the system. A number of features have been implemented, such as support for private and group chats, sending multimedia files, and a notification system that informs users of new messages.

It is recommended to use the research results for further development and improvement of web applications for online communication. The results of this work can be applied both for educational purposes and for commercial projects that require effective solutions for online communication.

KEYWORDS: CHAT, WEB APPLICATION, MERN, SOCKET.IO

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	2
ABSTRACT.....	3
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	5
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ СИСТЕМИ ОНЛАЙН ЧАТУ В РЕАЛЬНОМУ ЧАСІ.....	9
1.1 Постановка задачі.....	9
1.2 Аналіз аналогів програмного продукту	10
1.3 Обґрунтування вибору інструментальних засобів	16
Висновки до розділу 1.....	20
РОЗДІЛ 2. ПРОЕКТУВАННЯ ОНЛАЙН ЧАТУ В РЕАЛЬНОМУ ЧАСІ....	21
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури	21
2.2 Розробка бази даних системи	29
2.3 Проектування та реалізація алгоритмів роботи системи	33
2.4 Реалізація функціоналу онлайн чату в реальному часі	37
Висновки до розділу 2.....	51
РОЗДІЛ 3. ІНТЕРФЕЙС І ПОРЯДОК РОБОТИ З ОНЛАЙН ЧАТОМ В РЕАЛЬНОМУ ЧАСІ	53
3.1 Порядок встановлення та налаштування параметрів платформи.....	53
3.2 Інтерфейс та порядок роботи.....	55
3.3 Тестування роботи онлайн чату.....	64
Висновки до розділу 3.....	65
ВИСНОВКИ	67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	69
ДОДАТКИ	72

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MERN – MongoDB, Express, React, Node.

UI – (User Interface) інтерфейс користувача

ПЗ – програмне забезпечення

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми полягає в тому, що в умовах стрімкого розвитку інформаційних технологій, особливо в контексті цифрової трансформації, ефективний засіб комунікації в режимі реального часу [1] стає надзвичайно важливим у різних сферах людської діяльності. Від онлайн консультацій у сфері охорони здоров'я до спільної роботи над проектами у сфері розробки програмного забезпечення, потреба у швидкому і зручному способі обміну інформацією залишається актуальною. Дослідження показують, що існуючі рішення не завжди відповідають вимогам ефективності, масштабованості та безпеки. Тому розробка нового онлайн чату в реальному часі є важливим кроком у поліпшенні комунікаційних процесів у веб-застосунках.

Додатковий аналіз результатів досліджень в цій області, а також аналіз робіт розробників програмного забезпечення [2], підтверджує необхідність та перспективність подальшого розвитку і вдосконалення засобів комунікації в онлайн середовищі.

Мета роботи є розробка та впровадження онлайн чату в реальному часі. Ця робота спрямована на створення функціонального і ефективного інструменту комунікації для веб-застосунків, що дозволить користувачам обмінюватися повідомленнями миттєво та ефективно.

Загальна мета полягає в створенні потужного та гнучкого інструменту комунікації, який відповідає сучасним вимогам ефективності та безпеки і може бути успішно використаний у різних галузях людської діяльності.

Завдання роботи:

- Аналіз існуючих рішень: Провести огляд і аналіз існуючих рішень у сфері онлайн чатів в реальному часі, визначити їх переваги та недоліки з точки зору функціональності, масштабованості та безпеки.
- Проектування архітектури системи: Розробити архітектурну схему та узагальнену структуру системи, враховуючи потреби користувачів у швидкості, безпеці та зручності використання.

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

- Вибір технологічного стеку: вибрати найбільш відповідний засоби для реалізації кожного компонента системи.
- Розробка програмного забезпечення: Реалізувати клієнтську та серверну частини чату в реальному часі з використанням вибраних технологій, забезпечуючи ефективну обробку повідомлень та безпеку даних.
- Тестування та оптимізація: Провести ретельне тестування розробленого чату, виявити та виправити можливі помилки та недоліки, а також оптимізувати продукт для досягнення максимальної швидкості та ефективності.
- Документування та впровадження: Підготувати документацію, яка описує функціональність, архітектуру та використані технології чату в реальному часі. Після завершення розробки впровадити чат у відповідному веб-застосунку та забезпечити користувачів необхідною підтримкою.

Об'єктом дослідження є веб-застосунки, спрямовані на комунікацію в режимі реального часу.

Об'єкт дослідження, який складається з веб-застосунків для миттєвої комунікації, породжує проблемну ситуацію в контексті потреб користувачів у зручних та ефективних інструментах спілкування. Предмет дослідження, а саме розробка онлайн чату в реальному часі, визначає конкретний набір завдань та методів, які досліджуються для вирішення цієї проблеми. Об'єкт і предмет дослідження співвідносяться як загальне (веб-застосунки для комунікації) і часткове (розробка конкретного чату в реальному часі).

Предметом дослідження є використання веб-орієнтованих технологій для розробки системи онлайн чату в реальному часі.

В роботі було використано методи системного аналізу методи об'єктно орієнтованого проектування, веб-програмування та дизайну.

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

Розроблений онлайн чат в реальному часі на базі стеку MERN має значний потенціал застосування у практичній господарській діяльності підприємств різних галузей. Деякі з можливих сфер використання включають:

- Корпоративні комунікації: Впровадження чату в реальному часі для внутрішнього спілкування та спільної роботи співробітників підприємства. Це дозволить швидко обмінюватися ідеями, розв'язувати проблеми та координувати проекти.
- Клієнтська підтримка: Використання чату для надання швидкої та ефективної підтримки клієнтам. Користувачі зможуть звертатися до представників підприємства з будь-яких питань та отримувати негайні відповіді.
- Електронна комерція: Інтеграція чату в реальному часі в онлайн-магазини для консультацій з покупцями, вирішення питань щодо товарів та послуг та покращення обслуговування клієнтів.

Впровадження розробленого програмного продукту у ці сфери може значно підвищити ефективність комунікації, зменшити час вирішення проблем та покращити задоволення клієнтів і співробітників.

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ СИСТЕМИ ОНЛАЙН ЧАТУ В РЕАЛЬНОМУ ЧАСІ

1.1 Постановка задачі

Метою даної роботи є розробка системи онлайн-чату в режимі реального часу з використанням сучасних технологій для забезпечення простоти використання та майбутньої масштабованості.

У сучасному контексті чат-система є основним інструментом комунікації в онлайн-середовищі. Вона дозволяє користувачам миттєво обмінюватися повідомленнями, незалежно від їхнього місця перебування. Чати сприяють особистим розмовам, професійному спілкуванню, управлінню груповими завданнями, організації робочих процесів і проектів.

Оскільки віддалена робота стає все більш поширеною, ефективна комунікація має вирішальне значення для успіху. Система чату в реальному часі може значно покращити комунікацію між членами команди або співробітниками, надаючи зручний і доступний канал для обміну інформацією, думками та ідеями. Цей проект відповідає актуальним вимогам сучасного ринку праці та задовольняє потребу в ефективній віддаленій комунікації. Крім того, розробка чат-системи дозволить поглиблено вивчити веб-розробку та сучасні технології, що стане цінним досвідом для майбутньої кар'єри.

Для досягнення поставленої мети та створення програмного продукту, що відповідає вимогам користувачів, необхідно вирішити декілька завдань.

Потрібно спланувати роботу та розробити дизайн. Це включає в себе аналіз та вибір компонентів Material-UI (MUI) для використання в додатку, налаштування та кастомізацію MUI відповідно до дизайну, а також розробку загальної концепції дизайну на основі обраних компонентів MUI.

Окрім цього потрібно реалізувати клієнтську частину. Це включає встановлення та налаштування середовища розробки для фронтенду, створення користувацького інтерфейсу за допомогою React.js та інтеграцію компонентів для відображення повідомлень та інтерактивного спілкування.

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

Також потрібно реалізувати серверну частину, що вимагає налаштування серверного середовища за допомогою Node.js та Express.js, реалізації серверного чату за допомогою Socket.io для спілкування в режимі реального часу, а також створення системи автентифікації та управління користувачами за допомогою токенів JWT та визначення ролей користувачів і обмежень доступу.

Після цього можна перейти до розробки основного функціоналу, що включає в себе реалізацію логіки аутентифікації та взаємодії з користувачами, створення віддаленої бази даних за допомогою MongoDB для зберігання повідомлень та інформації про користувачів, організацію зв'язку між серверною та клієнтською сторонами за допомогою Socket.io, а також інтеграцію смарт-контракту для забезпечення безпеки та конфіденційності.

Останнім завданням є протестувати та випробувати наш чат. На цьому етапі потрібно буде провести функціональне тестування для перевірки коректної роботи всіх функцій та тестування графічного інтерфейсу для забезпечення коректного відображення та зручності використання.

Розробка системи буде складатися з декількох модулів. Модуль аутентифікації забезпечує ідентифікацію користувачів та контроль доступу. Модуль комунікації забезпечує основну функціональність, дозволяючи користувачам обмінюватися повідомленнями в режимі реального часу. Модуль управління чатами дозволяє користувачам додавати, видаляти та керувати чатами для спрощення спілкування. Модуль повідомлень обробляє, зберігає та відображає повідомлення. Модуль сповіщень надсилає користувачам сповіщення про нові повідомлення або події в чаті. Ці модулі призначені для задоволення функціональних потреб системи, забезпечуючи зручну та надійну взаємодію користувачів у чаті.

1.2 Аналіз аналогів програмного продукту

Перш ніж розпочати розробку програмного продукту, важливо провести аналіз існуючих аналогів у відповідній області. Цей етап дозволяє виявити сильні та слабкі сторони існуючих рішень, зрозуміти потреби користувачів та визначити можливості для удосконалення. У даному розділі буде проведено

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

порівняльний аналіз п'яти онлайн чатів, щоб з'ясувати їхні особливості, функціональність та можливості.

Slack – одна з найпопулярніших платформ для комунікації в командах та організаціях. Slack надає широкий набір інструментів для спільної роботи, обміну повідомленнями та документами, а також інтегрується з різними зовнішніми сервісами (рис. 1.1).

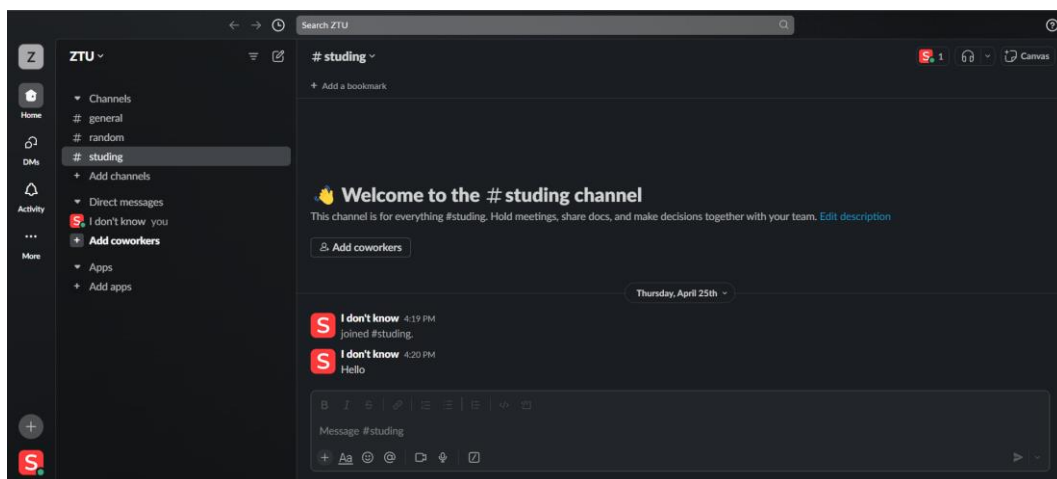


Рис. 1.1 Інтерфейс Slack

Slack має кілька переваг. Він надає широкий функціонал, включаючи інструменти для спільної роботи та спілкування за допомогою текстових, голосових і відеочатів, обміну файлами та інтеграції із зовнішніми сервісами. Гнучкість платформи дозволяє користувачам налаштовувати різні канали для різних проектів або команд, використовувати різні інтеграції та налаштовувати сповіщення відповідно до своїх потреб. Мобільні додатки дозволяють користувачам залишатися на зв'язку зі своєю командою, навіть перебуваючи поза офісом, отримуючи сповіщення та швидко реагуючи на повідомлення. Інтеграція з іншими сервісами, такими як Google Drive, Dropbox і Trello, полегшує спільну роботу та обмін файлами. Крім того, Slack має велику базу знань та підтримку користувачів, що дозволяє швидко вирішувати проблеми та отримувати відповіді на запитання.

Однак Slack також має деякі недоліки. Деякі користувачі вважають його високу ціну порівняно з іншими подібними інструментами, особливо для великих команд або підприємств. У великих командах або проектах обсяг

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

повідомлень може стати величезним, що ускладнює пошук інформації та організацію робочих процесів. Slack, як і більшість онлайн-інструментів, вимагає підключення до Інтернету, що може бути проблематичним під час перебоїв у роботі мережі. Нарешті, використання Slack може бути ресурсоємним на комп'ютерах або мобільних пристроях, особливо якщо відкрито багато чатів або обробляється великий обсяг даних.

Microsoft Teams – інструмент для співпраці та комунікації, розроблений корпорацією Microsoft. Він включає в себе чати, відеоконференції, спільну роботу над документами та інші функції (рис. 1.2).

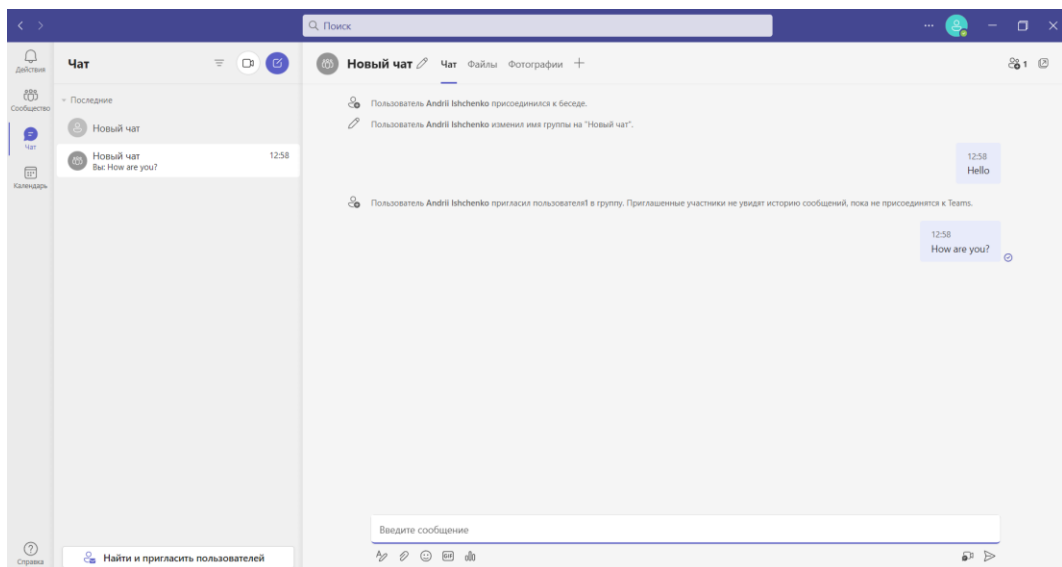


Рис. 1.2 Інтерфейс Microsoft Teams

Microsoft Teams має кілька переваг. Інтеграція з екосистемою Microsoft 365 дозволяє безперешкодно використовувати такі інструменти, як Outlook, Word і Excel. Платформа підтримує відеоконференції, включно з демонстрацією екрану та записом зустрічей, що полегшує віддалену співпрацю. Microsoft Teams також може похвалитися широкою функціональністю, що охоплює текстові чати, спільний доступ до файлів, спільну роботу з документами, наради та завдання. Він забезпечує високий рівень безпеки даних завдяки шифруванню та іншим заходам, що має вирішальне значення для ділового спілкування та обміну конфіденційною інформацією. Наявність мобільних додатків дозволяє користувачам отримувати доступ до інструментів комунікації та співпраці, навіть перебуваючи за межами офісу.

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Однак є й певні недоліки. Деякі користувачі вважають інтерфейс Microsoft Teams складним порівняно з іншими платформами для спільної роботи. Певні функції доступні лише в платних планах, що може бути обмеженням для деяких користувачів. Як і більшість онлайн-інструментів, Microsoft Teams вимагає підключення до Інтернету для роботи, що створює проблему під час перебоїв у мережі. Хоча він добре інтегрується в екосистему Microsoft, інтеграція з іншими сторонніми сервісами може бути менш зручною порівняно з іншими платформами. Крім того, певні функції, такі як онлайн-чат, можуть вимагати встановлення на комп'ютері певного додатку або програми, що може бути незручним для користувачів, які працюють на комп'ютерах з обмеженими правами на встановлення програмного забезпечення.

WhatsApp – популярний месенджер, який дозволяє користувачам обмінюватися повідомленнями, фотографіями, відео та іншими мультимедійними файлами у режимі реального часу (рис. 1.3).

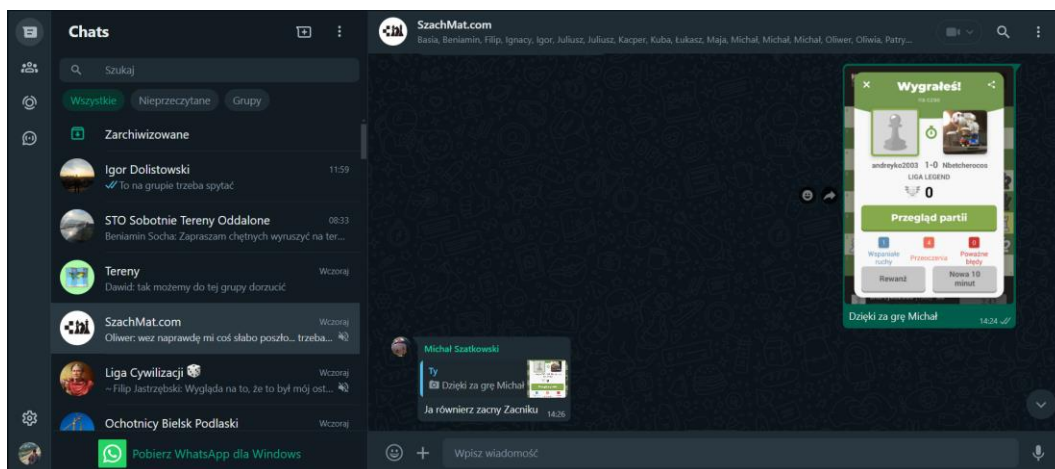


Рис. 1.3 Інтерфейс WhatsApp

WhatsApp має кілька переваг. Це один з найпопулярніших месенджерів у світі з мільярдами користувачів. Додаток дозволяє безкоштовно обмінюватися повідомленнями за допомогою інтернет-з'єднання. WhatsApp також надає різноманітні функції, включаючи текстові повідомлення, голосові та відеодзвінки, обмін файлами та оновлення статусу.

Однак є й певні недоліки. Платформа обмежує кількість учасників у групах до 256, що може бути недостатньо для деяких великих груп або

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

організацій. Використання WhatsApp вимагає стабільного інтернет-з'єднання, що може бути проблематичним у районах з обмеженим доступом до інтернету. Крім того, WhatsApp зазнає критики щодо конфіденційності та безпеки даних, що викликає занепокоєння серед користувачів щодо захисту їхньої особистої інформації.

Telegram – ще один месенджер з великим функціоналом, включаючи шифровані чати, канали для масових повідомлень, вбудовані боти та інші можливості (рис. 1.4).

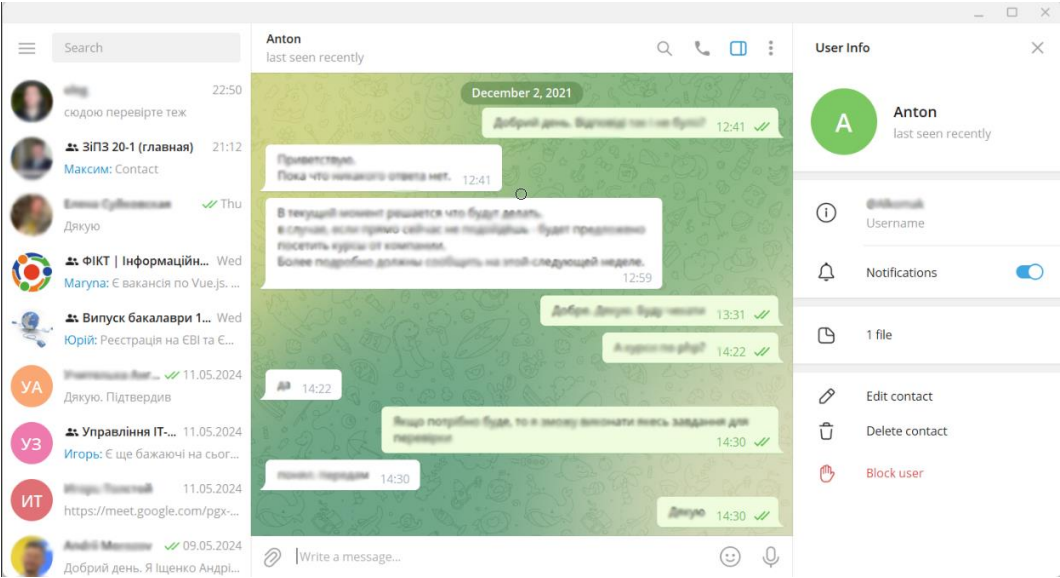


Рис. 1.4 Інтерфейс Telegram

Telegram - ще один месенджер з великим функціоналом, включаючи шифровані чати, канали для масових повідомлень, вбудовані боти та інші можливості.

Telegram має кілька переваг. Він відомий своїм високим рівнем безпеки та шифруванням повідомлень, що робить його привабливим для користувачів, які надають перевагу конфіденційності даних. Платформа підтримує масштабування груп, дозволяючи створювати групи до 200 000 учасників, що робить її придатною для великих спільнот або організацій. Telegram пропонує розширені функції, такі як секретні чати, канали, боти, вбудовані ігри та голосові повідомлення.

Однак є й певні недоліки. У деяких країнах Telegram може бути менш популярним, ніж інші месенджери, що обмежує його доступність для

спілкування в цих регіонах. Донедавна Telegram не підтримував вбудовані голосові та відеодзвінки, що робило його менш привабливим для користувачів, для яких ці функції є пріоритетними. Крім того, відкритість Telegram може призвести до появи спаму та фейкової інформації в чатах і каналах, що може бути проблематичним для деяких користувачів.

Висновки з порівняння

Порівнюючи Slack, Microsoft Teams, WhatsApp та Telegram, можна зазначити, що кожен з них має свої переваги та недоліки [3].

Slack відомий своєю простотою використання та великим набором інтеграцій, але його платформа для комунікації може бути менш зручною для невеликих команд.

Microsoft Teams вирізняється інтеграцією з іншими сервісами Office 365 та надає багато функцій для співпраці в команді, але може бути менш популярним за межами офісного середовища.

WhatsApp, як великою мірою мобільний застосунок, забезпечує широкий охоплення користувачів та простоту використання, але його можливості для бізнесу обмежені.

Telegram славиться своєю конфіденційністю та безпекою, а також розширеними можливостями для спілкування, проте його поширеність може бути меншою, особливо у деяких регіонах.

У порівнянні з вищезгаданими месенджерами, наш чат надає унікальні переваги, які відповідають потребам користувачів. Наш чат працює в режимі реального часу, що дозволяє користувачам миттєво взаємодіяти. Крім того, веб-версія дозволяє користувачам отримувати доступ до чату з будь-якого пристрою без завантаження будь-яких додатків. Ми приділяємо особливу увагу безпеці та конфіденційності, використовуючи передові технології шифрування та захисту даних для захисту даних користувачів. Наш чат пропонує розширені можливості спілкування, включаючи можливість створювати групові чати та обмінюватися файлами, що полегшує спільну роботу в команді. Розроблений з думкою про простоту та зручність використання, наш чат гарантує, що кожен

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

користувач може легко зрозуміти та ефективно використовувати його для своїх потреб. Крім того, ми постійно працюємо над вдосконаленням і розвитком нашого чату, додаючи нові функції та покращуючи існуючі, щоб задовольнити зростаючі потреби наших користувачів.

Отже, наш чат пропонує комбінацію безпеки, зручності та розширених можливостей спілкування, яка робить його ідеальним вибором для командної роботи та особистого спілкування.

Порівняння основних функцій чату (табл. 1.1).

Таблиця 1.1

Порівняння основних функцій чату

Функціонал	Slack	Microsoft Teams	WhatsApp	Telegram	WebChat (Наш чат)
Приватні чати	+	+	+	+	+
Групові чати	+	+	+	+	+
Шифрування	+	+	+	+	+
Відео/аудіо-дзвінки	+	+	+	+	-
Обмін файлами	+	+	+	+	+
Простота використання	±	±	+	+	+
Статус онлайн/офлайн	+	+	+	+	+
Статус повідомлень	+	+	+	+	+
Редагування повідомлень	+	+	-	+	+
Видалення повідомлень	+	±	±	+	+
Історія чатів	+	+	+	+	+
Ціна	±	±	+	±	+

1.3 Обґрунтування вибору інструментальних засобів

Наш веб-додаток буде побудований на основі стеку MERN (MongoDB, Express.js, React.js, Node.js) і використовуватиме технологію Socket.io для забезпечення реального часу у чаті. Для створення користувацького інтерфейсу ми обрали Material-UI (MUI), який надає гнучкість та можливості для швидкого розвитку.

MongoDB [4] – це потужна NoSQL база даних, яка використовує документ-орієнтовану модель зберігання даних. На відміну від традиційних реляційних баз даних, MongoDB зберігає дані у вигляді документів у форматі JSON. Кожен документ представляє собою набір ключ-значення, що дозволяє зберігати структуровані дані без суворої схем.

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

MongoDB має кілька переваг. Гнучка схема даних дозволяє зберігати дані без жорсткої структури, що робить її ідеальною для веб-додатків, які повинні швидко адаптуватися до мінливих вимог. MongoDB легко масштабується як вертикально, так і горизонтально, дозволяючи збільшувати розмір і продуктивність бази даних зі зростанням обсягу даних і навантаження. Архітектура та оптимізація MongoDB забезпечують швидкий доступ до даних та ефективне використання серверних ресурсів. Крім того, MongoDB має велику та активну спільноту користувачів, а також добре задокументовані офіційні ресурси, що полегшує вивчення та використання цієї бази даних.

MongoDB є відмінним вибором для нашого веб-додатку завдяки своїй гнучкості, масштабованості та ефективності. Її можливості в поєднанні з активною спільнотою та добре документованою документацією роблять її ідеальним варіантом для наших потреб.

Node.js [5] - це середовище виконання JavaScript [6], яке дозволяє виконувати код JavaScript на сервері. Воно побудоване на двигуні V8 JavaScript від Google Chrome, що забезпечує високу продуктивність та швидкодію виконання коду.

Node.js має багато переваг. Він працює за принципом асинхронності, що дозволяє виконувати багатозадачні операції, не блокуючи виконання програми, тим самим підвищуючи продуктивність і швидкість роботи додатків. Використовуючи движок V8, Node.js забезпечує швидке виконання коду, що особливо важливо для високонавантажених веб-додатків. Масштабованість - ще одна перевага; Node.js легко масштабується як по горизонталі, так і по вертикалі, сприяючи збільшенню розміру та продуктивності додатків. Завдяки великій та активній спільноті розробників, Node.js постійно розвивається і надає безліч готових рішень та бібліотек для розробки додатків. Крім того, Node.js може похвалитися широким спектром додатків, від веб-серверів до інструментів командного рядка і мобільних додатків, що робить його універсальним рішенням для різних завдань.

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

Node.js є оптимальним вибором для нашого веб-додатку завдяки своїй асинхронності, швидкодії та масштабованості. Його активна спільнота та широкі можливості дозволяють нам ефективно розробляти та підтримувати наш додаток у майбутньому.

Express.js [7] - це легкий та гнучкий веб-фреймворк для Node.js, який допомагає швидко створювати потужні веб-додатки та API. Він надає різноманітні інструменти та функціональність для обробки запитів, маршрутизації, роботи з шаблонами та багато іншого.

Express.js має кілька переваг. Вона має мінімалістичний і зручний інтерфейс, що дозволяє швидко розробляти веб-додатки. Завдяки великій кількості розширень і посилань на сторонні бібліотеки Express.js забезпечує розширюваність, дозволяючи користувачам розширювати його функціональність і використовувати готові рішення. Express.js має зручний механізм маршрутизації для визначення маршрутів та обробки різних типів запитів, що спрощує створення API та розробку веб-додатків. Бібліотека полегшує побудову швидких та ефективних веб-серверів, забезпечуючи високу продуктивність та швидку реакцію на запити. Крім того, Express.js має велику та динамічну спільноту розробників, яка постійно розвивається, надає підтримку, документацію та різноманітні рішення для розробки веб-додатків.

Express.js є надійним та потужним фреймворком, який допомагає розробникам створювати швидкі, масштабовані та ефективні веб-додатки. Його простота використання та гнучкість роблять його ідеальним вибором для нашого проекту, що дозволить нам швидко розпочати та успішно завершити розробку нашого веб-додатку.

Socket.io [8] - це бібліотека JavaScript, яка дозволяє забезпечити багатонаправлений зв'язок в реальному часі між клієнтом та сервером через веб-сокети. Вона дозволяє створювати миттєві та ефективні з'єднання для передачі даних в обидва напрямки, що робить її ідеальним інструментом для реалізації онлайн чатів, спілкування в реальному часі та інших застосувань, які вимагають миттєвої передачі даних.

		Ицено А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

Socket.io має кілька переваг. Його простий і зрозумілий API сприяє швидкому встановленню з'єднання між клієнтом і сервером без зайвих ускладнень, що підвищує зручність використання. Socket.io підтримує різні технології, включаючи веб-сокети та Long Polling, забезпечуючи сумісність з різними браузерами та серверами. Платформа легко масштабується, здатна обробляти численні одночасні з'єднання і великі обсяги даних, що робить її придатною для великих проектів і додатків. Крім того, Socket.io автоматично вирішує проблеми зі зв'язком і відновлює з'єднання у разі втрати зв'язку або інших проблем, забезпечуючи надійну передачу даних.

Socket.io є потужним інструментом для реалізації багатонаправленого зв'язку в реальному часі між клієнтом та сервером. Його простота використання, гнучкість та надійність роблять його ідеальним вибором для нашого проекту, де потрібна миттєва передача даних та спілкування в реальному часі.

React.js [9] - це бібліотека JavaScript для створення користувацьких інтерфейсів. Вона дозволяє розбити веб-сайт на окремі компоненти, які можуть бути легко перевикористані та організовані в ієрархічну структуру, що робить її ідеальним інструментом для розробки складних та динамічних веб-додатків.

React.js має багато переваг. Його висока продуктивність завдяки використанню віртуальної DOM та ефективних алгоритмів повторного рендерингу забезпечує швидку та ефективну роботу навіть з великою кількістю компонентів. Компонентний підхід React.js дозволяє розробникам розбивати веб-додатки на невеликі та багаторазові компоненти, що спрощує розробку та підтримку коду. React також дозволяє розробникам використовувати JavaScript як для клієнтської, так і для серверної частини додатку, що спрощує розробку та підвищує переносимість коду. Крім того, React.js може похвалитися великою екосистемою бібліотек та розширень, що надає розробникам широкий вибір інструментів для розширення та покращення функціональності додатків без особливих зусиль.

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

React.js - це потужна та ефективна бібліотека для розробки користувацьких інтерфейсів, яка пропонує швидку та продуктивну розробку, високу перевикористовуваність компонентів та широкий вибір розширень та бібліотек для розширення функціональності додатків. Вона є ідеальним вибором для нашого проекту, де потрібна динамічна та ефективна розробка користувацького інтерфейсу.

Висновки до розділу 1

У даному розділі ми провели аналіз основних аспектів нашого проекту, що передбачав встановлення зручного та масштабованого онлайн чату в реальному часі. Проаналізувавши сучасні технології, ми визначили основні вимоги до функціоналу, що повинен бути реалізований у нашій програмі.

У процесі аналізу аналогів та схожих програмних рішень, ми вивчили та порівняли кілька популярних онлайн чатів, зосереджуючись на їхніх перевагах та недоліках. Це дозволило нам краще зрозуміти потреби користувачів та забезпечити реалізацію функціоналу, який відповідає їх очікуванням.

Обґрунтувавши вибір інструментальних засобів, ми обрали стек технологій MERN [10] (MongoDB, Express.js, React.js, Node.js) та Socket.io для розробки нашого проекту. Цей вибір обґрунтований їхньою популярністю, ефективністю та зручністю в роботі, що дозволить нам ефективно реалізувати поставлені завдання та досягти мети проекту.

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ОНЛАЙН ЧАТУ В РЕАЛЬНОМУ ЧАСІ

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури

Для наглядної взаємодії між елементами додатку було створено Діаграму варіантів використання (рис. 2.1).

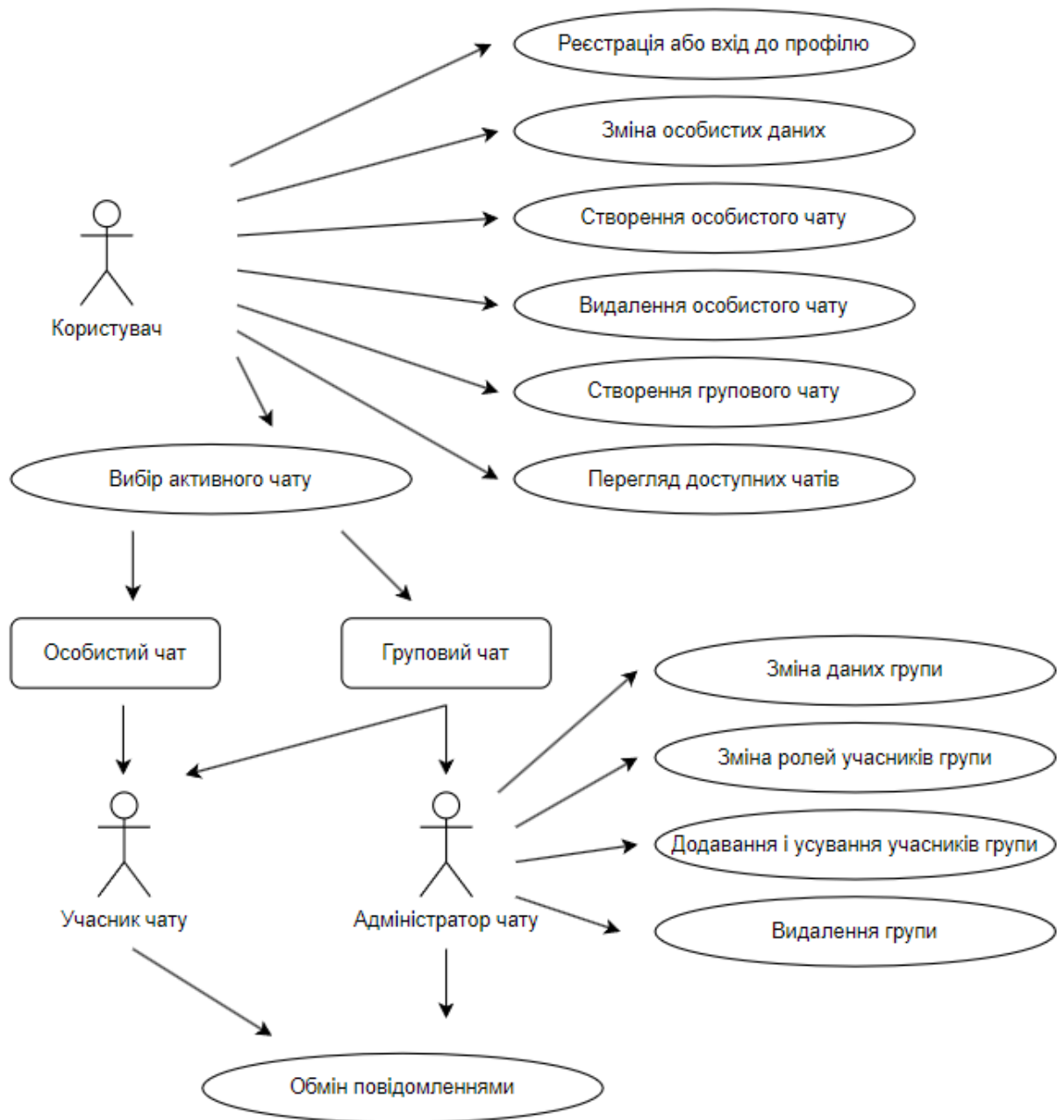


Рис. 2.1 Діаграма варіантів використання

Наш чат надає вебінтерфейс, через який користувачі можуть створювати особисті чати або групи і обмінюватися повідомленнями. Кожний може зареєструватися або вийти до профілю, Створити особистий або груповий чат, видалити особистий чат, переглядати доступні чати і вибрати активний чат. Окрім цього функціональність чату залежить від його типу. В особистих чатах користувачі є звичайними учасниками і можуть тільки обмінюватися повідомленнями між собою. В групових чатах користувачі розділені на дві основні групи: адміністратори і учасники. Адміністратори можуть додавати або усувати учасників, змінювати їх роль в чаті (адміністратор\учасник), видаляти чат і змінювати дані чату, такі як назва, опис і аватар.

Опис прецеденту «Вхід/Реєстрація» (табл. 2.1).

Таблиця 2.1

Опис прецеденту «Вхід/Реєстрація»

Назва варіанту використання	Вхід або Реєстрація
Основний актор	Користувач
Короткий опис	Користувачі можуть зареєструватися або увійти до свого профілю, щоб створити чати і розпочати обмін повідомленнями з іншими зареєстрованими учасниками

Опис прецеденту «Зміна особистих даних» (табл. 2.2).

Таблиця 2.2

Опис прецеденту «Зміна особистих даних»

Назва варіанту використання	Зміна особистих даних
Основний актор	Користувач
Короткий опис	Користувачі можуть змінити свої дані: аватар, ім'я, mail або пароль

Опис прецеденту «Створення особистого чату» (табл. 2.3).

Таблиця 2.3

Опис прецеденту «Створення особистого чату»

Назва варіанту використання	Створення особистого чату
Основний актор	Користувач
Короткий опис	Користувачі можуть створювати чати між двома учасниками, один з яких є користувач, що створив чат. Завдяки цьому можна бути розпочати обмін повідомленнями

Опис прецеденту «Видалення особистого чату» (табл. 2.4).

Таблиця 2.4

Опис прецеденту «Видалення особистого чату»

Назва варіанту використання	Видалення особистого чату
Основний актор	Користувач
Короткий опис	Користувачі можуть видалити особисті чати. В такій спосіб можна очистити історію повідомлень між користувачами і усунути чат з бокового меню вибору чатів.

Опис прецеденту «Створення групового чату» (табл. 2.5).

Таблиця 2.5

Опис прецеденту «Створення групового чату»

Назва варіанту використання	Створення групового чату
Основний актор	Користувач
Короткий опис	Користувачі можуть створювати групові чати між двома і більше учасниками. Користувач що створив чат стає адміністратором групи. Завдяки створення такого чату можна розпочати обмін повідомленнями.

Опис прецеденту «Перегляд доступних чатів» (табл. 2.6).

Таблиця 2.6

Опис прецеденту «Перегляд доступних чатів»

Назва варіанту використання	Перегляд доступних чатів
Основний актор	Користувач
Короткий опис	Користувачі можуть переглянути доступні їм чати в боковому меню

Опис прецеденту «Вибір активного чату» (табл. 2.7).

Таблиця 2.7

Опис прецеденту «Вибір активного чату»

Назва варіанту використання	Вибір активного чату
Основний актор	Користувач
Короткий опис	Користувачі можуть вибрати чат, щоб побачити історію повідомлень і висилати нові.

Опис прецеденту «Обмін повідомленнями» (табл. 2.8).

Таблиця 2.8

Опис прецеденту «Обмін повідомленнями»

Назва варіанту використання	Обмін повідомленнями
Основний актор	Користувач (учасник, адміністратор)
Короткий опис	Користувачі можуть висилати і отримувати нові повідомлення.

Опис прецеденту «Зміна даних групи» (табл. 2.9).

Таблиця 2.9

Опис прецеденту «Зміна даних групи»

Назва варіанту використання	Зміна даних групи
Основний актор	Користувач (адміністратор)
Короткий опис	Користувач, який є адміністратором групи може змінити її дані: назву, опис, аватар.

Опис прецеденту «Зміна ролей учасників групового чату» (табл. 2.10).

Таблиця 2.10

Опис прецеденту «Зміна ролей учасників групового чату»

Назва варіанту використання	Зміна ролей учасників групового чату
Основний актор	Користувач (адміністратор)
Короткий опис	Користувачі, який є адміністратором групи може змінювати ролі всім учасникам чату. Давати їм права адміністратора або їх забирати

Опис прецеденту «Додавання і усунення учасників групи» (табл. 2.11).

Таблиця 2.11

Опис прецеденту «Додавання і усунення учасників групи»

Назва варіанту використання	Додавання і усунення учасників групи
Основний актор	Користувач (адміністратор)
Короткий опис	Користувач, який є адміністратором групи може додати до неї нових учасників або усунути уже доданих.

Опис прецеденту «Видалення групи» (табл. 2.12).

Таблиця 2.12

Опис прецеденту «Видалення групи»

Назва варіанту використання	Видалення групи
Основний актор	Користувач (адміністратор)
Короткий опис	Користувач, який є адміністратором групи може її усунути. Всі повідомлення з того чату зникнуть і він пропаде в усіх учасників з доступних чатів.

Основною ціллю нашої програми є створення зручного та ефективного засобу для обміну повідомленнями в реальному часі, який може бути використаний як для особистого спілкування, так і для групових обговорень. Наша програма спрямована на надання користувачам можливості створювати особисті чати або групові чати, обмінюватися текстовими повідомленнями, а також керувати параметрами своїх чатів, таких як склад учасників, назва, опис

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

тощо. Головна мета полягає в створенні зручного та інтуїтивно зрозумілого інтерфейсу для спілкування, який відповідає сучасним стандартам користувачів.

Вимоги користувачів

1. Авторизація;
2. Редагування своїх особистих даних;
3. Створення особистих і групових чатів;
4. Зручний обмін повідомленнями;
5. Можливість зміни або видалення повідомлення;
6. Можливість переглядати список всіх своїх чатів;
7. Видалення чатів;
8. Зміна даних групового чату;
9. Управління учасниками групи;

Функціональні вимоги:

1. Авторизація у системі: в системі повинна бути представлена можливість реєстрації користувача.
2. Можливість збереження інформації: система повинна зберігати інформацію і надавати можливість користувачу керувати нею.
3. Забезпечення безпечного обміну повідомленнями.

Нефункціональні вимоги:

1. Сприйняття
 - час, необхідний для навчання звичайних користувачів – 3-5 хвилини, а для навчання просунутих користувачів – менше 2-х хвилин;
 - час відгуку для типових задач – не більше 15 секунд, для складних завдань – не більше 30 секунд.
 - інтерфейс має бути зрозумілим та з відповідними підказками.
2. Надійність
 - доступність – час, що витрачається на обслуговування системи не повинно перевищувати 6% від загального часу роботи.

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

- середній час безвідмовної роботи – 15 робочих днів.
- максимальна норма помилок та дефектів в роботі системи – 1 помилка на десять тисяч рядків коду.

3. Продуктивність

- велика кількість користувачів мають змогу переглядати вебресурс одночасно.

4. Можливість експлуатації

- масштабування – система повинна мати можливість збільшувати потужності (продуктивність), зі збільшенням користувачів таким чином, щоб це не вплинуло негативно на її роботу.

Аналіз функціональних вимог дозволив виділити наступні сутності, що забезпечать програмну реалізацію додатку. Діаграму класів, яка використовуються для його роботи (рис. 2.2).

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

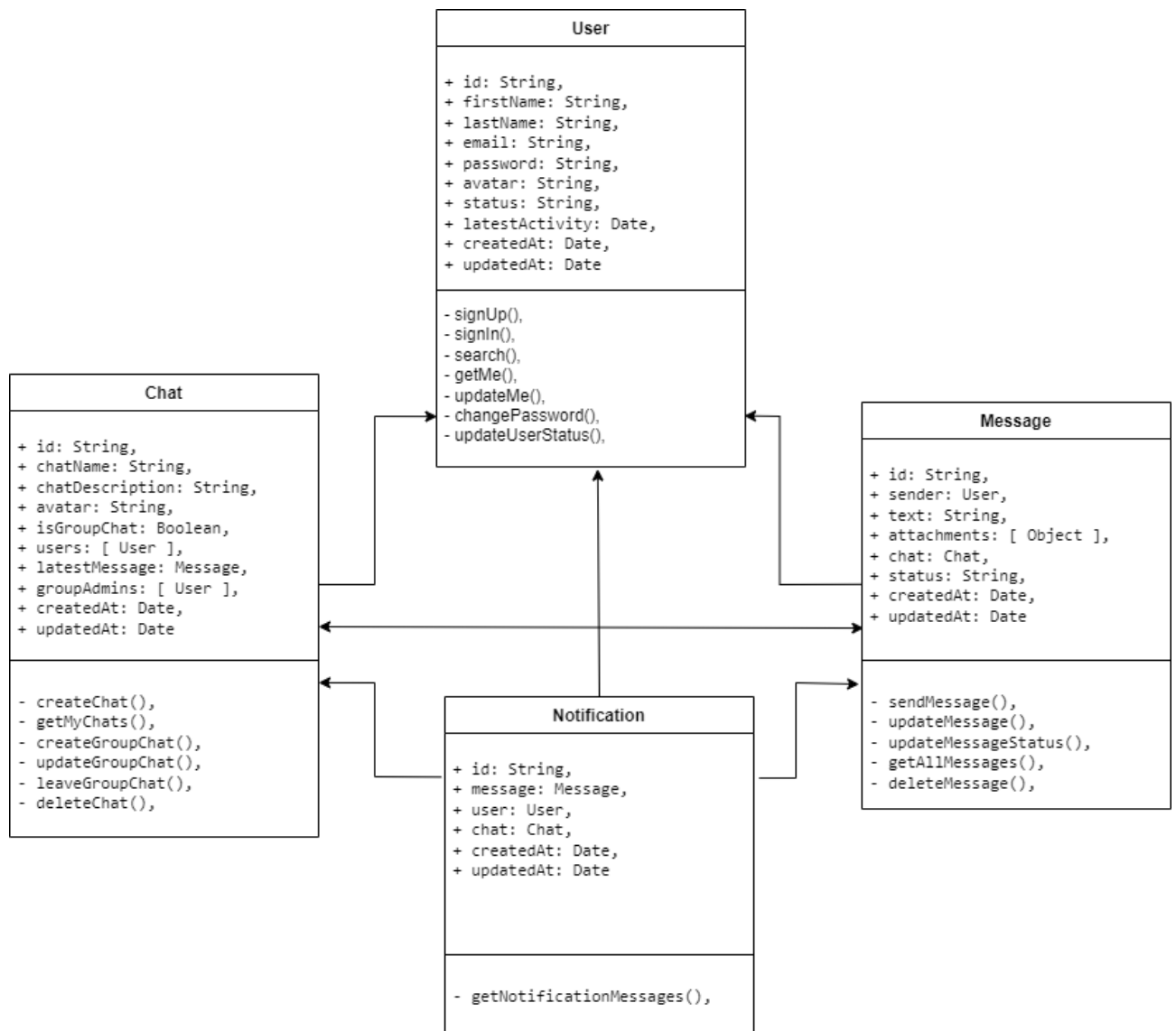


Рис. 2.2 Діаграма класів

User – клас, який відповідає за користувача:

- signUp() – відповідає за реєстрацію.
- signIn() – відповідає за авторизацію.
- search() – відповідає за пошук користувачів.
- getMe() – дає інформацію про користувача.
- updateMe() – відповідає за зміну даних користувача.
- changePassword() – відповідає за зміну пароля.
- updateUserStatus() – відповідає за зміну статусу користувача.

Chat – клас, який відповідає за чат:

- createChat() – відповідає за створення нових особистих чатів.
- getMyChats() – повертає всі доступні чати користувача.

- createGroupChat() – відповідає за створення нових групових чатів.
- updateGroupChat() – відповідає за зміну даних групового чату.
- leaveGroupChat() – відповідає за усування учасників з групи.
- deleteChat() – відповідає за усування чату.

Message – клас, який відповідає за повідомлення:

- sendMessage() – відповідає за створення нових повідомлень.
- updateMessage() – відповідає за зміну вмісту повідомлень.
- updateMessageStatus() – відповідає за зміну статусу повідомлення.
- getAllMessages() – повертає всі повідомлення з чату.
- deleteMessage() – відповідає за видалення повідомлень.

Notification – клас, який відповідає за сповіщення:

- getNotificationMessages() – повертає всі сповіщення про повідомлення.

2.2 Розробка бази даних системи

Для зберігання і керування інформацією про користувачів, повідомлення та інші додаткові дані, пов'язані з функціоналом нашого чату, ми обрали MongoDB як базу даних. MongoDB - це документ-орієнтована база даних, яка дозволяє зберігати дані у вигляді документів у форматі JSON, що робить її ідеальним вибором для розробки веб-додатків.

Схема бази даних нашого чату зображено на рисунку (рис. 2.3).

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

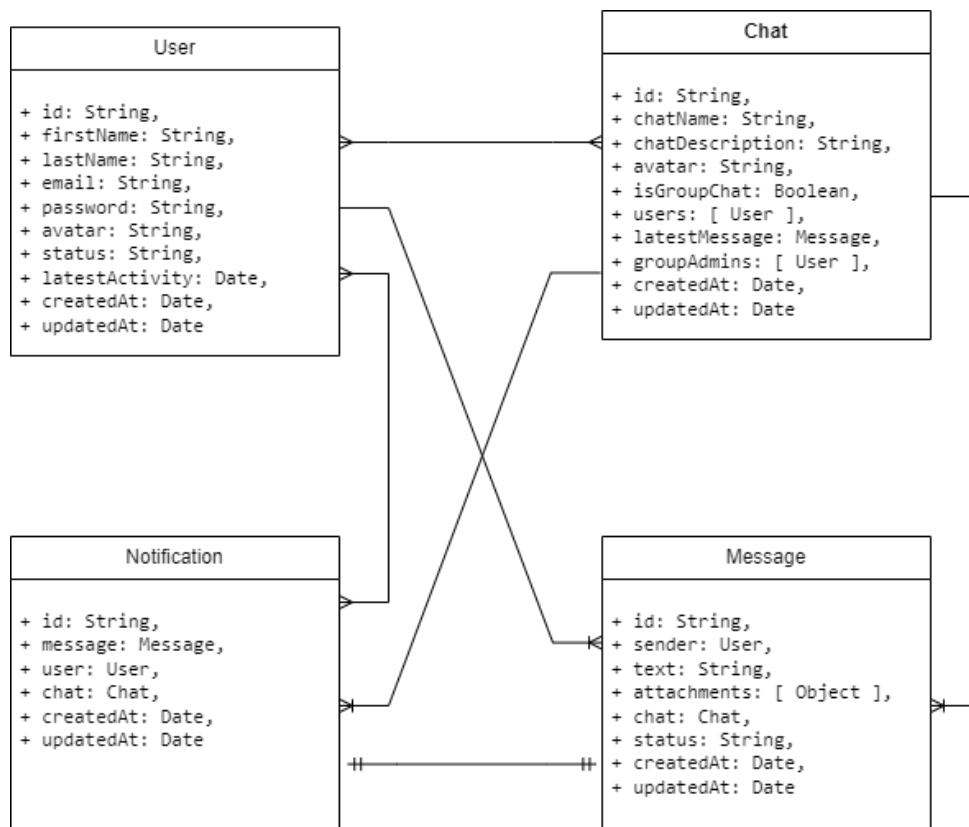


Рис. 2.3 Схема бази даних

База даних системи складається з чотирьох таблиць [11]:

1. User – таблиця, яка зберігає данні користувачів;
2. Chat – таблиця, яка зберігає чати;
3. Message – таблиця, яка зберігає повідомлення;
4. Notification – таблиця, яка зберігає сповіщення.

Опис кожної з таблиць бази даних наведений у наступних таблицях.

1. Опис таблиці User (табл. 2.13).

Таблиця 2.13

Опис таблиці User

Поле	Тип	Опис
id	String	Унікальний ідентифікатор користувача (uuid)
firstName	String	Ім'я
lastName	String	Прізвище
email	String	Адрес email

password	String	Зашифрований пароль
avatar	String	Посилання на фотографію профілю
status	String	Статус користувача. Онлайн/Оффлайн
latestActivity	Date	Дата і час останньої активності
updatedAt	Date	Дата і час останнього оновлення даних користувача
createdAt	Date	Дата і час створення профілю

2. Опис таблиці Chat (табл. 2.14).

Таблиця 2.14

Опис таблиці Chat

Поле	Тип	Опис
id	String	Унікальний ідентифікатор чату (uuid)
chatName	String	Назва чату (тільки для групових чатів)
chatDescription	String	Опис чату (тільки для групових чатів)
avatar	String	Посилання на фотографію чату (тільки для групових чатів)
isGroupChat	Boolean	Чи даний чат є групою
users	[User]	Учасники чату
latestMessage	Message	Останнє повідомлення в данім чаті
groupAdmins	[User]	Адміністратори чату (тільки для групових чатів)
updatedAt	Date	Дата і час останнього оновлення даних чату
createdAt	Date	Дата і час створення чату

3. Опис таблиці Message (табл. 2.15).

Таблиця 2.15

Опис таблиці Message

Поле	Тип	Опис
id	String	Унікальний ідентифікатор повідомлення (uuid)

sender	User	Користувач, що вислав повідомлення
text	String	Текст повідомлення
attachments	[Object]	Додатки до повідомлення (фотографії, документи)
chat	Chat	Чат до якого належить повідомлення
status	String	Статус повідомлення (Вислано/Доставлено/Прочитано)
updatedAt	Date	Дата і час останнього оновлення повідомлення
createdAt	Date	Дата і час створення повідомлення

4. Опис таблиці Notification (табл. 2.16).

Таблиця 2.16

Опис таблиці Notification

Поле	Тип	Опис
id	String	Унікальний ідентифікатор сповіщення (uuid)
message	Message	Повідомлення о яке треба сповістити
user	User	Хто вислав те повідомлення
chat	Chat	З якого чату було вислано повідомлення
updatedAt	Date	Дата і час останнього оновлення сповіщення
createdAt	Date	Дата і час створення сповіщення

Набір функцій MongoDB відіграв ключову роль у формуванні нашого чат-додатку. Її гнучкість у структурі документів дозволила динамічно адаптуватися до мінливих потреб нашої платформи. Вбудовані об'єкти, такі як «sender», «chat» і «user», посилюють зв'язки між документами і спрощують роботу з ними. Використання посилань на інші колекції за допомогою поля "ref" дозволило нам зберігати зв'язки між різними типами даних і забезпечила узгодженість даних. Ми використовували enum для поля "status" моделі повідомлень і користувачів, щоб обмежити доступні значення і забезпечити

контроль над даними. Більше того, автоматичне оновлення значень дат, через поле «timestamps», спростило відстеження часу створення та зміни даних..

2.3 Проектування та реалізація алгоритмів роботи системи

Розробка та реалізація системних алгоритмів відповідають за виконання різних обчислювальних завдань, обробку даних і прийняття рішень в системі. Ми створили декілька діаграм активності і детально їх описали.

Алгоритм реєстрації/автентифікації (рис. 2.4).

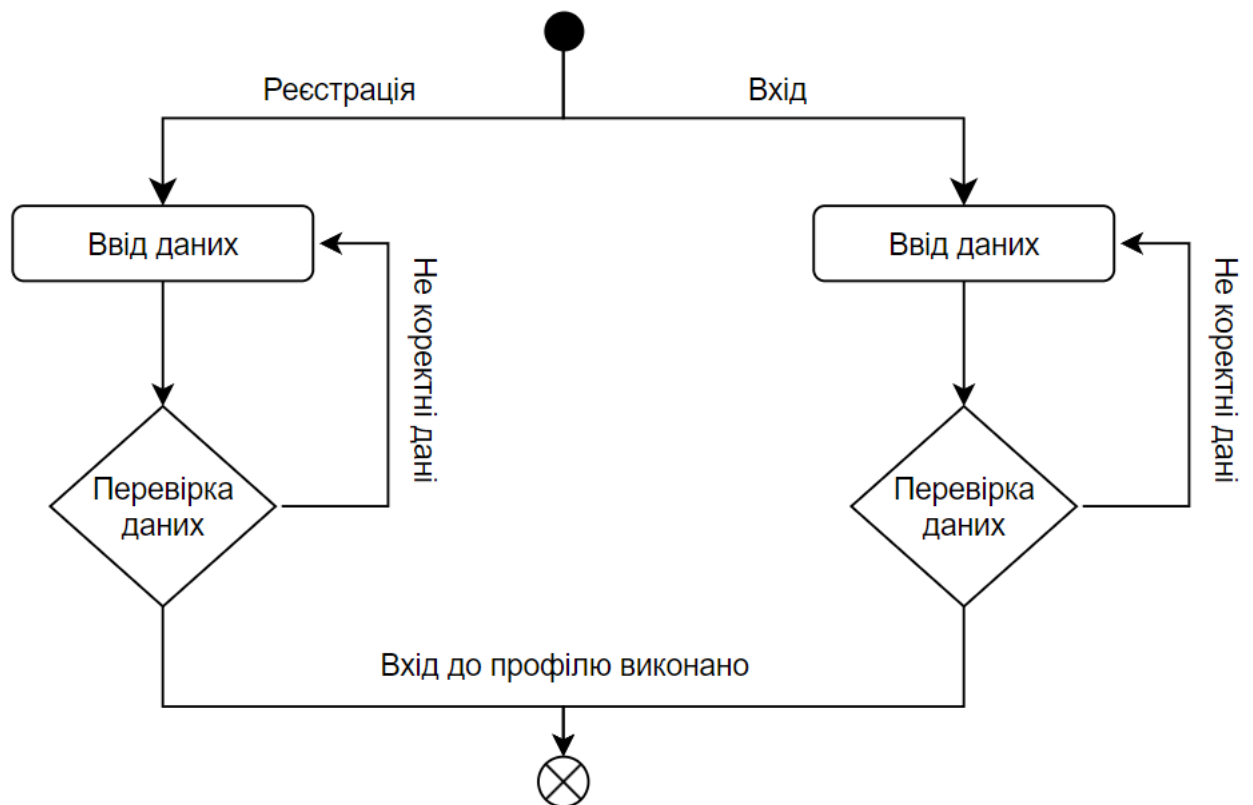


Рис. 2.4 Діаграма активності для реєстрації/автентифікації

Реєстрація нових користувачів: Під час реєстрації користувач вводить свої особисті дані, такі як ім'я, прізвище, електронна пошта та пароль. Система перевіряє унікальність електронної пошти та зберігає дані нового користувача у базі даних.

Автентифікація: При спробі входу в систему користувач вводить свою електронну пошту та пароль. Система перевіряє ці дані та надає доступ до особистого облікового запису у разі їх вірності.

Алгоритм управління профілем (рис. 2.5).

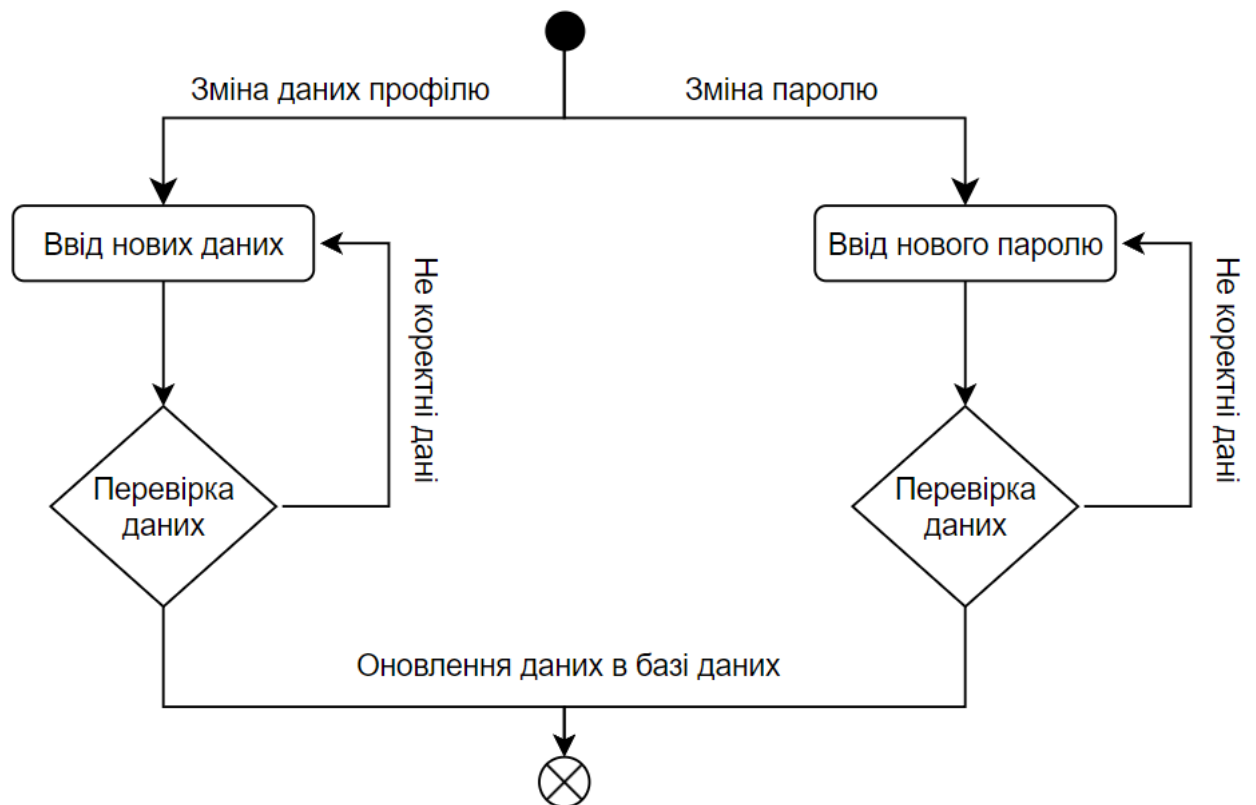


Рис. 2.5 Діаграма активності для управління профілем користувача

Зміна особистих даних: Користувач змінює свої особисті дані, такі як аватар, ім'я, прізвище і email. Система перевіряє ці дані та зберігає їх в базі даних.

Зміна паролю: Користувач вводить старий пароль і новий. Система перевіряє дані і оновлює пароль в базі даних.

Алгоритм обробки повідомлень (рис. 2.6).

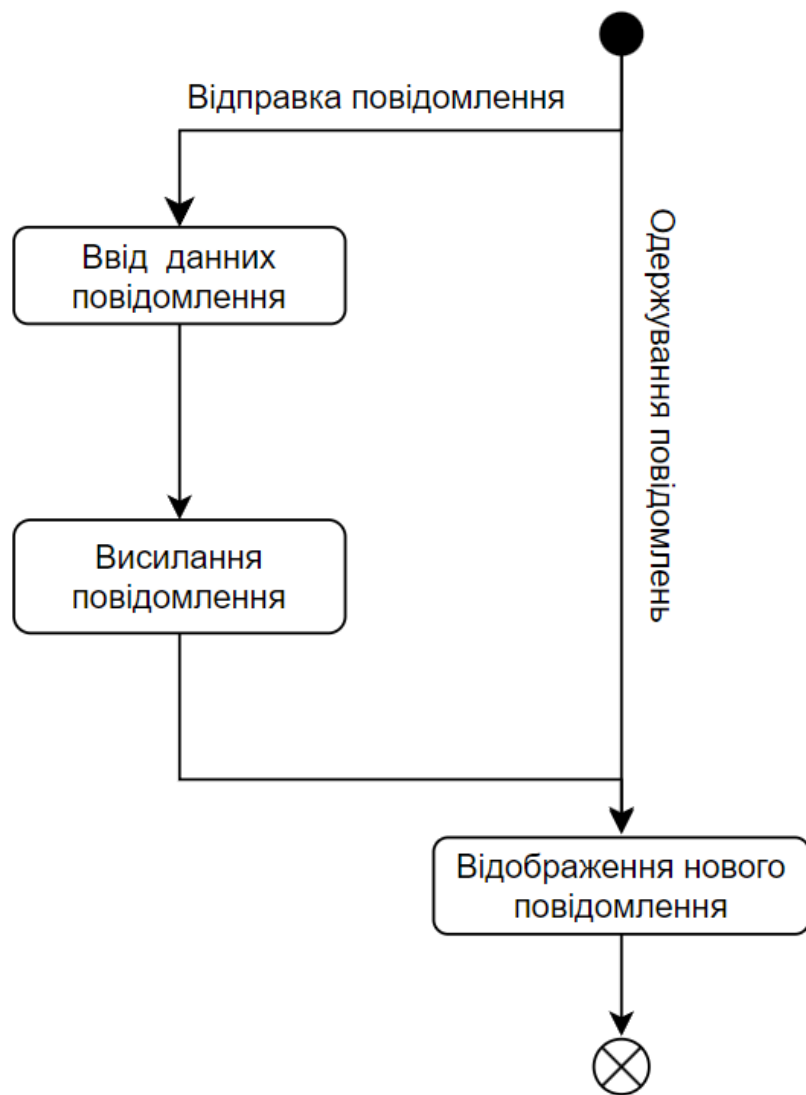


Рис. 2.6 Діаграма активності для обробки повідомлень

Відправлення повідомлень: Користувач може відправляти текстові повідомлення та прикріплювати до них файли. Сервер обробляє ці повідомлення та зберігає їх у базі даних.

Отримання повідомлень: Клієнтська частина програми періодично запитує сервер про нові повідомлення, які можуть бути відправлені користувачеві. Сервер відповідає на запит, надсилаючи дані про нові повідомлення.

Зберігання та відображення: Повідомлення зберігаються у базі даних та відображаються у відповідному чаті користувача.

Алгоритм управління чатами (рис. 2.7).

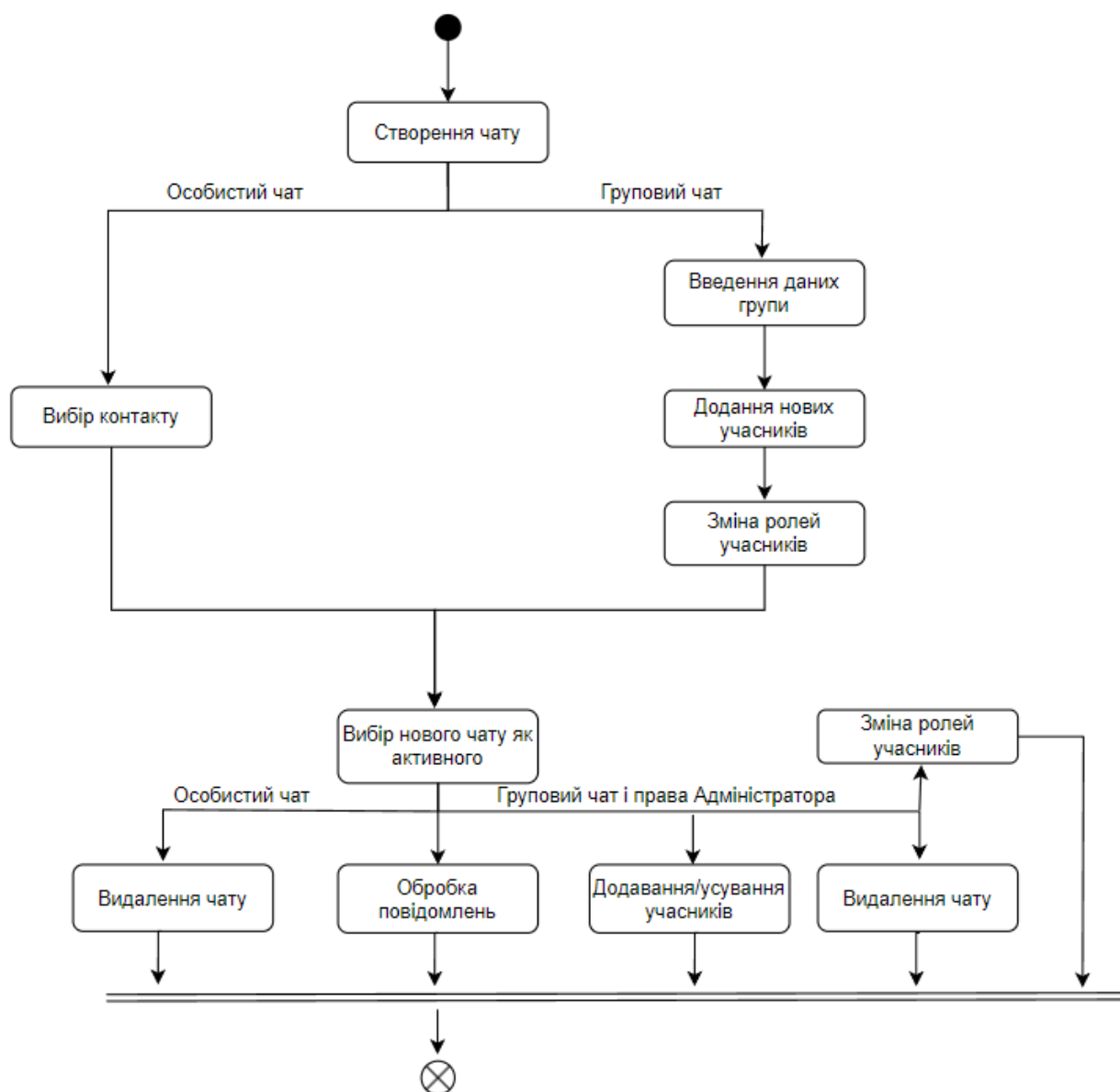


Рис. 2.7 Діаграма активності для управління чатами

Створення чатів: Користувач може створити новий особистий чат або груповий чат та додати до нього інших користувачів.

Додавання та видалення учасників: Адміністратор групового чату може додавати або видаляти учасників, а також змінювати їхні ролі у чаті.

Зміна назви та інших параметрів чату: Адміністратор може змінювати назву, опис та інші параметри групового чату.

Видалення чату: Адміністратор може видалити груповий чат. Також кожний учасник особистого чату може видалити його.

2.4 Реалізація функціоналу онлайн чату в реальному часі

2.4.1. Backend

Для розробки серверної частини чату [12] використано Express.js [13], який дозволяє швидко створити сервер та легко обробляти запити. Реалізацію серверної частини чату, базується на архітектурному патерні Model-View-Controller (MVC) [14]. Даний патерн допомагає розділити логіку програми на три основні частини: модель (Model), представлення (View) та контролер (Controller).

1. Модель (Model): Модель відповідає за структуру даних та логіку, пов'язану з базою даних. У коді наведено схеми моделей користувачів, повідомлень, чатів та сповіщень, які визначають поля та їхні типи. Наприклад, у схемі користувача визначено поля для збереження імені, електронної пошти, пароля тощо.
2. Представлення (View): У контексті серверної частини, представленням може вважатися обробка HTTP-запитів та відправка HTTP-відповідей. Використання роутінгу (`express.Router()`) дозволяє визначити різні шляхи (endpoints) та пов'язані з ними обробники запитів. Наприклад, `POST /user/sign-up` обробляє запит на реєстрацію нового користувача.
3. Контролер (Controller): Контролер відповідає за обробку запитів та взаємодію між моделлю та представленням. У коді використано контролери, які містять функції обробки запитів (наприклад, `signUp`, `signIn`, `sendMessage` тощо). Ці функції виконують логіку обробки даних (наприклад, перевірка наявності користувача в базі даних, збереження нового повідомлення тощо) та відправляють відповіді на запити.

В реалізації серверної частини чату було створено різні роутінги [15] для обробки HTTP-запитів. Ось опис кожного з них:

- **userRoutes.js:** Цей роутінг відповідає за обробку запитів, що стосуються користувачів. Він містить наступні шляхи:

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

- POST /user/sign-up: Обробляє запит на реєстрацію нового користувача.
- POST /user/sign-in: Обробляє запит на входження в систему.
- GET /user/me: Отримує інформацію про авторизованого користувача.
- PUT /user/me: Оновлює особисті дані користувача.
- PUT /user/password: Змінює пароль користувача.
- GET /user/:search: Пошук користувачів за певним критерієм.
- **messageRoutes.js:** Цей роутінг відповідає за обробку запитів, що стосуються повідомлень. Він містить наступні шляхи:
 - POST /message: Відправляє повідомлення у чат.
 - GET /message/notif: Отримує сповіщення про нові повідомлення.
 - GET /message/:chatId: Отримує всі повідомлення у певному чаті.
- **chatRoutes.js:** Цей роутінг відповідає за обробку запитів, що стосуються чатів. Він містить наступні шляхи:
 - POST /chat: Створює новий чат.
 - GET /chat: Отримує всі чати користувача.
 - DELETE /chat/:chatId: Видаляє чат за його ідентифікатором.
 - POST /chat/group: Створює новий груповий чат.
 - PUT /chat/group: Оновлює груповий чат.
 - PUT /chat/group/leave: Залишає груповий чат.

Також був створений роутінг для POST /upload: Цей роут дозволяє клієнтам завантажувати файли на сервер, а також отримувати доступ до завантажених файлів через їх URL. Давайте розглянемо його детальніше:

1. Multer [16] Configuration: Використовується пакет Multer для обробки завантажених файлів. Спочатку налаштовується об'єкт storage, що вказує, куди зберігати завантажені файли і як їх називати. У цьому випадку файли зберігаються у папці "uploads", а їх ім'ям є поточна дата з розширенням оригінального імені файлу.

2. Middleware Configuration: Додано middleware для обробки JSON-даних та CORS-запитів [17]. Це дозволяє серверу приймати запити від клієнта і обробляти їх правильно.

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				38
Змн.	Арк.	№ докум.	Підпис	Дата		

3. POST /upload Route: За допомогою методу `app.post`, створюється роут `"/upload"`, який очікує POST-запити для завантаження файлів. Обробка завантаження файлів відбувається за допомогою `Multer middleware upload.single("file")`, що приймає файл з ім'ям `"file"`. Після успішного завантаження файлу, сервер відправляє відповідь з даними про завантажений файл, включаючи його оригінальне ім'я, назву, URL та розмір.

4. Static File Serving: За допомогою `express.static`, налаштовується `middleware` для обслуговування статичних файлів з папки `"uploads"`. Це дозволяє клієнту звертатися безпосередньо до завантажених файлів за їх URL.

У контролерах для кожного роуту реалізовані відповідні функції, які обробляють запити та взаємодіюють з базою даних. Давайте розглянемо кожен з них більш детально:

- **UserController.js:** відповідає за обробку різних запитів, пов'язаних з користувачами системи [18]. Розглянемо кожен з них:

- **signUp:** Ця функція призначена для реєстрації нових користувачів. Вона перевіряє, чи введені всі необхідні дані, чи користувач з таким email вже існує в базі даних, хешує пароль [19], створює нового користувача в базі даних та надсилає йому токен доступу.

- **signIn:** Ця функція відповідає за автентифікацію користувача. Вона перевіряє введений email та пароль з даними у базі даних, і якщо вони співпадають, вона надає користувачу токен доступу.

- **search:** Ця функція використовується для пошуку користувачів за їхнім ім'ям, прізвищем або email. Вона використовує регулярні вирази для пошуку в базі даних та повертає результати.

- **getMe:** Ця функція повертає дані про поточного користувача, які були додані до запиту в `middleware`.

- **updateMe:** Ця функція оновлює дані користувача. Вона приймає нові дані про користувача, оновлює їх у базі даних та повертає оновлені дані та новий токен доступу.

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

– **changePassword:** Ця функція призначена для зміни паролю користувача. Вона перевіряє, чи вірний старий пароль, хешує новий пароль та зберігає його у базі даних.

– **updateUserStatus:** Ця функція оновлює статус користувача в базі даних (online або offline) та оновлює дату останньої активності.

- **messageController.js:** Цей контролер відповідає за обробку запитів, пов'язаних з повідомленнями. Давайте розглянемо кожен з функцій:

– **sendMessage:** Ця функція викликається при надсиланні нового повідомлення у чат. Вона перевіряє, чи відправлені дані коректні, створює нове повідомлення в базі даних, оновлює останнє повідомлення у чаті, створює сповіщення для кожного користувача чату, крім автора повідомлення, і повертає нове повідомлення у відповідь на запит.

– **updateMessage:** Ця функція оновлює вже існуюче повідомлення за його ID. Вона перевіряє, чи відправлені дані коректні, оновлює вміст та вкладення повідомлення в базі даних та повертає оновлене повідомлення.

– **updateMessageStatus:** Ця функція оновлює статус повідомлення (наприклад, "sent" або "read"). Вона оновлює статус повідомлення в базі даних та, якщо це повідомлення помічається як "read", видаляє відповідне сповіщення про це повідомлення для користувача.

– **deleteMessage:** Ця функція видаляє повідомлення за його ID з бази даних.

– **getAllMessages:** Ця функція повертає всі повідомлення для певного чату за його ID.

– **getNotificationMessages:** Ця функція повертає всі повідомлення, для яких у користувача є сповіщення.

- **chatController.js:** Цей контролер містить функції, які відповідають за обробку різних запитів, пов'язаних з чатами:

– **createChat:** Ця функція створює новий приватний чат між двома користувачами. Вона перевіряє, чи передані дані коректні, перевіряє наявність

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

існуючого чату між цими користувачами, і якщо він не існує, створює новий чат.

- **getMyChats:** Ця функція отримує всі чати, в яких користувач бере участь. Вона вибирає чати, в яких користувач є учасником або адміністратором, вибирає останнє повідомлення у кожному чаті, та сортує їх за датою останнього повідомлення.

- **createGroupChat:** Ця функція створює новий груповий чат. Вона перевіряє, чи передані дані коректні, і створює новий чат з вказаними параметрами.

- **updateGroupChat:** Ця функція оновлює інформацію про груповий чат. Вона оновлює назву, опис, аватар, список користувачів та список адміністраторів групи.

- **leaveGroupChat:** Ця функція видаляє користувача з групового чату. Вона перевіряє, чи користувач бере участь у чаті, та видаляє його як учасника чату.

- **deleteChat:** Ця функція видаляє чат за його ID. Вона перевіряє, чи користувач має дозвіл на видалення чату, та видаляє його з бази даних.

Ці контролери забезпечують повний функціонал для взаємодії з користувачами, повідомленнями та чатами у системі.

Крім того, у коді, був створений middleware **'protect'**, який використовується для захисту захищених маршрутів від неавторизованого доступу. Давайте розглянемо детальніше, як цей middleware працює:

1. Перевірка заголовка авторизації: Спочатку middleware перевіряє наявність та формат заголовка авторизації. Заголовок повинен починатися зі слова "Bearer", після чого йде пробіл та токен доступу.

2. Розбиття токена: Якщо заголовок авторизації існує і містить токен доступу, middleware розбиває токен на дві частини: "Bearer" та сам токен.

3. Перевірка наявності токена: Після розбиття токена, middleware перевіряє, чи був успішно розбитий токен. Якщо токен відсутній, middleware відправляє відповідь з кодом статусу 401 (Unauthorized).

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

4. Перевірка валідності токена: Якщо токен був успішно розбитий, `middleware` використовує бібліотеку `jsonwebtoken` для перевірки валідності токена. Якщо токен невірний або прострочений, `middleware` також відправляє відповідь з кодом статусу 401.

5. Додавання користувача до запиту: Якщо токен валідний, `middleware` декодує дані з токена та додає об'єкт користувача до запиту (`req`). Це дозволяє наступним обробникам запитів отримати доступ до ідентифікованого користувача.

6. Продовження виконання: Після успішної перевірки токена та додавання користувача до запиту, керування передається наступному `middleware` або обробнику запиту за допомогою функції `next()`.

7. Відправлення помилки: Якщо будь-який з перевірок не пройде (наприклад, відсутній токен або токен невірний), `middleware` відправляє відповідь з кодом статусу 401, що позначає відмову в авторизації.

Цей `middleware` додає безпеку до захищених маршрутів, перевіряючи правильність та валідність токена доступу перед наданням доступу до ресурсів.

2.4.2. Frontend

Наш фронтенд - це клієнтська частина нашого програмного забезпечення, яка відповідає за інтерфейс, який користувач бачить та взаємодіє з ним.

Підхід. Наш підхід в створенні компонентів та реалізації фронтенду базується на багатьох сучасних практиках, серед яких варто відзначити наступні особливості:

– Використання функціональних компонентів React [20]: Ми в основному використовуємо функціональні компоненти React замість класових. Це дозволяє нам писати більш зрозумілий, компактний та лаконічний код. Крім того, з введенням хуків у React, функціональні компоненти стали ще потужнішими та зручнішими для використання.

– Використання хуків: Ми активно використовуємо хуки React, такі як `useState`, `useEffect`, `useContext` та інші, для керування станом компонентів,

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

виконання побічних ефектів та реалізації логіки компонентів. Хуки дозволяють нам писати чистий та легко зрозумілий код.

- Використання контексту React [21]: Ми використовуємо контекст React для передачі даних через дерево компонентів без необхідності передачі пропсів через кожний проміжний компонент. Це дозволяє нам підтримувати чистоту коду та полегшує роботу зі станом додатку та глобальними даними.

- Використання стилів MUI та Emotion [22]: Ми використовуємо бібліотеку Material-UI (MUI) для готових UI компонентів та бібліотеку Emotion для стилізації компонентів. Material-UI надає широкий набір готових компонентів згідно з дизайном Material Design, тоді як Emotion дозволяє легко використовувати CSS-in-JS підходи для стилізації компонентів.

- Рефакторинг та оптимізація коду: Ми постійно працюємо над рефакторингом коду та оптимізацією шляхом вилучення зайвих або повторюваних елементів, покращення структури проекту та вдосконалення ефективності коду.

- Реактивний підхід: Наш підхід до реалізації фронтенду полягає в тому, щоб зробити додаток якнайбільш реактивним, тобто швидким та чуйним на дії користувачів. Ми стежимо за тим, щоб наші компоненти були оптимізовані та працювали ефективно навіть при великій кількості даних чи складних операціях.

Загалом, наш підхід до розробки фронтенду ґрунтується на використанні сучасних технологій та практик, які дозволяють нам писати чистий, ефективний та легко збережений код, а також створювати користувацький інтерфейс, який відповідає найвищим стандартам зручності.

Providers. Ми маємо кілька провайдерів [23], які допомагають у керуванні різними аспектами додатку. Ось детальніше про це:

AuthProvider – це контекстний провайдер, який надає всю необхідну функціональність для автентифікації та авторизації користувачів у нашому додатку.

Функції та стан:

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

- user: стан, який містить дані про поточного користувача.
- token: стан, який зберігає токен доступу користувача.
- isAuthenticated: стан, який вказує, чи автентифікований користувач.
- login(newToken): функція для входу користувача з вказаним токеном.
- logout(): функція для виходу користувача та очищення станів.
- fetchUserData(): функція для отримання даних користувача з сервера на основі токена.
- useAuth(): кастомний хук, який дозволяє отримати доступ до даних та функцій AuthProvider.

AuthProvider обгортає всі компоненти додатку, що потребують доступ до автентифікованого користувача або його даних.

SocketProvider – це провайдер для роботи з сокетом, який надає з'єднання з сервером через WebSocket та обробляє повідомлення в реальному часі.

Функції та стан:

- socket: стан, який містить об'єкт сокета для з'єднання з сервером.
- socketConnected: стан, який показує, чи встановлено з'єднання з сервером.
- usersStatuses: стан, який містить дані про статуси користувачів (онлайн, офлайн тощо).
- notification: стан, який містить сповіщення про нові повідомлення для користувача.
- isUserOnline(userId): функція, яка перевіряє, чи користувач з вказаним ID є онлайн.
- getUserStatus(userId): функція, яка повертає статус користувача (онлайн, час останньої активності тощо).
- useSocket(): кастомний хук, який дозволяє отримати доступ до даних та функцій SocketProvider.

SocketProvider обгортає всі компоненти додатку, які використовують сокети для отримання або надсилання даних в реальному часі, такі як чати або сповіщення.

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

ThemeProvider – це компонент з бібліотеки Material-UI, який надає можливість використовувати теми (themes) в додатку, щоб стилізувати його компоненти за допомогою готових стилів або власних персоналізованих стилів.

Ці провайдери допомагають у створенні потужного та гнучкого фронтенду, який може відповідати потребам наших користувачів та бізнесу. Вони забезпечують зручний спосіб організації та управління функціональністю додатку, що робить розробку більш ефективною та швидкою.

Маршрутизація – це процес визначення, який компонент React має бути відображений на сторінці в залежності від поточного URL. У React цю задачу виконує бібліотека react-router-dom [24], яка надає набір компонентів для організації маршрутів у веб-додатках.

У нашому коді використовується компонент `<Routes>` для визначення маршрутів. Кожен маршрут визначається за допомогою компонента `<Route>`, де вказується шлях (path) і компонент, який буде відображений для цього маршруту.

PublicRoute – це компонент використовується для маршрутів, до яких може отримати доступ будь-хто, незалежно від авторизації. У компоненті PublicRoute перевіряється, чи користувач автентифікований за допомогою `useAuth()` з `AuthContext`. Якщо користувач вже автентифікований, то він перенаправляється на головну сторінку за допомогою `<Navigate to="/" replace />`. В іншому випадку, компонент, який відповідає поточному маршруту, буде відображений.

PrivateRoute. Цей компонент використовується для захищених маршрутів, до яких можуть отримати доступ лише автентифіковані користувачі. У компоненті PrivateRoute також використовується `useAuth()` для перевірки автентифікації користувача. Якщо користувач автентифікований, то відображається компонент `<SocketProvider>`, який забезпечує доступ до контексту сокетів. В іншому випадку, користувач перенаправляється на сторінку входу за допомогою `<Navigate to="/sign-in" />`.

		Іщенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

Forms. У нашому чаті всі форми будуть використовувати бібліотеку react-hook-form [25] для управління даними введеними користувачем. Ось деякі особливості та причини використання цієї бібліотеки:

1. Простота використання: react-hook-form має зрозумілий та легкий у використанні API, що дозволяє швидко розробляти та налаштовувати форми без зайвого коду.

2. Висока продуктивність: Бібліотека оптимізована для забезпечення високої продуктивності та ефективної роботи навіть з великими формами та великим обсягом даних.

3. Валідація в реальному часі: react-hook-form надає можливість виконувати валідацію введених даних в реальному часі, що дозволяє користувачам отримувати миттєві повідомлення про помилки.

4. Контроль над полями форми: Бібліотека дозволяє точно контролювати кожне поле вводу окремо, що дозволяє налаштовувати логіку обробки даних та валідації для кожного поля.

5. Інтеграція з іншими бібліотеками: react-hook-form легко інтегрується з різними бібліотеками компонентів, такими як Material-UI, що дозволяє використовувати готові компоненти для створення форм.

Використання react-hook-form в нашому чаті дозволить створювати зручні та ефективні форми, які працюватимуть швидко та надійно, що є ключовим аспектом для забезпечення хорошого користувацького досвіду.

Стилізація. В нашому компоненті ми використовуємо styled-components [26] для стилізації деяких елементів. Ось деякі особливості цього підходу та причини його використання:

– Гнучкість: За допомогою styled-components ми можемо створювати стилізовані компоненти, які можна перевикористовувати в інших частинах програми. Це дозволяє забезпечити гнучкість та легкість управління стилями.

– Локальність стилів: Кожен стилізований компонент має свій власний область видимості, що дозволяє уникнути конфліктів імен класів та забезпечити чистоту коду.

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

– Можливості JavaScript: Використання JavaScript в стилях дозволяє динамічно змінювати стилі в залежності від умов або властивостей компонентів.

– Інтеграція з іншими бібліотеками: styled-components легко інтегрується з іншими бібліотеками компонентів, такими як Material-UI, що дозволяє стилізувати готові компоненти зручним для розробника способом.

– Підтримка CSS-in-JS: styled-components надає можливість використання всіх переваг CSS-in-JS, таких як збільшення продуктивності, зручність управління стилями та інші.

Ми задаємо стилі прямо в коді компоненту, що робить їх легкими для розуміння та збереження в контексті компонента. Цей підхід дозволяє нам створювати красиві та функціональні інтерфейси для нашого додатку.

На додаток до styled-components, у нашому компоненті також використовуються інші методи стилізації, такі як пряме застосування стилів через sx та використання вбудованих стилів з бібліотеки Material-UI.

– Пряме застосування стилів через sx: Ми використовуємо атрибут sx для прямого задання стилів до окремих компонентів. Цей підхід дозволяє нам швидко та зручно застосовувати стилі без необхідності створювати окремі стилізовані компоненти.

– Використання вбудованих стилів з Material-UI: Ми також використовуємо стилізовані компоненти з Material-UI, які вже мають попередньо визначені стилі. Ці компоненти мають широкий набір можливостей для стилізації, таких як TextField, AppBar, Toolbar, Typography, тощо. Ми можемо задавати деякі стилі безпосередньо через пропси цих компонентів.

Цей підхід дозволяє нам швидко та зручно використовувати готові компоненти з Material-UI, не втрачаючи можливості керувати їхнім виглядом та поведінкою.

Обидва ці методи доповнюють використання styled-components і дають нам можливість створювати стильні інтерфейси з урахуванням наших потреб та вимог дизайну.

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

Запити. Ми використовуємо бібліотеку axios [27] для виконання HTTP-запитів у нашому додатку з наступних причин:

- Простота використання: axios надає зручний та простий інтерфейс для виконання HTTP-запитів. Він пропонує послідовні методи для кожного типу запиту (GET, POST, PUT, DELETE), що робить його використання зрозумілим та простим для розробників.
- Підтримка промісів: axios повертає проміси, що дозволяє зручно обробляти результати запитів за допомогою async/await або методів then/catch.
- Підтримка заголовків: Ми можемо легко встановлювати заголовки запиту, такі як авторизаційні токени, для взаємодії з захищеними API.
- Інтерцептори: axios дозволяє використовувати інтерцептори для глобальної обробки запитів та відповідей. Це робить його дуже потужним для виконання різних завдань, таких як авторизація, обробка помилок, тощо.
- Підтримка серверного та клієнтського коду: axios може використовуватися як на стороні сервера (Node.js) так і на стороні клієнта (браузер або React Native). Це дозволяє нам легко використовувати одну бібліотеку для виконання запитів у всьому додатку.
- Добре документовано: axios має докладну документацію та широку базу користувачів, що робить навчання та використання його дуже зручними.

Загалом, axios є потужним та зручним інструментом для виконання HTTP-запитів у нашому додатку, і ми обрали його через його простоту використання, гнучкість та потужність.

Підсумування. Наш фронтенд - це динамічний інтерфейс користувача для нашого додатку, який взаємодіє з користувачем через веб-браузер. Він розроблений з використанням сучасних технологій, зокрема React.js, який дозволяє нам створювати ефективні, масштабовані та легко збережені веб-додатки.

Наш фронтенд складається з різних компонентів, які взаємодіють між собою для створення багатофункціонального інтерфейсу. Ми використовуємо

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

функціональні компоненти React, оскільки вони надають більшу простоту та зручність у порівнянні з класовими компонентами.

Для стилізації наших компонентів ми використовуємо Material-UI (MUI) для готових UI компонентів, які відповідають дизайну Material Design. Крім того, ми також використовуємо Emotion для створення CSS-in-JS, що дозволяє нам зберігати стилі разом з компонентами та забезпечує більшу гнучкість у роботі зі стилями.

Наш фронтенд використовує різні підходи до управління станом, такі як використання хуків React для стану компонентів, контексту React для глобального стану та реактивного підходу до оновлення інтерфейсу користувача.

Загалом, наш фронтенд є потужним інструментом для створення зручного та ефективного інтерфейсу користувача, який надає користувачам можливість легко взаємодіяти з нашим додатком та виконувати різноманітні завдання.

2.4.3. Socket.io

Socket.IO - це бібліотека JavaScript, яка дозволяє реалізувати багатонаправлене з'єднання між клієнтом і сервером через веб-сокети. Вона забезпечує можливість миттєвої передачі даних в обидва напрямки і дозволяє створювати реально-часні додатки, такі як чати, ігри та інші веб-додатки, які потребують швидкої інтерактивності.

У нашому випадку ми використовуємо Socket.IO для реалізації реального часу в нашому додатку чату. На фронтенді ми підключаємося до сервера Socket.IO за допомогою бібліотеки socket.io-client [28] і створюємо з'єднання з сервером. Ми передаємо інформацію про користувача при налагодженні з'єднання, щоб сервер міг відстежувати статус користувача і надсилати повідомлення.

На сервері ми використовуємо бібліотеку socket.io для створення серверу WebSocket, який слухає підключення користувачів. Ми реєструємо обробники подій для різних типів подій, таких як підключення користувача, відправлення повідомлення, редагування повідомлення, видалення повідомлення та інші.

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

Кожен користувач при підключенні реєструється у відповідному кімнаті для обміну повідомленнями з іншими користувачами.

Щойно будь-яка подія відбувається на сервері, вона миттєво транслюється на клієнтський бік через WebSocket. Користувачі отримують оновлення в реальному часі, що дозволяє їм бачити нові повідомлення, зміни статусу та інші події негайно.

Ми відслідковуємо кілька різних подій за допомогою сокетів у нашому додатку чату. Ось список деяких з них:

1. **setup**: Коли користувач підключається до додатку, ми відправляємо подію "setup", щоб сервер зміг налагодити з'єднання для цього користувача. Це дозволяє серверу відстежувати статус та дії користувача.

2. **join**: Коли користувач приєднується до певної кімнати чату (наприклад, групового чату), ми відправляємо подію "join", щоб додати користувача до цієї кімнати на сервері.

3. **typing**: Коли користувач починає вводити текст у чаті, ми відправляємо подію "typing", щоб повідомити інших користувачів у чаті про те, що цей користувач пише.

4. **stopTyping**: Коли користувач припиняє вводити текст у чаті, ми відправляємо подію "stopTyping", щоб повідомити інших користувачів, що цей користувач припинив писати.

5. **send**: Коли користувач надсилає повідомлення у чат, ми відправляємо подію "send" на сервер. Сервер оброблює це повідомлення і ретранслює його всім користувачам у цьому чаті.

6. **editMessage**: Коли користувач редагує своє повідомлення, ми відправляємо подію "editMessage" на сервер разом із новим текстом або вкладеннями. Сервер оновлює повідомлення у своїй базі даних та ретранслює зміни усім користувачам у чаті.

7. **deleteMessage**: Коли користувач видаляє своє повідомлення, ми відправляємо подію "deleteMessage" на сервер разом із ідентифікатором

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

повідомлення, яке потрібно видалити. Сервер видаляє повідомлення з бази даних та ретранслює зміни усім користувачам у чаті.

8. **markAsSent**: Коли повідомлення надсилається успішно, ми відправляємо подію "markAsSent" на сервер разом із ідентифікатором повідомлення. Сервер оновлює статус повідомлення на "sent" та ретранслює зміни усім користувачам у чаті.

9. **markAsRead**: Коли користувач відкриває повідомлення і читає його, ми відправляємо подію "markAsRead" на сервер разом із ідентифікатором повідомлення та ідентифікатором користувача, який прочитав повідомлення. Сервер оновлює статус повідомлення на "read" та ретранслює зміни усім користувачам у чаті.

10. **disconnect**: Коли користувач відключається від сервера, ми відправляємо подію "disconnect" на сервер. Сервер оновлює статус користувача на "offline" та ретранслює зміни усім іншим користувачам, щоб вони бачили, що цей користувач більше не в мережі.

В цілому, Socket.IO дозволяє нам створювати ефективні та реалізовані в реальному часі додатки, які забезпечують високу швидкість інтерактивності для користувачів.

Висновки до розділу 2

У цьому розділі ми розглянули проектування та реалізацію онлайн чату в реальному часі. По-перше, ми розглянули варіанти використання та визначили функціональні вимоги до нашого чату, щоб забезпечити його ефективне використання користувачами.

Наступним кроком було створення об'єктно-орієнтованої структури нашого чату. Ми розглянули основні компоненти, такі як чати, повідомлення, користувачі, і визначили їхні взаємозв'язки та властивості.

Після цього ми зосередили увагу на розробці фронкенду та бекенду чату. На фронтенді ми використовували бібліотеку React та інші технології, щоб створити інтерактивний та зручний для користувача інтерфейс. Ми також

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

використовували бібліотеку Socket.IO для забезпечення реального часу спілкування та оновлення даних між користувачами.

На бекенді ми використовували Node.js та Express.js для створення серверної частини чату. Ми використовували сокети для забезпечення спілкування між клієнтом та сервером в реальному часі, а також для відстеження подій, таких як набір тексту, надсилання повідомлень, редагування та видалення повідомлень.

Завершальним етапом було тестування та впровадження нашого чату. Ми перевірили функціональність, продуктивність та надійність системи, а також забезпечили відповідність всім вимогам користувачів. Усі ці кроки дозволили нам створити комплексне рішення для спілкування в реальному часі, яке задовольняє потреби користувачів у зручному та ефективному чаті.

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС І ПОРЯДОК РОБОТИ З ОНЛАЙН ЧАТОМ В РЕАЛЬНОМУ ЧАСІ

3.1 Порядок встановлення та налаштування параметрів платформи

Порядок встановлення та налаштування параметрів платформи для нашої програми є наступним:

1. Інсталяція залежностей: Перш за все, потрібно встановити необхідне програмне забезпечення. Це включає встановлення середовища Node.js для серверної частини, а також пакетний менеджер npm або yarn. Для фронтенду можна використовувати будь-який пакетний менеджер, але зазвичай використовують npm або yarn.

2. Клонування репозиторію: Після інсталяції залежностей можна скопіювати репозиторій з вихідним кодом нашої програми на локальний комп'ютер.

3. Встановлення змінних середовища: Переконайтеся, що ви встановили всі необхідні змінні середовища для правильної роботи програми. Це може включати змінні середовища для доступу до бази даних, API-ключів, конфігураційних параметрів тощо.

4. Інсталяція залежностей проекту: Після клонування репозиторію перейдіть у його каталог і виконайте команду npm install або yarn install, щоб встановити всі залежності проекту.

5. Налаштування бази даних: Якщо програма використовує базу даних, переконайтеся, що ви налаштували з'єднання з базою даних та створили всі необхідні таблиці та схеми.

6. Конфігурація файлів середовища: Перевірте всі конфігураційні файли, такі як .env [29], щоб переконатися, що всі налаштування відповідають потребам вашого середовища.

7. Запуск сервера і фронтенду: Після налаштування всіх параметрів можна запуснути серверну частину програми за допомогою команди npm start або yarn start, а фронтенд - за допомогою відповідної команди для фронтенду.

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

8. Тестування та налагодження: Після запуску програми проведіть тестування, щоб переконатися, що все працює правильно. Виправте будь-які помилки та внесіть необхідні зміни для поліпшення функціональності та продуктивності.

9. Розгортання на сервері: Коли програма готова для використання, розгорніть її на сервері. Виберіть платформу розгортання, таку як AWS, Heroku, DigitalOcean тощо, і виконайте необхідні кроки для розгортання вашого додатку.

10. Моніторинг та підтримка: Після розгортання програми слід продовжувати відстежувати роботу системи, вирішувати будь-які проблеми, які виникають, і вносити необхідні зміни для покращення її функціональності та продуктивності.

Для опису архітектури платформи було створено діаграму розгортання (рис. 3.1).

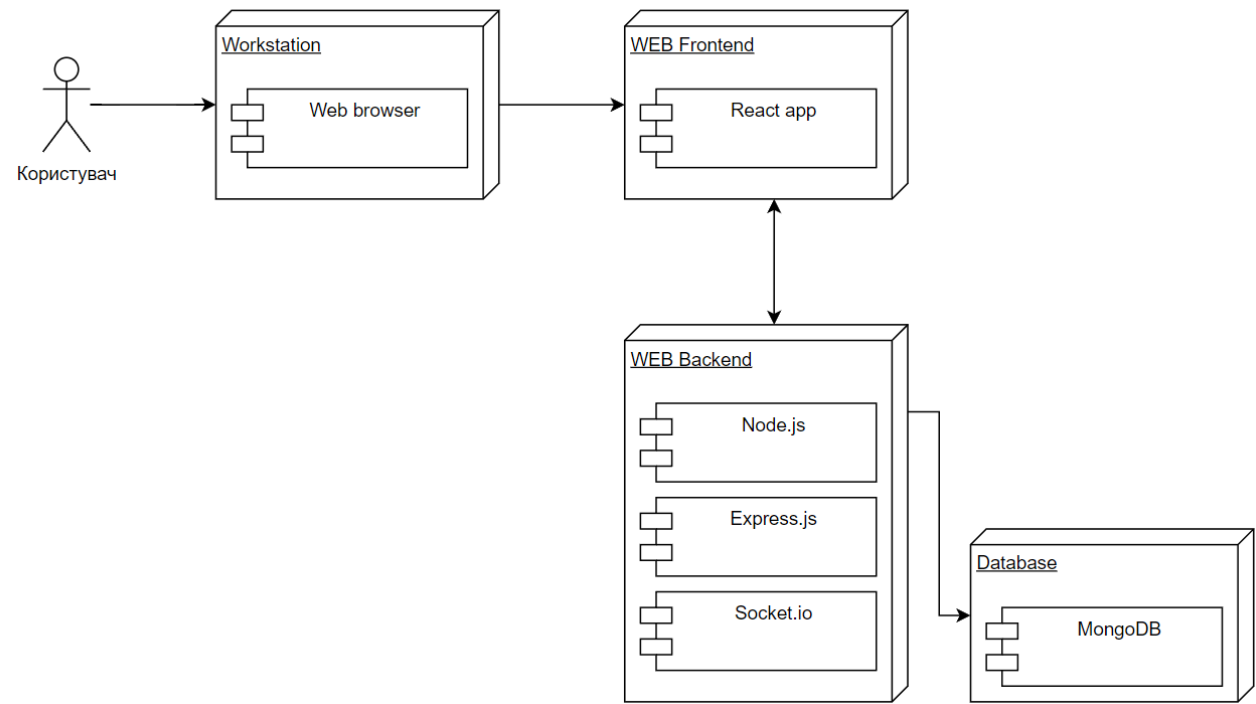


Рис. 3.1 Діаграма розгортання платформи

У цій діаграмі:

- Користувач: Це кінцевий користувач системи, який взаємодіє з нею через веб-браузер на своєму робочому місці.

- Workstation: Це робоче місце користувача, де він використовує веб-браузер для доступу до системи через інтерфейс користувача.
- WEB Frontend: Це фронтендова частина системи, реалізована за допомогою React, яка відображає інтерфейс користувача та обробляє його дії.
- WEB Backend: Це бекендова частина системи, яка відповідає за обробку запитів користувачів, забезпечення взаємодії між фронтендом та базою даних, а також реалізує взаємодію в реальному часі за допомогою Socket.io.
- Database: Це база даних MongoDB, яка використовується для зберігання інформації про користувачів, повідомлення та інші дані, необхідні для роботи системи.

Ця діаграма ілюструє шлях, яким даний проект розгортається в середовищі з кінцевим користувачем, веб-браузером, фронтендом, бекендом та базою даних.

3.2 Інтерфейс та порядок роботи

Перейдемо до огляду інтерфейсу та робочого процесу нашого онлайн чату. Ми розглянемо різні скріншоти і надамо короткі описи до кожного з них, підкреслюючи ключові особливості та функціональні можливості програмного продукту. Ці скріншоти демонструють елементи користувацького інтерфейсу, з якими користувачі взаємодіятимуть під час роботи.

Відкриття сторінки: При відкритті сторінки (перший раз) користувач потрапляє на сторінку входу (рис. 3.2).

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

Рис. 3.2 Сторінка входу

Якщо користувач має створений обліковий запис може вписати email та пароль і увійти до чату. Якщо не маємо то можемо створивши перейшовши на сторінку реєстрації (рис. 3.3).

Рис. 3.3 Сторінка реєстрації

Після входу в обліковий запис: Користувач потрапляє на сторінку чату (рис. 3.4).

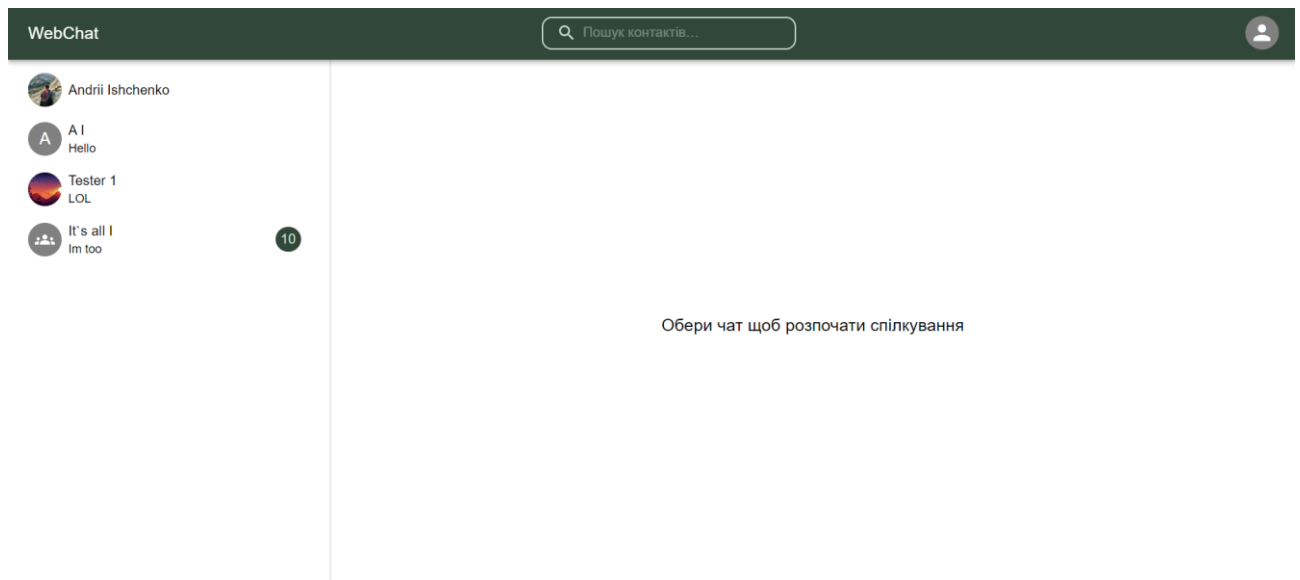


Рис. 3.4 Головна сторінка чату

Розглянемо основні компоненти:

Header: Знаходиться на горі сторінки. Містить назву нашого чату, поле для пошуку контактів і фотографія профілю при натисненні на яку відкриється меню. В тому меню можемо побачити детальнішу інформацію про нас, створити груповий чат, відкрити налаштування для зміни даних профілю і вийти з облікового запису (рис. 3.5).

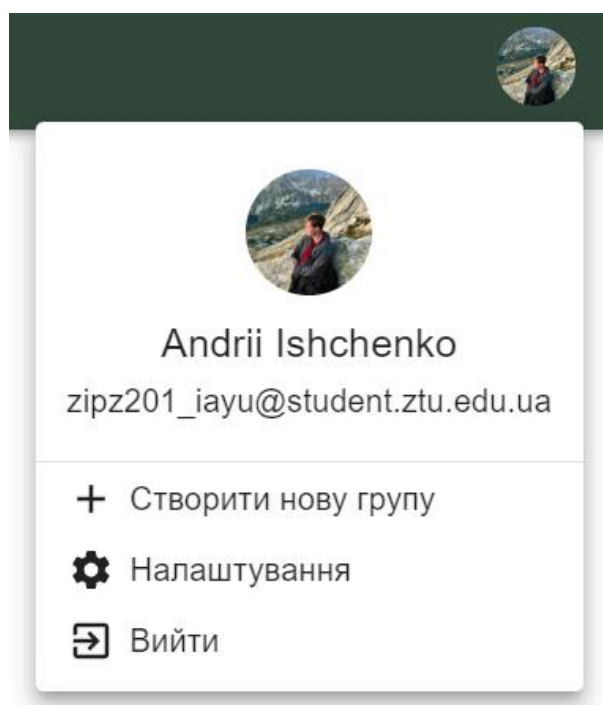


Рис. 3.5 Відкрите меню

Sidebar: Знаходиться по лівій стороні сторінки. В ньому знаходяться всі чати користувача. Натиснувши на якийсь з них він стане активний. Також якщо

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

користувач буде мати непрочитане повідомлення, то буде відображатися на нашому боковому меню (рис. 3.6).

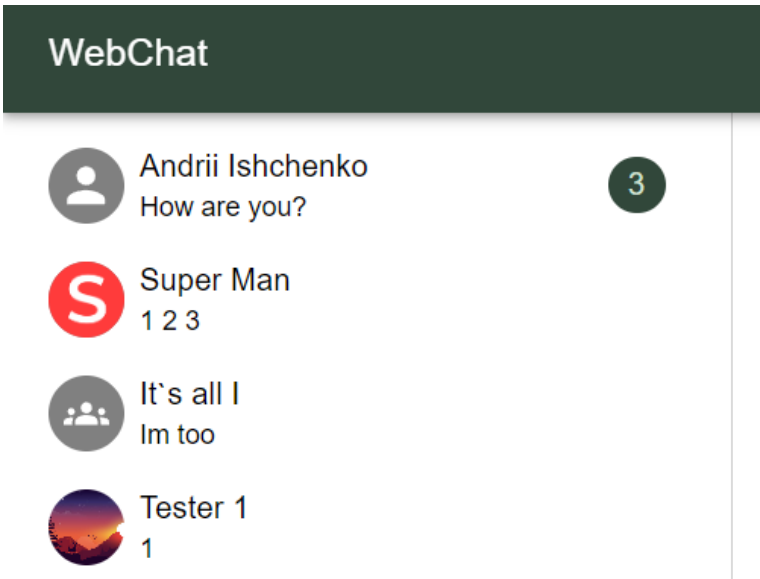


Рис. 3.6 Сповіщення про нові повідомлення в чаті

ChatBox: Основна частина нашого чату. Якщо не має активного чату, то буде відображатися повідомлення про це (див. рис. 3.4). Але якщо є вибраний то в ньому відображається вся історія повідомлень, інформація про чат і поле вводу повідомлень (рис. 3.7).

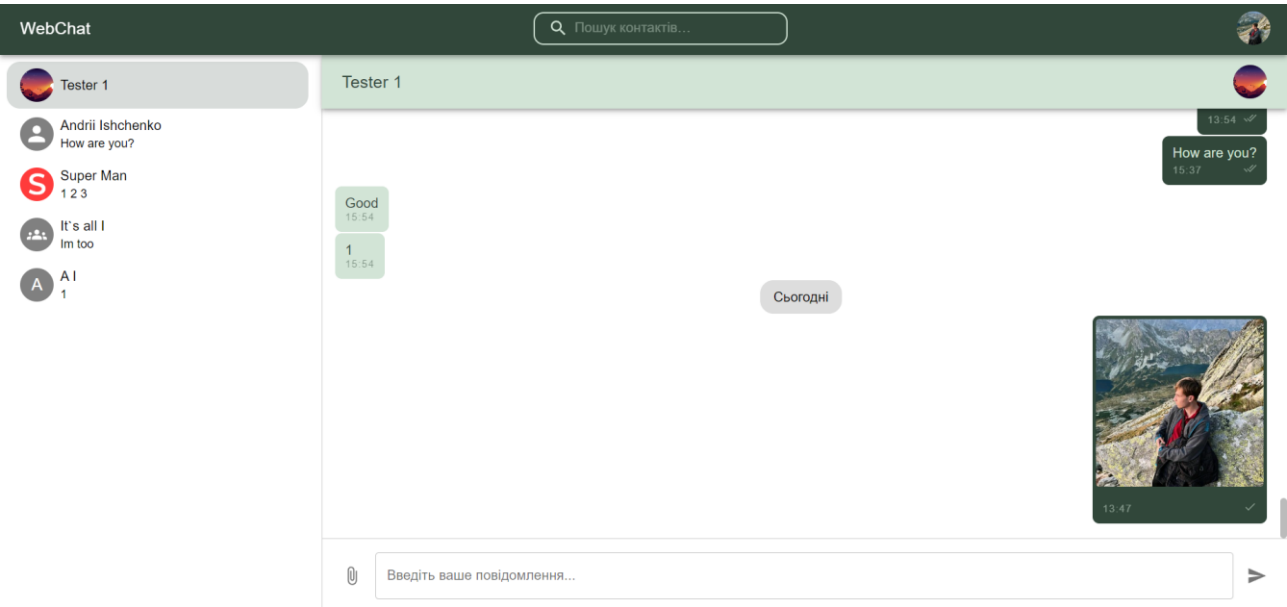


Рис. 3.7 Головна сторінка з вибраним чатом

Інформація про чат: На горі компоненту чату знаходиться по лівій стороні назва чату або ім'я і прізвище другого контакту в особистих чатах. По правій стороні фотографія профілю контакту в особистих чатах, а в групових

фотографія групового чату. При натисненні на неї відкриється модальне вікно в залежності від чату і ролі користувача в ньому. Модальне вікно учасника групового чату (рис. 3.8).

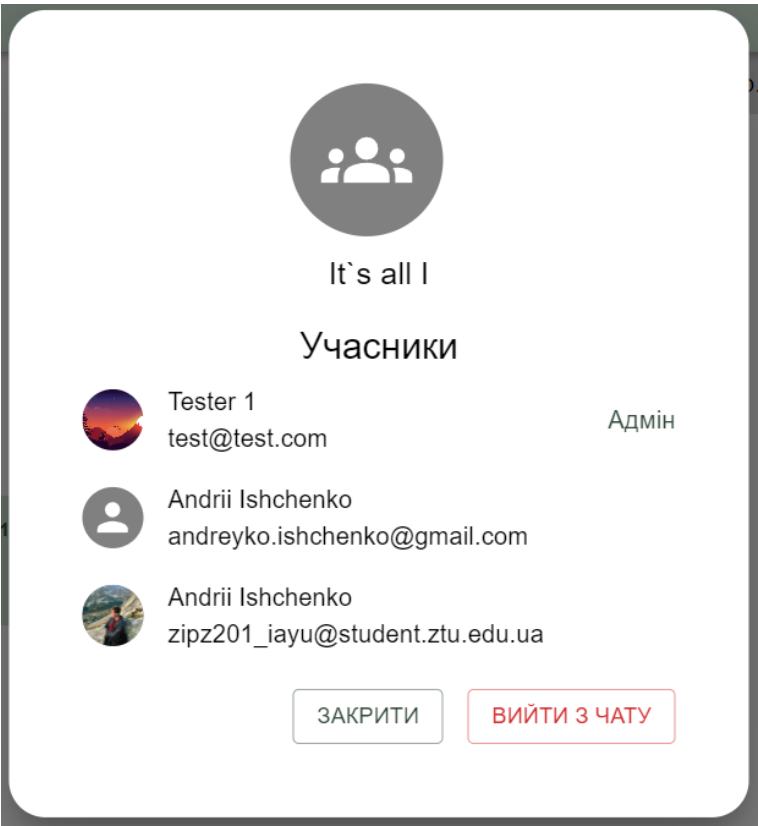


Рис. 3.8 Модальне вікно учасника групового чату
Модальне вікно адміністратора групового чату (рис. 3.9).

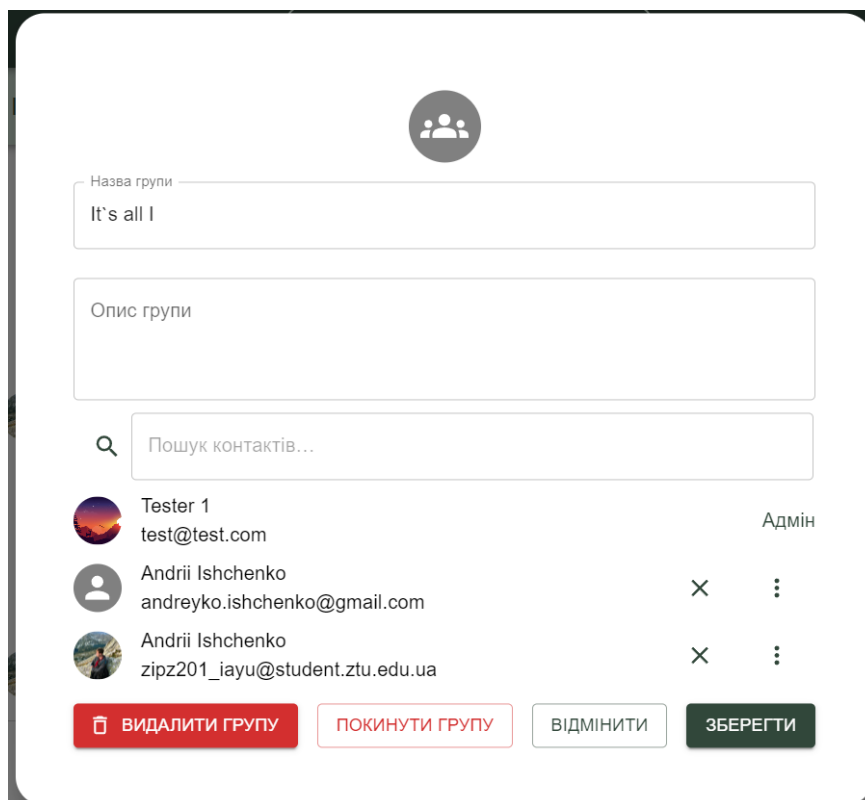


Рис. 3.9 Модальне вікно адміністратора групового чату

В цьому вікні можемо змінити назву, опис, фотографію групи. Додати нових учасників, змінити їх ролі (Натиснувши три крапки по правій стороні), видалити їх (Хрестик). Також можемо видалити або покинути чат.

Модальне вікно особистого чату (рис. 3.10).

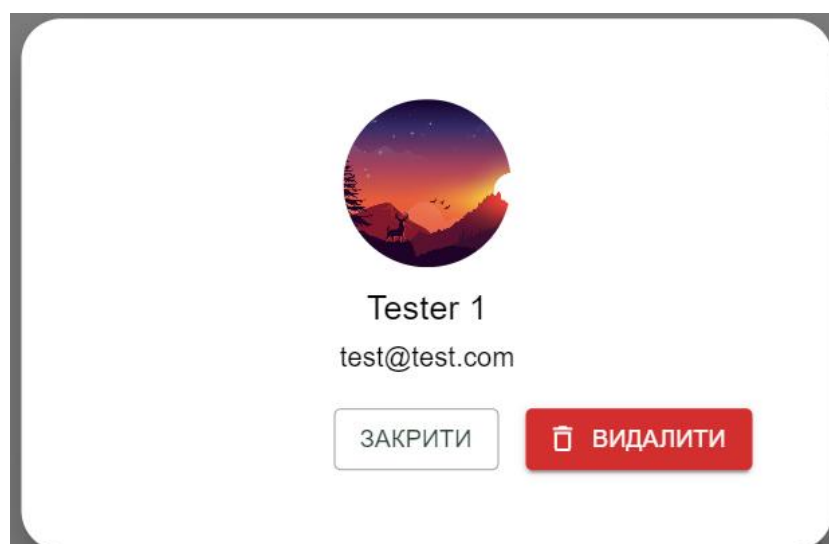


Рис. 3.10 Модальне вікно особистого чату

Історія повідомлень: В цьому компоненті відображаються повідомлення чату (і день в який вони були вислані). По лівій стороні знаходяться повідомлення інших користувачів, а по правій наші. Окрім цього повідомлення

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

в чатах містять в собі текст і/або файл, дату вислання і статус (годинник – висилається, одна галочка – доставлено, дві – прочитано). В групових чатах повідомлення інших користувачів містять також їх ім'я і фотографію профілю. По натисненню на наше повідомлення, відкриється меню за допомогою якого можемо змінити або видалити дане повідомлення (рис. 3.11).

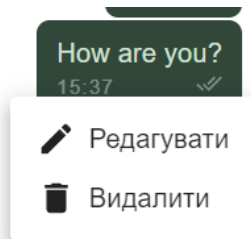


Рис. 3.11 Меню повідомлення

Поле вводу повідомлень: Знаходить внизу компоненту. Має три частини: 1. кнопка вставити файл, яка відкриває провідник з якого можна вибрати зображення, документ, відео, тощо. 2. Поле введення тексту. 3. Кнопка висилання повідомлення. Клавіша Enter теж висилає повідомлення. Крім цього цього вигляд трохи змінюється при едиції повідомлення. Над полем введення з'являється старе повідомлення і по правій стороні кнопка скасування. (рис. 3.12).

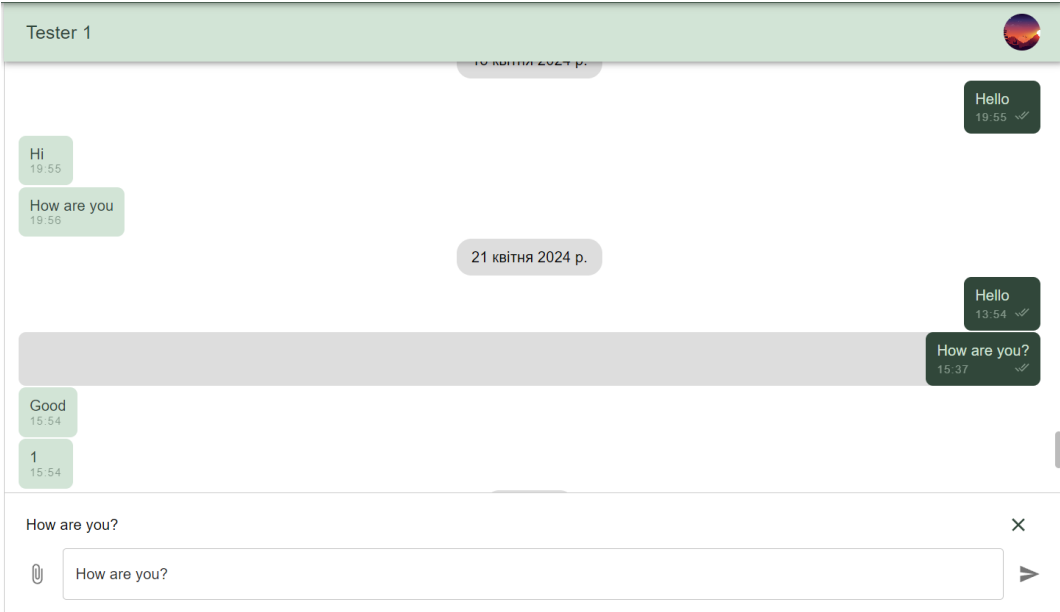


Рис. 3.12 Поле введення в режимі едиції

Створення нових чатів: Можемо створити нові особисті чати використовуючи поле пошуку в верхній частині сторінки. Достатньо лише почати вводити його ім'я прізвище або email (рис. 3.13).

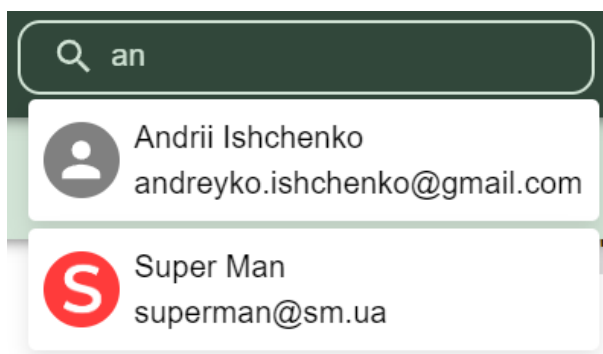


Рис. 3.13 Пошук і створення нового особистого чату

Щоб створити груповий чат треба в меню профілю (див. рис. 3.5) натиснути на “Створити новий груповий чат”. Після цього відкриється модальне вікно за допомогою якого можемо створити груповий чат (рис. 3.14).

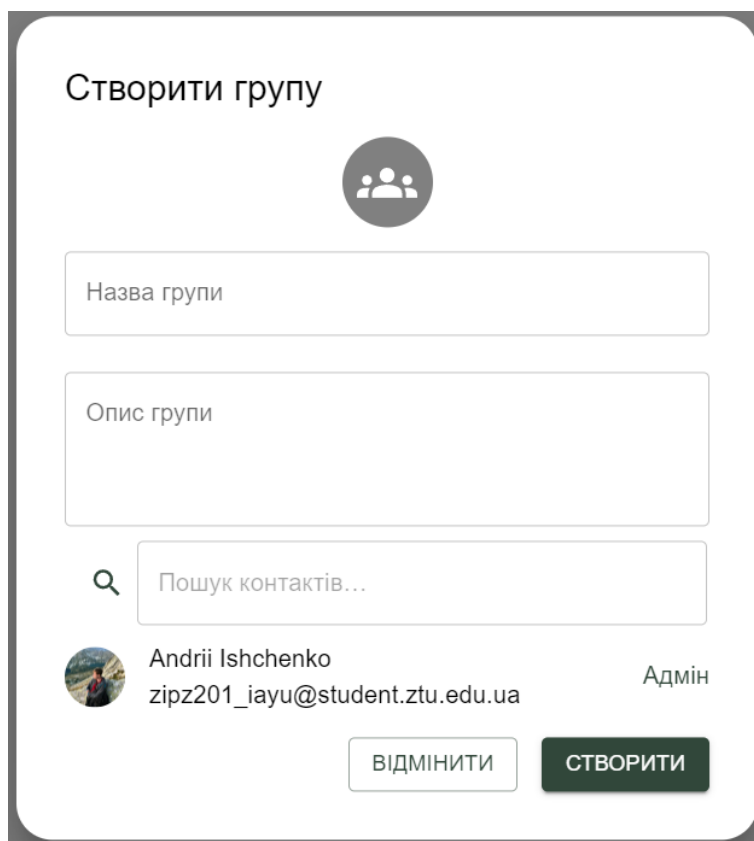


Рис. 3.14 Модальне вікно створення групового чату

Зміна даних профілю: Користувач може змінити свої дані натиснувши “Налаштування” в меню профілю (див. рис. 3.5). Після цього відкриється модальне вікно в якому можемо змінити інформацію (рис. 3.15).

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

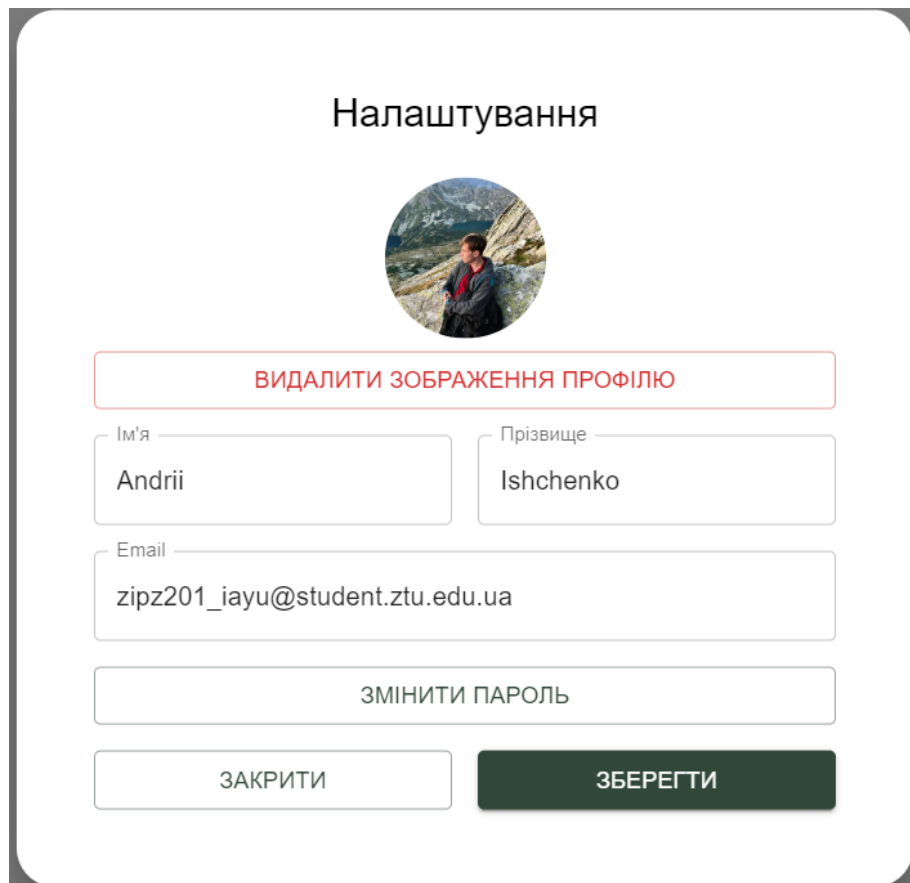


Рис. 3.15 Модальне вікно зміни особистих даних

Також можемо натиснути на кнопку зміни пароля, щоб змінити і його. Відкриється модальне вікно де треба ввести старий пароль і новий (рис. 3.16).

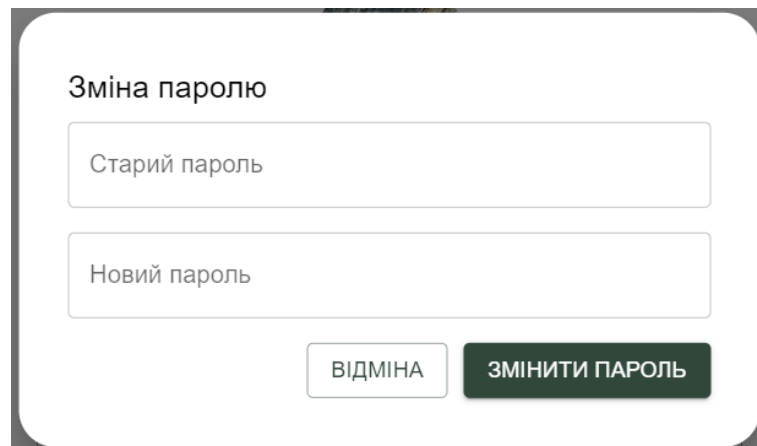


Рис. 3.16 Модальне вікно зміни паролю

У цьому підрозділі ми дослідили інтерфейс та робочий процес нашого онлайн чату в реальному часі. Ці візуальні матеріали дають уявлення про зручний та інтуїтивний дизайн, а також функціональні можливості, що пропонує наша система, надаючи користувачам зручні інструменти для ефективного спілкування.

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3 Тестування роботи онлайн чату

Тестування роботи нашого онлайн чату в реальному часі може включати кілька етапів, щоб перевірити різні аспекти його функціональності та продуктивності. Важливі аспекти, які потрібно врахувати під час тестування:

Функціональність інтерфейсу: тестування кожної функції окремо, включаючи надсилання повідомлень, створення нових чатів, додавання користувачів тощо. Перевірка можливості зміни параметрів, взаємодії з іншими функціями, швидкості реакції та коректності роботи інтерфейсу.

Стабільність і продуктивність: перевірка стабільності та продуктивності програми під навантаженням. Оптимізація ресурсів комп'ютера та швидкодії різних функцій, таких як надсилання повідомлень та обробка даних.

Інтерфейс користувача: перевірка зручності та естетичності інтерфейсу, зокрема, забезпечення зручного розташування елементів, логіки навігації та взаємодії з користувачем.

Тестування з використанням Lighthouse [30]: використання інструменту Lighthouse для аналізу якості та продуктивності веб-сайту. Lighthouse допомагає виявити проблеми з швидкістю завантаження сторінки, доступністю та іншими ключовими метриками продуктивності. Результати для десктопної версії (рис. 3.17).

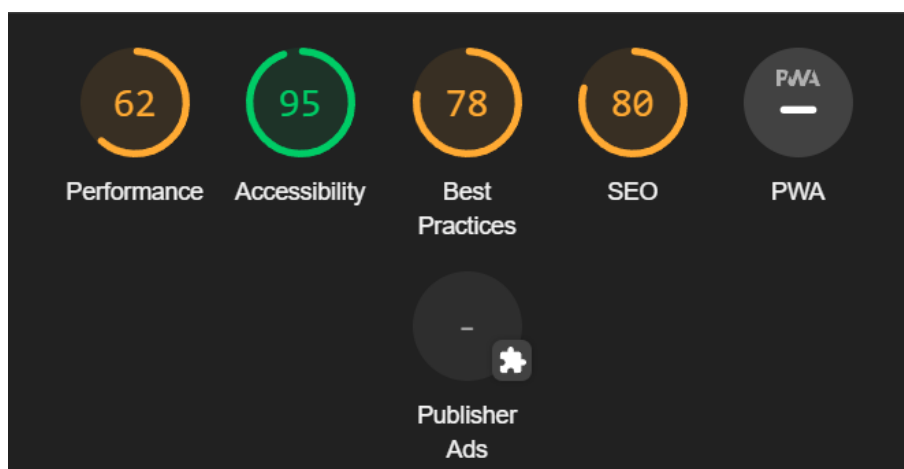


Рис. 3.17 Результати тестування за допомогою Lighthouse десктопної версії чату
Результати для мобільної версії (рис. 3.18).

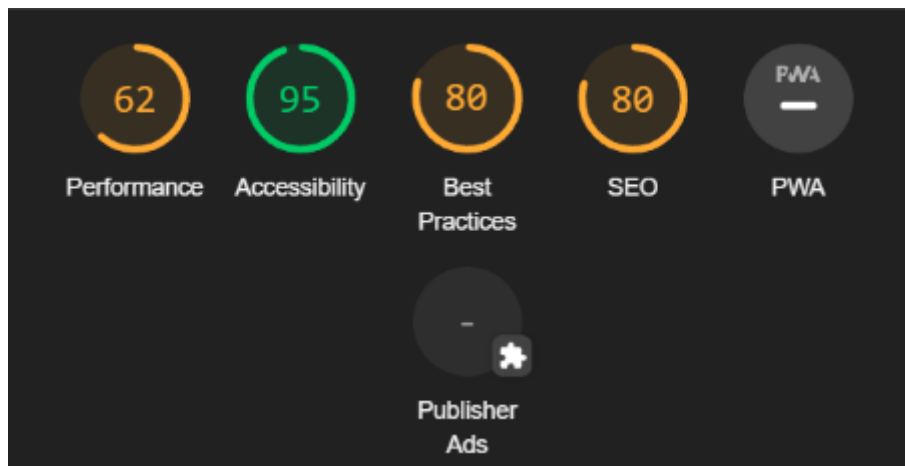


Рис. 3.18 Результати тестування за допомогою Lighthouse мобільної версії чату

Тестування безпеки: перевірка на вразливості та захист від атак, таких як перехоплення даних, введення невідомих скриптів тощо.

Тестування в різних браузерах: перевірка сумісності та коректності роботи програми в різних веб-браузерах, таких як Google Chrome, Mozilla Firefox, Safari, Microsoft Edge тощо.

Проводячи тестування роботи нашого онлайн чату, ми можемо переконалися, що він відповідає необхідним функціональним можливостям, працює стабільно та ефективно в різних умовах використання. Ці тести допомагають виявити та усунути будь-які потенційні проблеми або помилки, забезпечуючи надійну та зручну роботу нашого онлайн чату.

Висновки до розділу 3

У цьому розділі ми детально розглянули різні аспекти нашого проекту онлайн чату в реальному часі. Почали ми з опису порядку встановлення та налаштування параметрів платформи, необхідних для роботи нашого чату. За допомогою налаштування середовища розробки та підключення відповідних бібліотек та інструментів, ми забезпечили надійну базу для подальшої розробки та впровадження нашого чату.

Далі ми детально розглянули інтерфейс та порядок роботи нашого чату. Описали функціональні можливості, які пропонує наша система, і зосередилися на зручному дизайні та ефективному користуванні. Пояснили, як користувач

може зареєструватися, увійти та взаємодіяти з іншими користувачами через наш чат.

Останній пункт розділу був присвячений тестуванню роботи нашого чату. Ми перерахували кілька ключових аспектів, які потрібно перевірити під час тестування, і навели приклади методів, які можна використовувати для тестування онлайн чату в реальному часі, зокрема, використання інструменту Lighthouse та тестування в різних браузерах.

Загалом, цей розділ надав нам глибокого розуміння всіх аспектів розробки та функціонування нашого онлайн чату, що дозволить нам ефективно працювати над подальшим вдосконаленням та розвитком проекту.

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Основний результат

Метою нашого дослідження було створення та аналіз ефективності функціонування онлайн чату в реальному часі. Ми успішно розробили та впровадили чат, який дозволяє користувачам взаємодіяти між собою у реальному часі через веб-додаток.

Реалізовані функції

- Реєстрація та авторизація користувачів, що дозволяє забезпечити безпеку та обмеження доступу до чату.
- Відправлення текстових повідомлень та файлів між користувачами, що забезпечує зручну комунікацію.
- Підтримка групових чатів, що дозволяє користувачам обговорювати теми у розширеному колективі.
- Відслідковування статусу користувачів (онлайн, офлайн), що полегшує взаємодію та сприяє активності в чаті.
- Інтеграція з Socket.io для реалізації миттєвого оновлення даних та сповіщення користувачів про нові повідомлення.

Оцінка відповідності завданню

Всі реалізовані функції відповідають завданню створення онлайн чату в реальному часі. Чат забезпечує зручну та ефективну комунікацію між користувачами, має стабільну роботу та швидкий обмін повідомленнями.

Практична цінність результатів

Створений чат може бути використаний для спілкування користувачів у реальному часі на веб-сайтах, платформах або додатках. Він дозволяє створювати спільноти, обговорювати теми, спільно працювати та вирішувати завдання.

Напрямки подальшого вдосконалення

Розширення функціональності чату за допомогою додаткових можливостей, таких як відеозв'язок, висилання стікерів тощо.

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

Покращення інтерфейсу користувача для забезпечення ще зручнішого та привабливішого дизайну.

Оптимізація продуктивності та безпеки чату для забезпечення масштабованості та стійкості під час великого навантаження.

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Миттєві повідомлення [Електронний ресурс] – Режим доступу до ресурсу: <https://umity.in.ua/concept/?id=276>.
2. Duarte F. Most Popular Messaging Apps (2024) [Електронний ресурс] / Fabio Duarte. – 2024. – Режим доступу до ресурсу: <https://explodingtopics.com/blog/messaging-apps-stats>.
3. 10 Best Messaging Apps and How to Choose the Right One for You [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ringcentral.com/gb/en/blog/messaging-apps/>.
4. Документація MongoDB [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mongodb.com/docs/>.
5. Документація Node.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/docs/latest/api/>.
6. Ерік Ф. Програмування на JavaScript / Ф. Ерік, Р. Елізабет., 2022. – 672 с.
7. Express/Node introduction [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction.
8. Документація Socket.io [Електронний ресурс] – Режим доступу до ресурсу: <https://socket.io/docs/v4/>.
9. Документація React.js [Електронний ресурс] – Режим доступу до ресурсу: <https://legacy.reactjs.org/docs/getting-started.html>.
10. Nur I. The MERN stack tutorial [Електронний ресурс] / Islam Nur. – 2023. – Режим доступу до ресурсу: <https://blog.logrocket.com/mern-stack-tutorial/>.
11. Reddy V. MongoDB chat schema and Mongoose chat schema for Chat Application [Електронний ресурс] / Vithal Reddy. – 2017. – Режим доступу до ресурсу: <https://stackfame.com/mongodb-chat-schema-mongoose-chat-schema-chat-application>.
12. Smusin M. How to Make a Chat Web App: A Step-by-Step Guide [Електронний ресурс] / Mitya Smusin. – 2023. – Режим доступу до ресурсу: <https://yellow.systems/blog/how-to-make-a-chat-web-app>.

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Lim G. Beginning Node.js, Express & MongoDB Development / Greg Lim., 2020. – (Jason Lim).
14. Precious L. Building and structuring a Node.js MVC application [Електронний ресурс] / Luke Precious. – 2024. – Режим доступу до ресурсу: <https://blog.logrocket.com/building-structuring-node-js-mvc-application/>.
15. Routing [Електронний ресурс] – Режим доступу до ресурсу: <https://expressjs.com/en/guide/routing.html>.
16. Dillion M. Multer: Easily upload files with Node.js and Express [Електронний ресурс] / Megida Dillion. – 2022. – Режим доступу до ресурсу: <https://blog.logrocket.com/multer-nodejs-express-upload-file/>.
17. Nilesh R. How to Configure CORS in Node.js With Express [Електронний ресурс] / Raut Nilesh. – 2024. – Режим доступу до ресурсу: <https://dev.to/speaklouder/how-to-configure-cors-in-nodejs-with-express-11h>.
18. Denenberg D. How To Use JSON Web Tokens (JWTs) in Express.js [Електронний ресурс] / Danny Denenberg. – 2021. – Режим доступу до ресурсу: <https://www.digitalocean.com/community/tutorials/nodejs-jwt-expressjs>.
19. Harsh P. Password hashing in Node.js with bcrypt [Електронний ресурс] / Patel Harsh. – 2023. – Режим доступу до ресурсу: <https://blog.logrocket.com/password-hashing-node-js-bcrypt/>.
20. Griffiths D. React Cookbook: Recipes for Mastering the React Framework / D. Griffiths, D. Griffiths., 2021. – 510 с.
21. Onyebuchi B. How to Use React Context in Your Project – Beginner's Guide [Електронний ресурс] / Benedicta Onyebuchi. – 2024. – Режим доступу до ресурсу: <https://www.freecodecamp.org/news/how-to-use-react-context/>.
22. Документація Material UI [Електронний ресурс] – Режим доступу до ресурсу: <https://mui.com/material-ui/getting-started/>.
23. Britto V. React Design Patterns: Provider Pattern [Електронний ресурс] / Vitor Britto. – 2023. – Режим доступу до ресурсу:

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

<https://medium.com/@vitorbritto/react-design-patterns-provider-pattern-b273ba665158>.

24. Tutorial react-router-dom [Електронний ресурс] – Режим доступу до ресурсу: <https://reactrouter.com/en/main/start/tutorial>.
25. Документація react-hook-form [Електронний ресурс] – Режим доступу до ресурсу: <https://react-hook-form.com/docs>.
26. MUI styled-components [Електронний ресурс] – Режим доступу до ресурсу: <https://mui.com/system/styled/>.
27. Документація axios [Електронний ресурс] – Режим доступу до ресурсу: <https://axios-http.com/docs/intro>.
28. How to build a basic Socket.IO client [Електронний ресурс] – Режим доступу до ресурсу: <https://socket.io/how-to/build-a-basic-client>.
29. Medlock J. An Introduction to Environment Variables and How to Use Them [Електронний ресурс] / Jim Medlock. – 2018. – Режим доступу до ресурсу: <https://medium.com/chingu/an-introduction-to-environment-variables-and-how-to-use-them-f602f66d15fa>.
30. Lighthouse docs [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.chrome.com/docs/lighthouse/overview>.

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

packages/backend/index.js:

```
require("dotenv").config();

const bodyParser = require("body-parser");
const cors = require("cors");
const express = require("express");
const multer = require("multer");
const path = require("path");

const connectDB = require("../configs/bd");

const userRouter = require("../routes/userRoutes");
const messageRouter = require("../routes/messageRoutes");
const chatRouter = require("../routes/chatRoutes");
const { updateUserStatus } = require("../controllers/userController");
const {
  updateMessageStatus,
  updateMessage,
  deleteMessage,
} = require("../controllers/messageController");

connectDB();
const app = express();

const host = process.env.HOST;
const port = process.env.PORT;

const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, "uploads/");
  },
  filename: function (req, file, cb) {
    cb(null, Date.now() + path.extname(file.originalname));
  },
});
const upload = multer({ storage: storage });

app.use(cors());
app.use(bodyParser.json());

app.post("/upload", upload.single("file"), (req, res) => {
  const file = req.file;
  const { originalname, filename, size, mimetype } = file;
  const url = `/uploads/${filename}`;
  res.status(200).json({ originalname, filename, url, size, mimetype });
});

app.use("/uploads", express.static("uploads"));

app.use("/user", userRouter);
app.use("/chat", chatRouter);
app.use("/message", messageRouter);

const server = app.listen(port, () => {
  console.log(`Server is running on ${host}:${port}`);
});

const io = require("socket.io")(server, {
  pingTimeout: 60000,
  cors: {
    origin: process.env.FRONTEND_ROUTE,
  },
});

const userStatuses = {};

io.on("connection", (socket) => {
  socket.on("setup", (userData) => {
```

		Іщенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    socket.join(userData._id);
    socket.emit("connected");
    socket.userId = userData._id;
    updateUserStatus(userData._id, "online");
    userStatuses[userData._id] = { status: "online" };
    io.emit("userStatus", userStatuses);
  });

  socket.on("join", (room) => {
    socket.join(room);
  });

  socket.on("typing", (room) => socket.in(room).emit("typing", room));
  socket.on("stopTyping", (room) => socket.in(room).emit("stopTyping", room));

  socket.on("send", (message) => {
    const chat = message.chat;

    const users = chat.isGroupChat
      ? chat.groupAdmins.concat(chat.users)
      : chat.users;

    if (!users || users.length === 0) return console.log("No chat users");

    users.forEach((user) => {
      if (user._id === message.sender._id) return;
      socket.in(user._id).emit("received", message);
    });
  });

  socket.on("editMessage", ({ editMessageId, updatedMessage, attachments }) => {
    updateMessage(editMessageId, updatedMessage, attachments);
    io.emit("messageEdited", {
      _id: editMessageId,
      text: updatedMessage,
      attachments,
    });
  });

  socket.on("deleteMessage", (messageId) => {
    deleteMessage(messageId);
    io.emit("messageDeleted", messageId);
  });

  socket.on("markAsSent", async (messageId) => {
    updateMessageStatus(messageId, "sent");
    io.emit("messageStatusUpdate", { messageId, status: "sent" });
  });

  socket.on("markAsRead", async ({ messageId, userId }) => {
    updateMessageStatus(messageId, "read", userId);
    io.emit("messageStatusUpdate", { messageId, status: "read" });
  });

  socket.on("disconnect", () => {
    const userId = socket.userId;
    if (userId) {
      updateUserStatus(userId, "offline");
      userStatuses[userId] = { status: "offline", latestActivity: new Date() };
      io.emit("userStatus", userStatuses);
    }
  });

  socket.on("logout", (userId) => {
    updateUserStatus(userId, "offline");
    userStatuses[userId] = { status: "offline", latestActivity: new Date() };
    io.emit("userStatus", userStatuses);
  });

```

```

    socket.off("setup", () => {
      socket.leave(userData._id);
    });
  });
});

```

packages/backend/configs/db.js:

```

const mongoose = require("mongoose");

const connectDB = async () => {
  try {
    const connection = await mongoose.connect(process.env.MONGO_PATH);
  } catch (err) {
    console.error(err);
    process.exit();
  }
};

module.exports = connectDB;

```

packages/backend/configs/getToken.js:

```

const jwt = require("jsonwebtoken");

const secretKey = process.env.SECRET_KEY;

const getToken = (data) => {
  return jwt.sign(data, secretKey, { expiresIn: "12h" });
};

module.exports = getToken;

```

packages/backend/controllers/chatController.js:

```

const Chat = require("../models/chatModel");
const User = require("../models/userModel");

const createChat = async (req, res) => {
  const { userId } = req.body;

  if (!userId) {
    return res.status(400).json({ error: "UserId wasn't sent" });
  }

  try {
    const existingChat = await Chat.findOne({
      isGroupChat: false,
      users: { $all: [req.user._id, userId] },
    })
      .populate("users", "-password")
      .populate("latestMessage");

    if (existingChat) {
      return res.status(200).json(existingChat);
    }

    const chatData = {
      chatName: "sender",
      isGroupChat: false,
      users: [req.user._id, userId],
    };

    const createdChat = await Chat.create(chatData);
    const chat = await Chat.findOne({ _id: createdChat._id }).populate(
      "users",
      "-password"
    );
  }
};

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    return res.status(200).json(chat);
  } catch (err) {
    console.error("Error creating chat:", err);
    return res.status(500).json({ error: "Internal Server Error" });
  }
};

const getMyChats = async (req, res) => {
  try {
    let chats = await Chat.find({
      $or: [
        { users: { $elemMatch: { $eq: req.user._id } } },
        { groupAdmins: { $elemMatch: { $eq: req.user._id } } },
      ],
    })
      .populate("users", "-password")
      .populate("groupAdmins", "-password")
      .populate("latestMessage")
      .sort({ "latestMessage.createdAt": -1, createdAt: -1 });

    if (!chats || chats.length === 0) {
      return res.json([]);
    }

    let results = await User.populate(chats, {
      path: "latestMessage.sender",
      select: "firstName lastName avatar email",
    });

    results = results.sort((a, b) => {
      const dateA = a.latestMessage ? a.latestMessage.createdAt : a.createdAt;
      const dateB = b.latestMessage ? b.latestMessage.createdAt : b.createdAt;
      return new Date(dateB) - new Date(dateA);
    });

    return res.json(results);
  } catch (error) {
    console.error("Error fetching chats:", error);
    return res.status(500).json({ error: "Internal Server Error" });
  }
};

const createGroupChat = async (req, res) => {
  const { title, description, avatar, users, groupAdmins } = req.body;

  if (!users || !title) {
    return res.status(400).json({
      error: "Invalid request parameters. Users and title are required.",
    });
  }

  try {
    const groupChat = await Chat.create({
      chatName: title,
      chatDescription: description,
      avatar: avatar,
      users: users,
      isGroupChat: true,
      groupAdmins: groupAdmins,
    });

    const groupChatData = await Chat.findOne({ _id: groupChat._id })
      .populate("users", "-password")
      .populate("groupAdmins", "-password");

    return res.status(200).json(groupChatData);
  } catch (error) {
    return res.status(500).json({ error: error });
  }
};

```

		Іщенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
  };

  const updateGroupChat = async (req, res) => {
    const { chatId, title, description, avatar, users, groupAdmins } = req.body;

    const updatedChat = await Chat.findByIdAndUpdate(
      chatId,
      {
        chatName: title,
        chatDescription: description,
        avatar,
        users,
        groupAdmins,
      },
      { new: true }
    )
      .populate("users", "-password")
      .populate("groupAdmins", "-password");

    if (!updatedChat) {
      return res.status(404).send("Chat not found");
    } else {
      return res.json(updatedChat);
    }
  };

  const leaveGroupChat = async (req, res) => {
    const { chatId } = req.body;
    const userId = req.user._id;

    try {
      const chat = await Chat.findById(chatId);

      if (!chat) {
        return res.status(404).json({ error: "Chat not found" });
      }

      const isUserMember = chat.users.some((user) => user.equals(userId));
      const isUserAdmin = chat.groupAdmins.some((admin) => admin.equals(userId));

      if (!isUserMember && !isUserAdmin) {
        return res
          .status(403)
          .json({ error: "User is not a member of the chat" });
      }

      if (isUserMember) {
        chat.users.pull(userId);
      }

      if (isUserAdmin) {
        chat.groupAdmins.pull(userId);
      }

      await chat.save();

      return res.status(200).json({ message: "User left the chat successfully" });
    } catch (error) {
      console.error("Error leaving chat:", error);
      return res.status(500).json({ error: "Internal Server Error" });
    }
  };

  const deleteChat = async (req, res) => {
    const { chatId } = req.params;
    const userId = req.user._id;

    try {

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const chat = await Chat.findById(chatId);

if (!chat) {
  return res.status(404).json({ error: "Chat not found" });
}

const isGroupChat = chat.isGroupChat;
const isUserInChat = chat.users.some((user) => user.equals(userId));

if (!isUserInChat && !isGroupChat) {
  return res
    .status(403)
    .json({ error: "You do not have permission to delete this chat" });
}

if (isGroupChat && chat.groupAdmins.length > 0) {
  const isAdmin = chat.groupAdmins.some((admin) => admin.equals(userId));
  if (!isAdmin) {
    return res
      .status(403)
      .json({ error: "You do not have permission to delete this chat" });
  }
}

await Chat.findByIdAndDelete(chatId);

return res
  .status(200)
  .json({ message: "Chat deleted successfully", _id: chatId });
} catch (error) {
  console.error("Error deleting chat:", error);
  return res.status(500).json({ error: "Internal Server Error" });
}
};

module.exports = {
  createChat,
  getMyChats,
  createGroupChat,
  updateGroupChat,
  leaveGroupChat,
  deleteChat,
};

```

packages/backend/controllers/messageController.js:

```

const Chat = require("../models/chatModel");
const Message = require("../models/messageModel");
const Notification = require("../models/notificationModel");

const sendMessage = async (req, res) => {
  const { chatId, content, attachments } = req.body;

  if (!chatId || (!content.trim() && attachments.length === 0)) {
    return res.status(400).json({ error: "Invalid data sent" });
  }

  try {
    const newMessage = await Message.create({
      sender: req.user._id,
      text: content,
      chat: chatId,
      attachments: attachments,
    });

    const populatedMessage = await Message.findById(newMessage._id)
      .populate("sender", "avatar firstName lastName")
      .lean();
  }
};

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const chat = await Chat.findById(chatId).lean();

if (chat && chat.users) {
  populatedMessage.chat = {
    _id: chat._id,
    users: chat.users.map((user) => ({
      _id: user._id,
    })),
    groupAdmins: chat.groupAdmins.map((admin) => ({
      _id: admin._id,
    })),
    isGroupChat: chat.isGroupChat,
  };
}

await Chat.findByIdAndUpdate(chatId, { latestMessage: newMessage._id });

for (const user of [...chat.users, ...chat.groupAdmins]) {
  if (user._id.toString() !== req.user._id.toString()) {
    await Notification.create({
      message: newMessage._id,
      user: user._id,
      chat: chat._id,
    });
  }
}

return res.status(200).json(populatedMessage);
} catch (error) {
  return res.status(400).json({ error: `Error sending message: ${error}` });
}
};

const updateMessage = async (messageId, updatedContent, attachments) => {
  if (!messageId || (!updatedContent.trim() && attachments.length === 0)) {
    return res.status(400).json({ error: "Invalid data sent" });
  }

  try {
    const updatedMessage = await Message.findByIdAndUpdate(
      messageId,
      { text: updatedContent, attachments: attachments },
      { new: true }
    ).populate("sender", "avatar firstName lastName");

    if (!updatedMessage) {
      return res.status(404).json({ error: "Message not found" });
    }
    return updatedMessage;
  } catch (error) {
    console.error(`Error updating message: ${error}`);
  }
};

const updateMessageStatus = async (messageId, status, userId) => {
  try {
    const message = await Message.findByIdAndUpdate(messageId, { status });
    status === "read" &&
      userId &&
      deleteNotification(messageId, message.chat, userId);
  } catch (error) {
    console.error("Error updating message status:", error);
  }
};

const deleteNotification = async (messageId, chatId, userId) => {
  try {
    await Notification.deleteMany({
      message: messageId,
    });
  }
};

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        user: userId,
        chat: chatId,
    });
    } catch (error) {
        console.error("Error deleting notification:", error);
    }
};

const deleteMessage = async (messageId) => {
    try {
        await Message.findByIdAndDelete(messageId);
    } catch (error) {
        console.error("Error deleting message:", error);
    }
};

const getAllMessages = async (req, res) => {
    try {
        const messages = await Message.find({ chat: req.params.chatId })
            .populate("sender", "firstName lastName avatar email")
            .populate("chat");

        return res.json(messages);
    } catch (error) {
        return res.status(400).json({ error: `Error getting messages: ${error}` });
    }
};

const getNotificationMessages = async (req, res) => {
    try {
        const notifications = await Notification.find({ user: req.user._id });

        const messageIds = notifications.map(
            (notification) => notification.message
        );

        const messages = await Message.find({ _id: { $in: messageIds } })
            .populate("sender", "firstName lastName avatar email")
            .populate("chat");

        return res.json(messages);
    } catch (error) {
        return res.status(400).json({ error: `Error getting messages: ${error}` });
    }
};

module.exports = {
    sendMessage,
    updateMessage,
    updateMessageStatus,
    getAllMessages,
    deleteMessage,
    getNotificationMessages,
};

```

packages/backend/controllers/userController.js:

```

const getToken = require("../configs/getToken");
const User = require("../models/userModel");
const bcrypt = require("bcrypt");

const signUp = async (req, res) => {
    const { firstName, lastName, avatar, email, password } = req.body;

    if (!firstName || !lastName || !email || !password) {
        return res.status(400).json({ error: "Some fields are empty" });
    }

    const userExist = await User.findOne({ email });

    if (userExist) {

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    return res
      .status(400)
      .json({ error: "User with this email already exist" });
  }

  const hashedPassword = await bcrypt.hash(password, 10);

  const user = await User.create({
    firstName,
    lastName,
    avatar,
    email,
    password: hashedPassword,
  });

  if (user) {
    const data = {
      _id: user._id,
      firstName: user.firstName,
      lastName: user.lastName,
      email: user.email,
      avatar: user.avatar,
    };
    return res.status(201).json({ ...data, token: getToken(data) });
  } else {
    return res.status(400).json({ error: "Failed to register user" });
  }
};

const signIn = async (req, res) => {
  const { email, password } = req.body;

  const user = await User.findOne({ email });

  if (user && (await bcrypt.compare(password, user.password))) {
    const data = {
      _id: user._id,
      firstName: user.firstName,
      lastName: user.lastName,
      email: user.email,
      avatar: user.avatar,
    };
    res.json({ ...data, token: getToken(data) });
  } else {
    return res.status(401).json({ error: "Invalid email or password" });
  }
};

const search = async (req, res) => {
  const search = req.params.search
    ? {
        $or: [
          { firstName: { $regex: req.params.search, $options: "i" } },
          { lastName: { $regex: req.params.search, $options: "i" } },
          { email: { $regex: req.params.search, $options: "i" } },
        ],
      }
    : {};

  const users = await User.find(search).find({ _id: { $ne: req.user._id } });
  res.send(users);
};

const getMe = async (req, res) => {
  const user = req.user;

  if (!user) return res.status("401");

  return res.json(user);
};

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

};

const updateMe = async (req, res) => {
  const { firstName, lastName, email, avatar } = req.body;
  const { _id } = req.user;
  try {
    await User.findByIdAndUpdate(_id, {
      firstName,
      lastName,
      email,
      avatar,
    });

    const data = {
      _id,
      firstName,
      lastName,
      email,
      avatar,
    };

    return res.status(200).json({ ...data, token: await getToken(data) });
  } catch (error) {
    return res.status(500).json({ error: `Error updating user: ${error}` });
  }
};

const changePassword = async (req, res) => {
  const { oldPassword, newPassword } = req.body;
  const { _id } = req.user;
  try {
    const user = await User.findById(_id);
    if (user && (await bcrypt.compare(oldPassword, user.password))) {
      await User.findByIdAndUpdate(_id, {
        password: await bcrypt.hash(newPassword, 10),
      });
      const data = {
        _id: user._id,
        firstName: user.firstName,
        lastName: user.lastName,
        email: user.email,
        avatar: user.avatar,
      };
      return res.status(200).json(data);
    }
    return res.status(400).json({ error: "Old password is incorrect" });
  } catch (error) {
    return res.status(500).json({ error: `Error changing password: ${error}` });
  }
};

const updateUserStatus = async (userId, status) => {
  if (status === "online")
    await User.findByIdAndUpdate(userId, { status: status });
  else if (status === "offline")
    await User.findByIdAndUpdate(userId, {
      status: status,
      latestActivity: new Date(),
    });
};

module.exports = {
  signUp,
  signIn,
  search,
  getMe,
  updateMe,
  changePassword,
  updateUserStatus,
};

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

```
};
```

packages/backend/middlewares/authMiddleware.js:

```
const jwt = require("jsonwebtoken");

const protect = async (req, res, next) => {
  const authHeader = req.headers.authorization;

  if (authHeader && authHeader.startsWith("Bearer ")) {
    const token = authHeader.split(" ")[1];

    if (!token) {
      return res.sendStatus(401);
    }

    jwt.verify(token, process.env.SECRET_KEY, (err, user) => {
      if (err) {
        return res.sendStatus(401);
      }
      req.user = user;
      next();
    });
  } else {
    return res.sendStatus(401);
  }
};

module.exports = protect;
```

packages/backend/models/chatModel.js:

```
const mongoose = require("mongoose");

const chatModel = mongoose.Schema(
  {
    chatName: { type: String, trim: true },
    chatDescription: { type: String, trim: true },
    avatar: { type: String },
    isGroupChat: { type: Boolean, default: false },
    users: [
      {
        type: mongoose.Schema.Types.ObjectId,
        ref: "User",
      },
    ],
    latestMessage: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "Message",
    },
    groupAdmins: [
      {
        type: mongoose.Schema.Types.ObjectId,
        ref: "User",
      },
    ],
  },
  { timestamps: true }
);

const Chat = mongoose.model("Chat", chatModel);

module.exports = Chat;
```

packages/backend/models/messageModel.js:

```
const mongoose = require("mongoose");
```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const messageModel = mongoose.Schema(
  {
    sender: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
    },
    text: { type: String, trim: true },
    attachments: [{ type: Object }],
    chat: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "Chat",
    },
    status: {
      type: String,
      enum: ["sending", "sent", "read"],
      default: "sending",
    },
  },
  { timestamps: true }
);

const Message = mongoose.model("Message", messageModel);

module.exports = Message;

```

packages/backend/models/notificationModel.js:

```

const mongoose = require("mongoose");

const notificationModel = mongoose.Schema(
  {
    message: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "Message",
    },
    user: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
    },
    chat: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "Chat",
    },
  },
  { timestamps: true }
);

const Notification = mongoose.model("Notification", notificationModel);

module.exports = Notification;

```

packages/backend/models/userModel.js:

```

const mongoose = require("mongoose");

const userModel = mongoose.Schema(
  {
    firstName: { type: String, trim: true, required: true },
    lastName: { type: String, trim: true, required: true },
    email: { type: String, trim: true, required: true, unique: true },
    password: { type: String, required: true },
    avatar: { type: String },
    status: { type: String, enum: ["online", "offline"], default: "offline" },
    latestActivity: { type: Date },
  },
  { timestamps: true }
);

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

```
const User = mongoose.model("User", userModel);

module.exports = User;
```

packages/backend/routes/chatRoutes.js:

```
const express = require("express");
const protect = require("../middlewares/authMiddleware");
const {
  createChat,
  getMyChats,
  createGroupChat,
  updateGroupChat,
  leaveGroupChat,
  deleteChat,
} = require("../controllers/chatController");

const router = express.Router();

router.post("/", protect, createChat);
router.get("/", protect, getMyChats);
router.delete("/:chatId", protect, deleteChat);
router.post("/group", protect, createGroupChat);
router.put("/group", protect, updateGroupChat);
router.put("/group/leave", protect, leaveGroupChat);

module.exports = router;
```

packages/backend/routes/messageRoutes.js:

```
const express = require("express");
const {
  sendMessage,
  getAllMessages,
  getNotificationMessages,
} = require("../controllers/messageController");
const protect = require("../middlewares/authMiddleware");

const router = express.Router();

router.post("/", protect, sendMessage);
router.get("/notif", protect, getNotificationMessages);
router.get("/:chatId", protect, getAllMessages);

module.exports = router;
```

packages/backend/routes/userRoutes.js:

```
const express = require("express");
const {
  signUp,
  signIn,
  search,
  getMe,
  updateMe,
  changePassword,
} = require("../controllers/userController");
const protect = require("../middlewares/authMiddleware");

const router = express.Router();

router.post("/sign-in", signIn);
router.post("/sign-up", signUp);
router.get("/me", protect, getMe);
router.put("/me", protect, updateMe);
router.put("/password", protect, changePassword);
router.get("/:search", protect, search);

module.exports = router;
```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

packages/frontend/src/App.jsx:

```
import React from "react";
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
import { createTheme, ThemeProvider } from "@mui/material/styles";

import SignIn from "../pages/SignIn";
import SignUp from "../pages/SignUp";

import Chat from "../pages/Chat";

import PrivateRoute from "../components/Route/PrivateRoute";
import PublicRoute from "../components/Route/PublicRoute";

import { AuthProvider } from "../contexts/AuthContext";

const theme = createTheme({
  palette: {
    background: {
      default: "#D2E4D6",
    },
    primary: {
      main: "#31473A",
    },
    secondary: {
      main: "#D2E4D6",
    },
  },
});

const App = () => {
  return (
    <Router>
      <AuthProvider>
        <ThemeProvider theme={theme}>
          <Routes>
            <Route path="/sign-in" element={<PublicRoute element={SignIn} />} />
            <Route path="/sign-up" element={<PublicRoute element={SignUp} />} />

            <Route path="/" element={<PrivateRoute element={Chat} />} />
          </Routes>
        </ThemeProvider>
      </AuthProvider>
    </Router>
  );
};

export default App;
```

packages/frontend/src/components/Attachment/Attachment.jsx:

```
import React from "react";
import { Box, Typography } from "@mui/material";
import InsertDriveFileIcon from "@mui/icons-material/InsertDriveFile";
import CloseIcon from "@mui/icons-material/Close";
import ImageIcon from "@mui/icons-material/Image";
import AudiotrackIcon from "@mui/icons-material/Audiotrack";
import VideoLibraryIcon from "@mui/icons-material/VideoLibrary";
import AudioPlayer from "../AudioPlayer/AudioPlayer";

const Attachment = ({ attachment, theme, onDelete, openFile }) => {
  const formatFileSize = (sizeInBytes) => {
```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    if (sizeInBytes >= 1024 * 1024) {
      return `${(sizeInBytes / (1024 * 1024)).toFixed(1)} MB`;
    } else if (sizeInBytes >= 1024) {
      return `${Math.ceil(sizeInBytes / 1024)} KB`;
    } else {
      return `1 KB`;
    }
  };

const bgColor = theme === "dark" ? "inherit" : "secondary.main";
const textColor = theme === "dark" ? "secondary.main" : "primary.main";

const getFileIcon = (mimeType) => {
  if (mimeType.startsWith("image/")) {
    return <ImageIcon sx={{ color: textColor, mr: 1 }} />;
  } else if (mimeType.startsWith("audio/")) {
    return <AudioTrackIcon sx={{ color: textColor, mr: 1 }} />;
  } else if (mimeType.startsWith("video/")) {
    return <VideoLibraryIcon sx={{ color: textColor, mr: 1 }} />;
  } else {
    return <InsertDriveFileIcon sx={{ color: textColor, mr: 1 }} />;
  }
};

const renderFileView = (attachment) => {
  const fileUrl = `${process.env.REACT_APP_API_BASE_URL}${attachment.url}`;
  const mimeType = attachment.mimetype || "None";

  if (mimeType.startsWith("image/")) {
    return (
      <img
        src={fileUrl}
        alt={attachment.originalname}
        style={{ maxHeight: "200px", borderRadius: "4px", margin: openFile && "-24px -24px 0px -24px" }}
      />
    );
  } else if (mimeType.startsWith("audio/")) {
    return <AudioPlayer src={fileUrl} color={textColor} sx={{ margin: openFile && "-12px -12px 0px -12px" }} />;
  } else if (mimeType.startsWith("video/")) {
    return <video
      controls src={fileUrl}
      style={{ maxHeight: "200px", borderRadius: "4px", margin: openFile && "-24px -24px 0px -24px" }}
    />;
  } else {
    return (
      <>
        {getFileIcon(mimeType)}
        <Box>
          <Typography variant="body2" color={textColor}>
            {attachment.originalname}
          </Typography>
          <Typography variant="body2" color={textColor} sx={{ opacity: 0.5 }}>
            {formatFileSize(attachment.size)}
          </Typography>
        </Box>
      </>
    );
  }
};

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

return (
  <Box
    sx={{
      p: 1,
      bgcolor: bgColor,
      borderRadius: "4px",
      display: "flex",
      alignItems: "center",
      m: 1,
      cursor: "pointer",
    }}
    onClick={() =>
      openFile && attachment.mimetype && attachment.mimetype.startsWith("image/") &&
      window.open(
        `${process.env.REACT_APP_API_BASE_URL}${attachment.url}`,
        "_blank"
      )
    }
  >
    {renderFileView(attachment)}
    {onDelete && (
      <CloseIcon
        sx={{ ml: 1 }}
        onClick={(e) => {
          e.stopPropagation();
          onDelete();
        }}
      />
    )}
  </Box>
);
};

export default Attachment;

```

packages/frontend/src/components/AudioPlayer/AudioPlayer.jsx:

```

import React, { useRef, useState, useEffect } from 'react';
import { IconButton, Slider, Typography, Box } from '@mui/material';
import PlayArrowIcon from '@mui/icons-material/PlayArrow';
import PauseIcon from '@mui/icons-material/Pause';

const AudioPlayer = ({ src, color, sx }) => {
  const audioRef = useRef(null);
  const [isPlaying, setIsPlaying] = useState(false);
  const [currentTime, setCurrentTime] = useState(0);
  const [duration, setDuration] = useState(0);

  useEffect(() => {
    const audio = audioRef.current;

    const handleTimeUpdate = () => {
      setCurrentTime(audio.currentTime);
    };

    const handleLoadedMetadata = () => {
      setDuration(audio.duration);
    };

    audio.addEventListener('timeupdate', handleTimeUpdate);
    audio.addEventListener('loadedmetadata', handleLoadedMetadata);

    return () => {
      audio.removeEventListener('timeupdate', handleTimeUpdate);
    };
  }, [audioRef]);

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О.В.				88
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        audio.removeEventListener('loadedmetadata', handleLoadedMetadata);
    };
}, []);

const togglePlayPause = () => {
    const audio = audioRef.current;
    if (isPlaying) {
        audio.pause();
    } else {
        audio.play();
    }
    setIsPlaying(!isPlaying);
};

const handleSliderChange = (event, newValue) => {
    const audio = audioRef.current;
    audio.currentTime = newValue;
    setCurrentTime(newValue);
};

const formatTime = (time) => {
    const minutes = Math.floor(time / 60);
    const seconds = Math.floor(time % 60);
    return `${minutes}:${seconds < 10 ? '0' : ''}${seconds}`;
};

return (
    <Box display="flex" alignItems="center" width="100%" sx={sx}>
        <IconButton onClick={togglePlayPause}>
            {isPlaying ? <PauseIcon sx={{ color: color }} /> : <PlayArrowIcon sx={{ color:
color }} />}
        </IconButton>
        <Typography variant="body2" color={color}>
            {formatTime(currentTime)}
        </Typography>
        <Slider
            value={currentTime}
            min={0}
            max={duration}
            onChange={handleSliderChange}
            sx={{ mx: 2, minWidth: "200px", color: color }}
        />
        <Typography variant="body2" color={color}>
            {formatTime(duration)}
        </Typography>
        <audio ref={audioRef} src={src} />
    </Box>
);
};

export default AudioPlayer;

```

packages/frontend/src/components/ChatBox/ChatBox.jsx:

```

import React, { useEffect, useState } from "react";
import {
    Box,
    IconButton,
    Typography,
    TextField,
    AppBar,
    Toolbar,
    Avatar,
    CircularProgress,
} from "@mui/material";
import axios from "../../utils/axios.js";
import ArrowBackIcon from "@mui/icons-material/ArrowBack";
import styled from "@emotion/styled";
import GroupsIcon from "@mui/icons-material/Groups";

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				89
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import { getChatInfo } from "../../utils/chat";
import GroupInfoModal from "../../Modals/Group/GroupInfoModal";
import ChatInfoModal from "../../Modals/Chat/ChatInfoModal";
import { useAuth } from "../../contexts/AuthContext";
import { formatDate } from "../../utils/datetime";
import ScrollableFeed from "react-scrollable-feed";
import GroupChatMessage from "../GroupChatMessage";
import { isLastMessage, isNewDayMessage } from "../../utils/messages";
import { useSocket } from "../../contexts/SocketContext";
import { getContact } from "../../utils/contacts";
import SingleChatMessage from "../SingleChatMessage";
import CreateMessageInput from "../CreateMessageInput";
import UpdateMessageInput from "../UpdateMessageInput";
import { decryptMessage, encryptMessage } from "../../utils/encryption";

const ContactAppBar = styled(AppBar)(({ theme }) => ({
  position: "static",
  background: theme.palette.secondary.main,
  color: theme.palette.primary.main,
  height: "64px",
  zIndex: "100",
})));

const ChatBoxContainer = styled(Box)({
  height: "calc(100vh - 64px)",
  display: "flex",
  flexDirection: "column",
});

const ChatContent = styled(ScrollableFeed)({
  padding: "16px",
  "&::-webkit-scrollbar": {
    width: "8px",
  },
  "&::-webkit-scrollbar-thumb": {
    backgroundColor: "#bbb",
    borderRadius: "4px",
  },
  "&::-webkit-scrollbar-thumb:hover": {
    backgroundColor: "#888",
  },
});

const ChatBox = ({
  isSmallScreen,
  user,
  currentChat,
  setCurrentChat,
  setUpdateChats,
  messages,
  setMessages,
}) => {
  const [loading, setLoading] = useState(false);
  const [newMessage, setNewMessage] = useState("");
  const [updatedMessage, setUpdatedMessage] = useState("");
  const [openChatInfoModal, setOpenChatInfoModal] = useState(false);
  const [openGroupInfoModal, setOpenGroupInfoModal] = useState(false);
  const [typing, setTyping] = useState(false);
  const [isTyping, setIsTyping] = useState(false);
  const [userStatus, setUserStatus] = useState("Offline");
  const [editMessage, setEditMessage] = useState(null);
  const { token } = useAuth();
  const {
    socket,
    socketConnected,
    usersStatuses,
    getUserStatus,
    selectedChatCompare,
    notification,
  } = useSocket();

```

```

    setNotification,
  } = useSocket();

  useEffect(() => {
    setUpdateChats(messages);
  }, [messages]);

  useEffect(() => {
    socket.on("typing", (room) => {
      if (room === currentChat._id) setIsTyping(true);
    });

    socket.on("stopTyping", (room) => {
      if (room === currentChat._id) setIsTyping(false);
    });

    return () => {
      socket.off("typing");
      socket.off("stopTyping");
    };
  }, [socket, currentChat]);

  useEffect(() => {
    const userId = getContact(currentChat, user._id);
    setUserStatus(getUserStatus(userId));
  }, [usersStatuses, currentChat, getUserStatus, user]);

  useEffect(() => {
    const getMessages = async () => {
      if (currentChat && currentChat._id) {
        try {
          setLoading(true);
          const response = await axios.get(`/message/${currentChat._id}`, {
            headers: {
              Authorization: `Bearer ${token}`,
            },
          });
        } catch (error) {}
      }
    };

    getMessages();
  }, [currentChat, token, socket]);

  useEffect(() => {
    socket.on("messageStatusUpdate", ({ messageId, status }) => {
      setMessages((prevMessages) =>
        prevMessages.map((message) =>
          message._id === messageId ? { ...message, status } : message
        )
      );
    });

    return () => {
      socket.off("messageStatusUpdate");
    };
  }, [messages, socket]);

  useEffect(() => {
    socket.on("messageEdited", ({ _id, text, attachments }) => {
      setMessages((prevMessages) =>
        prevMessages.map((message) =>
          message._id === _id ? { ...message, text, attachments } : message
        )
      );
    });
  });

```

```

    });

    return () => {
      socket.off("messageEdited");
    };
  }, [messages, socket]);

  useEffect(() => {
    socket.on("messageDeleted", (messageId) => {
      setMessages((prevMessages) =>
        prevMessages.filter((message) => message._id !== messageId)
      );
    });
  });

  return () => {
    socket.off("messageDeleted");
  };
}, [messages, socket]);

useEffect(() => {
  if (editMessage) setUpdatedMessage(decryptMessage(editMessage.text,
currentChat._id));
}, [editMessage]);

const updateMessage = async (attachments) => {
  if ((updatedMessage || attachments) && editMessage) {
    try {
      socket.emit("editMessage", {
        editMessageId: editMessage._id,
        updatedMessage: encryptMessage(updatedMessage, currentChat._id),
        attachments,
      });
      setEditMessage(null);
    } catch (error) {}
  }
};

const sendMessage = async (attachments) => {
  if (newMessage || attachments) {
    try {
      socket.emit("stopTyping", currentChat._id);
      const response = await axios.post(
        "/message/",
        {
          chatId: currentChat._id,
          content: encryptMessage(newMessage, currentChat._id),
          attachments: attachments,
        },
        {
          headers: {
            Authorization: `Bearer ${token}`,
          },
        }
      );
      socket.emit("send", response.data);
      socket.emit("markAsSent", response.data._id);
      setMessages((prevMessages) => [...prevMessages, response.data]);
      setNewMessage("");
    } catch (error) {}
  }
};

const typingHandler = (e) => {
  setNewMessage(e.target.value);

  if (!socketConnected) return;

  if (!typing) {
    setTyping(true);

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    socket.emit("typing", currentChat._id);
  }

  let lastTypingTime = new Date().getTime();
  var timerLength = 1500;

  setTimeout(() => {
    var timeNow = new Date().getTime();
    var timeDiff = timeNow - lastTypingTime;

    if (timeDiff >= timerLength && typing) {
      socket.emit("stopTyping", currentChat._id);
      setTyping(false);
    }
  }, timerLength);
};

const handleUpdateInput = (e) => {
  setUpdatedMessage(e.target.value);
};

const handleGoBack = () => {
  setCurrentChat({});
};

const chat = getChatInfo(currentChat, user);

const openModal = () => {
  currentChat.isGroupChat
    ? setOpenGroupInfoModal(true)
    : setOpenChatInfoModal(true);
};

return (
  <>
    <GroupInfoModal
      open={openGroupInfoModal}
      close={() => setOpenGroupInfoModal(false)}
      group={currentChat}
      user={user}
      setUpdateChats={setUpdateChats}
      setCurrentChat={setCurrentChat}
    />
    <ChatInfoModal
      open={openChatInfoModal}
      close={() => setOpenChatInfoModal(false)}
      chat={currentChat}
      user={user}
      setUpdateChats={setUpdateChats}
      setCurrentChat={setCurrentChat}
    />
    <ChatBoxContainer>
      <ContactAppBar>
        <Toolbar>
          {isSmallScreen && (
            <IconButton
              onClick={handleGoBack}
              edge="start"
              color="inherit"
              aria-label="back"
            >
              <ArrowBackIcon />
            </IconButton>
          )}
        </Toolbar>
      </ContactAppBar>
      <Box sx={{ flexGrow: 1 }}>
        <Typography variant="h6">{chat.title}</Typography>
        {!currentChat.isGroupChat && (
          <Typography variant="body1">

```

```

        {isTyping ? "Пише..." : userStatus}
      </Typography>
    ))
  </Box>
  <Box>
    <Avatar
      alt={chat.title}
      src={chat.avatar}
      sx={{ background: "gray", color: "white" }}
      onClick={openModal}
    >
      {currentChat.isGroupChat && !currentChat.avatar && (
        <GroupsIcon />
      )}
    </Avatar>
  </Box>
</Toolbar>
</ContactAppBar>

<ChatContent>
  {loading ? (
    <Box
      sx={{
        height: "100%",
        display: "flex",
        justifyContent: "center",
        alignItems: "center",
      }}
    >
      <CircularProgress />
    </Box>
  ) : (
    messages.map((message, i) => (
      <Box key={i}>
        {isNewDayMessage(messages, i) && (
          <Box
            sx={{
              display: "flex",
              justifyContent: "center",
              textAlign: "center",
            }}
          >
            <Typography
              sx={{
                width: "max-content",
                background: "#0002",
                borderRadius: 4,
                px: 2,
                py: 1,
                mb: "2px",
              }}
            >
              {formatDate(message.createdAt)}
            </Typography>
          </Box>
        )}
        {currentChat.isGroupChat ? (
          <GroupChatMessage
            key={message._id}
            message={message}
            user={user}
            messages={messages}
            setEditMessage={setEditMessage}
            index={i}
          />
        ) : (
          <SingleChatMessage
            message={message}
            user={user}

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				94
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        editMessage={editMessage}
        setEditMessage={setEditMessage}
      />
    ))
  </Box>
))
})
</ChatContent>

{editMessage ? (
  <UpdateMessageInput
    editMessage={editMessage}
    setEditMessage={setEditMessage}
    updatedMessage={updatedMessage}
    updateMessage={updateMessage}
    handleUpdateInput={handleUpdateInput}
  />
) : (
  <CreateMessageInput
    sendMessage={sendMessage}
    typingHandler={typingHandler}
    newMessage={newMessage}
  />
)}
</ChatBoxContainer>
</>
);
};

export default ChatBox;

```

packages/frontend/src/components/ChatBox/CreateMessageInput.jsx:

```

import { Box, TextField, IconButton } from "@mui/material";
import SendIcon from "@mui/icons-material/Send";
import AttachFileIcon from "@mui/icons-material/AttachFile";
import React, { useRef, useState } from "react";
import axios from "../../utils/axios.js";
import Attachment from "../../Attachment/Attachment";

const CreateMessageInput = ({ sendMessage, typingHandler, newMessage }) => {
  const [attachments, setAttachments] = useState([]);
  const [errorMessage, setErrorMessage] = useState();
  const fileInputRef = useRef(null);

  const openFileInput = () => {
    fileInputRef.current.click();
  };

  const handleUpload = async (event) => {
    const file = event.target.files[0];
    if (file.size > 10485760) {
      setErrorMessage("Файл є за великий. Максимальний розмір - 10MB");
    } else {
      const formData = new FormData();
      formData.append("file", file);

      try {
        const response = await axios.post("/upload", formData);
        setAttachments((prev) => [...prev, response.data]);
      } catch (error) {
        setErrorMessage("Помилка завантаження файлу. Спробуйте пізніше");
      }
    }
  };

  const deleteAttachment = (attachment) => {
    setAttachments((prev) =>
      prev.filter((prevAttachment) => attachment.url !== prevAttachment.url)
    );
  };

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    );
  };

  const send = () => {
    sendMessage(attachments);
    setAttachments([]);
  };

  return (
    <Box
      sx={{
        borderTop: "1px solid #ccc",
        padding: "16px",
        display: "flex",
        flexDirection: "column",
      }}
    >
      {attachments.length > 0 && (
        <Box sx={{ display: "flex", flexWrap: "wrap", pb: "8px" }}>
          {attachments.map((attachment) => (
            <Attachment
              key={attachment.url}
              attachment={attachment}
              onDelete={() => deleteAttachment(attachment)}
            />
          ))}
        </Box>
      )}
      <Box
        sx={{
          display: "flex",
          alignItems: "center",
        }}
      >
        <input
          ref={fileInputRef}
          type="file"
          style={{ display: "none" }}
          onChange={handleUpload}
        />
        <IconButton sx={{ mr: 1 }} onClick={openFileInput}>
          <AttachFileIcon />
        </IconButton>
        <TextField
          fullWidth
          variant="outlined"
          label="Введіть ваше повідомлення..."
          autoComplete="off"
          onKeyDown={(e) => e.key === "Enter" && send()}
          onChange={typingHandler}
          value={newMessage}
        />
        <IconButton sx={{ ml: 1 }} onClick={send}>
          <SendIcon />
        </IconButton>
      </Box>
    </Box>
  );
};

```

export default CreateMessageInput;

packages/frontend/src/components/ChatBox/GroupChatMessage.jsx:

```

import React, { useEffect } from "react";
import { Box, Typography, Avatar } from "@mui/material";
import { isLastMessage, isSameSender } from "../../utils/messages";
import { formatTime } from "../../utils/datetime";
import { useSocket } from "../../contexts/SocketContext";

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				96
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import MessageStatus from "../MessageStatus/MessageStatus";
import MessageMenu from "../MessageMenu";
import Attachment from "../Attachment/Attachment";
import { decryptMessage } from "../../utils/encryption";

const GroupChatMessage = ({ message, user, messages, index, setEditMessage }) => {
  const { socket, setNotification } = useSocket();

  useEffect(() => {
    if (message.sender._id !== user._id && message.status !== "read") {
      socket.emit("markAsRead", { messageId: message._id, userId: user._id });
      setNotification((prev) =>
        prev.filter((prevMessage) => message._id !== prevMessage._id)
      );
    }
  }, [socket]);

  const showAvatar =
    message.sender._id !== user._id && isLastMessage(messages, index);
  const showUserName =
    message.sender._id !== user._id && !isSameSender(messages, index);

  return (
    <Box
      sx={{
        display: "flex",
        justifyContent:
          message.sender._id === user._id ? "flex-end" : "flex-start",
        alignItems: "flex-end",
        mt: showUserName && 1,
      }}
    >
      {showAvatar && (
        <Avatar
          alt={` ${message.sender.firstName} ${message.sender.lastName} `}
          src={message.sender.avatar}
          sx={{ background: "gray", color: "white", mb: "2px", mr: "10px" }}
        />
      )}
      <Box
        sx={{
          width: "max-content",
          p: 1.5,
          borderRadius: 2,
          mb: "2px",
          maxWidth: "70%",
          wordWrap: "break-word",
          ml: !showAvatar && "50px",
        }}
        bgcolor={
          message.sender._id === user._id ? "primary.main" : "secondary.main"
        }
        color={
          message.sender._id !== user._id ? "primary.main" : "secondary.main"
        }
      >
        <MessageMenu message={message} user={user} setEditMessage={setEditMessage}>
          {showUserName && (
            <Typography
              variant="overline"
              sx={{ lineHeight: 1, fontWeight: "bold", mb: "6px" }}
            >
              {message.sender.firstName} {message.sender.lastName}
            </Typography>
          )}
          {message.attachments.length > 0 && (
            <Box>
              {message.attachments.map((attachment) => (
                <Attachment

```

```

        key={attachment.url}
        attachment={attachment}
        openFile={true}
        theme={message.sender._id === user._id ? "dark" : "light"}
      />
    )}}
  </Box>
)}
<Typography
  variant="body1"
  sx={{ lineHeight: 1, mt: showUserName && 1 }}
>
  {decryptMessage(message.text, message.chat._id)}
</Typography>
<Box
  sx={{
    display: "flex",
    justifyContent: "space-between",
    alignItems: "end",
    mt: "2px",
  }}
>
  <Typography
    variant="overline"
    sx={{
      lineHeight: 1,
      opacity: "50%",
      mt: 0.5,
      display: "flex",
      justifyContent:
        message.sender._id !== user._id ? "flex-end" : "flex-start",
    }}
  >
    {formatTime(message.createdAt)}
  </Typography>
  {message.sender._id === user._id && (
    <MessageStatus status={message.status} />
  )}
</Box>
</MessageMenu>
</Box>
</Box>
);
};

export default GroupChatMessage;

```

packages/frontend/src/components/ChatBox/MessageMenu.jsx:

```

import React, { useEffect, useRef, useState } from "react";
import FileDownloadIcon from '@mui/icons-material/FileDownload';
import DeleteIcon from '@mui/icons-material/Delete';
import EditIcon from '@mui/icons-material/Edit';
import { Box, Menu, MenuItem, Typography } from '@mui/material';
import { useSocket } from "../../contexts/SocketContext";
import axios from "../../utils/axios";

const MessageMenu = ({ message, user, children, setEditMessage }) => {
  const { socket } = useSocket();
  const [anchorEl, setAnchorEl] = useState(null);
  const menuRef = useRef(null);

  const handleMenuOpen = (event) => {
    setAnchorEl(event.currentTarget);
  };

  const handleMenuClose = () => {
    setAnchorEl(null);
  };

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const handleEditMessage = () => {
  setEditMessage(message);
};

const handleDeleteMessage = () => {
  socket.emit("deleteMessage", message._id);
};

useEffect(() => {
  const handleClickOutside = (event) => {
    if (menuRef.current && !menuRef.current.contains(event.target)) {
      setAnchorEl(null);
    }
  };

  document.addEventListener("click", handleClickOutside);

  return () => {
    document.removeEventListener("click", handleClickOutside);
  };
}, []);

const handleDownload = async () => {
  if (message.attachments.length > 0) {
    for (const attachment of message.attachments) {
      try {
        const response = await axios.get(attachment.url, {
          responseType: 'blob',
        });
        const url = window.URL.createObjectURL(new Blob([response.data]));
        const link = document.createElement('a');
        link.href = url;
        link.setAttribute('download', attachment.filename);
        document.body.appendChild(link);
        link.click();
        document.body.removeChild(link);
        window.URL.revokeObjectURL(url);
      } catch (error) {
        console.error('Download error:', error);
      }
    }
  }
};

return (
  <Box ref={menuRef} onClick={handleMenuOpen}>
    <Menu
      anchorEl={anchorEl}
      open={Boolean(anchorEl)}
      onClose={handleMenuClose}
    >
      {message.sender._id === user._id && (
        [
          message.attachments.length > 0 && (
            <MenuItem key="download" onClick={handleDownload}>
              <FileDownloadIcon sx={{ width: 24, height: 24, mr: 1 }} />
              <Typography>Завантажити</Typography>
            </MenuItem>
          ),
          <MenuItem key="edit" onClick={handleEditMessage}>
            <EditIcon sx={{ width: 24, height: 24, mr: 1 }} />
            <Typography>Редагувати</Typography>
          </MenuItem>,
          <MenuItem key="delete" onClick={handleDeleteMessage}>
            <DeleteIcon sx={{ width: 24, height: 24, mr: 1 }} />
            <Typography>Видалити</Typography>
          </MenuItem>
        ]
      )}
    </Box>
  )
);

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				99
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    ))
  </Menu>
  {children}
</Box>
);
};

export default MessageMenu;

```

packages/frontend/src/components/ChatBox/SingleChatMessage.jsx:

```

import React, { useEffect } from "react";
import { formatTime } from "../../utils/datetime";
import { Box, Typography } from "@mui/material";
import { useSocket } from "../../contexts/SocketContext";
import MessageStatus from "../../MessageStatus/MessageStatus";
import MessageMenu from "../../MessageMenu";
import Attachment from "../../Attachment/Attachment";
import { decryptMessage } from "../../utils/encryption";

const SingleChatMessage = ({ message, user, editMessage, setEditMessage }) => {
  const { socket, setNotification } = useSocket();

  useEffect(() => {
    if (message._id) {
      if (message.sender._id !== user._id && message.status !== "read") {
        socket.emit("markAsRead", { messageId: message._id, userId: user._id });
        setNotification((prev) =>
          prev.filter((prevMessage) => message._id !== prevMessage._id)
        );
      }
    }
  }, [socket]);

  return (
    <Box
      key={message._id}
      sx={{
        display: "flex",
        mb: "2px",
        justifyContent:
          message.sender._id === user._id ? "flex-end" : "flex-start",
        background: editMessage && editMessage._id === message._id && "#0002",
        borderRadius: 2,
      }}
    >
      <Box
        sx={{
          width: "max-content",
          p: 1.5,
          borderRadius: 2,
          maxWidth: "70%",
          wordWrap: "break-word",
        }}
        bgcolor={
          message.sender._id === user._id ? "primary.main" : "secondary.main"
        }
        color={
          message.sender._id !== user._id ? "primary.main" : "secondary.main"
        }
      >
        <MessageMenu
          message={message}
          user={user}
          setEditMessage={setEditMessage}
        >
          <Box>
            {message.attachments.length > 0 && (
              <Box>

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				100
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {message.attachments.map((attachment) => (
          <Attachment
            key={attachment.url}
            attachment={attachment}
            openFile={true}
            theme={message.sender._id === user._id ? "dark" : "light"}
          />
        ))}
      </Box>
    )}
    <Typography variant="body1" sx={{ lineHeight: 1 }}>
      {decryptMessage(message.text, message.chat._id)}
    </Typography>
    <Box
      sx={{
        display: "flex",
        justifyContent: "space-between",
        alignItems: "end",
        mt: "2px",
      }}
    >
      <Typography
        variant="overline"
        sx={{ lineHeight: 1, opacity: "50%" }}
      >
        {formatTime(message.createdAt)}
      </Typography>
      {message.sender._id === user._id && (
        <MessageStatus status={message.status} />
      )}
    </Box>
  </Box>
</MessageMenu>
</Box>
</Box>
);
};

export default SingleChatMessage;

```

packages/frontend/src/components/ChatBox/UpdateMessageInput.jsx:

```

import { Box, Button, TextField, Typography, IconButton } from "@mui/material";
import React, { useEffect, useRef, useState } from "react";
import CloseIcon from "@mui/icons-material/Close";
import AttachFileIcon from "@mui/icons-material/AttachFile";
import SendIcon from "@mui/icons-material/Send";
import axios from "../../utils/axios.js";
import Attachment from "../../Attachment/Attachment";
import { decryptMessage } from "../../utils/encryption.js";

const UpdateMessageInput = ({
  editMessage, // show input
  setEditMessage, // set show edit input
  updatedMessage, // message
  updateMessage, // submit function
  handleUpdateInput, // onChange function
}) => {
  const [attachments, setAttachments] = useState([]);
  const [errorMessage, setErrorMessage] = useState("");
  const fileInputRef = useRef(null);

  useEffect(() => {
    setAttachments(editMessage.attachments);
  }, [editMessage]);

  const openFileInput = () => {
    fileInputRef.current.click();
  };
};

```

```

const handleUpload = async (event) => {
  const file = event.target.files[0];
  if (file.size > 10485760) {
    setErrorMessage("Файл є за великий. Максимальний розмір - 10MB");
  } else {
    const formData = new FormData();
    formData.append("file", file);

    try {
      const response = await axios.post("/upload", formData);
      setAttachments((prev) => [...prev, response.data]);
    } catch (error) {
      setErrorMessage("Помилка завантаження файлу. Спробуйте пізніше");
    }
  }
};

const deleteAttachment = (attachment) => {
  setAttachments((prev) =>
    prev.filter((prevAttachment) => attachment.url !== prevAttachment.url)
  );
};

const update = () => {
  updateMessage(attachments);
};

return (
  <Box
    sx={{
      borderTop: "1px solid #ccc",
      padding: "16px",
      display: "flex",
      flexDirection: "column",
    }}
  >
    {attachments.length > 0 && (
      <Box
        sx={{
          display: "flex",
          flexDirection: "row",
          flexWrap: "wrap",
        }}
      >
        {attachments.map((attachment) => (
          <Attachment
            key={attachment.url}
            attachment={attachment}
            onDelete={() => deleteAttachment(attachment)}
          />
        ))}
      </Box>
    )}
    <Box
      sx={{
        display: "flex",
        justifyContent: "space-between",
        flexDirection: "row",
        width: "100%",
        alignItems: "center",
        pb: 1,
      }}
    >
      <Typography sx={{ ml: 1 }}>
        {decryptMessage(editMessage.text, editMessage.chat._id)}
      </Typography>
      <Button onClick={() => setEditMessage(null)}>
      <CloseIcon />
    </Box>
  </Box>
);

```

		Іщенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				102
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        </Button>
      </Box>
      <Box sx={{ display: "flex", width: "100%" }}>
        <input
          ref={fileInputRef}
          type="file"
          style={{ display: "none" }}
          onChange={handleUpload}
        />
        <IconButton sx={{ mr: 1 }} onClick={openFileInput}>
          <AttachFileIcon />
        </IconButton>
        <TextField
          fullWidth
          variant="outlined"
          autoComplete="off"
          onChange={handleUpdateInput}
          onKeyDown={(e) => e.key === "Enter" && update()}
          value={updatedMessage}
        />
        <IconButton sx={{ ml: 1 }} onClick={update}>
          <SendIcon />
        </IconButton>
      </Box>
    </Box>
  );
};

```

export default UpdateMessageInput;

packages/frontend/src/components/Header/Header.jsx:

```

import React, { useState } from "react";
import {
  AppBar,
  Toolbar,
  Typography,
  IconButton,
  Menu,
  MenuItem,
  Avatar,
  Divider,
} from "@mui/material";
import { styled } from "@mui/system";
import AddIcon from "@mui/icons-material/Add";
import SettingsIcon from "@mui/icons-material/Settings";
import ExitToAppIcon from "@mui/icons-material/ExitToApp";
import { useAuth } from "../../../contexts/AuthContext";
import SearchInput from "../../../Inputs/SearchInput";
import CreateGroupModal from "../../../Modals/Group/CreateGroupModal";
import { useSocket } from "../../../contexts/SocketContext";
import SettingsModal from "../../../Modals/Settings/SettingsModal";

const HeaderAppBar = styled(AppBar)({
  zIndex: (theme) => theme.zIndex.drawer + 1,
  height: "64px",
});

const HeaderToolbar = styled(Toolbar)({
  justifyContent: "space-between",
});

const AvatarIconButton = styled(IconButton)({
  cursor: "pointer",
  "& .MuiIconButton-label": {
    fontSize: "1.5rem",
  },
});

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				103
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const UserInfomenuItem = styled("div")(({ theme }) => ({
  padding: theme.spacing(2),
  display: "flex",
  justifyContent: "center",
  flexDirection: "column",
  alignItems: "center",
}));

const MenuItemWrapper = styled(MenuItem)(({ theme }) => ({
  display: "flex",
  alignItems: "center",
  gap: theme.spacing(1),
  minWidth: 260,
}));

const Header = ({ user, setCurrentChat, setUpdateChats, contacts }) => {
  const [anchorEl, setAnchorEl] = useState(null);
  const [createGroupModalIsOpen, setCreateGroupModalIsOpen] = useState(false);
  const [settingsModalIsOpen, setSettingsModalIsOpen] = useState(false);
  const { socket } = useSocket();
  const { logout } = useAuth();

  const handleMenuOpen = (event) => {
    setAnchorEl(event.currentTarget);
  };

  const handleMenuClose = () => {
    setAnchorEl(null);
  };

  const handleLogout = () => {
    logout();
    handleMenuClose();
    socket.emit("logout", user._id);
  };

  const openCreateGroupModal = () => {
    setCreateGroupModalIsOpen(true);
  };

  const closeCreateGroupModal = () => {
    setCreateGroupModalIsOpen(false);
  };

  const openSettingsModal = () => {
    setSettingsModalIsOpen(true);
  };

  const closeSettingsModal = () => {
    setSettingsModalIsOpen(false);
  };

  return (
    <>
      {createGroupModalIsOpen && (
        <CreateGroupModal
          isOpen={createGroupModalIsOpen}
          handleClose={closeCreateGroupModal}
          contacts={contacts}
          user={user}
          setUpdateChats={setUpdateChats}
        />
      )}
      {settingsModalIsOpen && (
        <SettingsModal
          open={settingsModalIsOpen}
          onClose={closeSettingsModal}
          user={user}
          setUpdateChats={setUpdateChats}

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				104
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    />
  })
  <HeaderAppBar position="fixed">
    <HeaderToolbar>
      <Typography variant="h6">WebChat</Typography>
      <SearchInput
        setCurrentChat={setCurrentChat}
        setUpdateChats={setUpdateChats}
      />
      <AvatarIconButton
        edge="end"
        aria-label="account of current user"
        aria-haspopup="true"
        onClick={handleMenuOpen}
        color="inherit"
      >
        <Avatar
          sx={{ background: "gray", color: "white" }}
          alt={` ${user.firstName} ${user.lastName}`}
          src={user.avatar}
        />
      </AvatarIconButton>
      <Menu
        anchorEl={anchorEl}
        open={Boolean(anchorEl)}
        onClose={handleMenuClose}
      >
        <UserInfoMenuItem>
          <Avatar
            alt={` ${user.firstName} ${user.lastName}`}
            src={user.avatar}
            sx={{
              width: 64,
              height: 64,
              background: "gray",
              color: "white",
            }}
          />
          <Typography
            variant="h6"
            sx={{ mt: 1 }}
          >` ${user.firstName} ${user.lastName}`</Typography>
          <Typography variant="subtitle1">{user.email}</Typography>
        </UserInfoMenuItem>
        <Divider />
        <MenuItemWrapper onClick={openCreateGroupModal}>
          <AddIcon />
          Створити нову групу
        </MenuItemWrapper>
        <MenuItemWrapper onClick={openSettingsModal}>
          <SettingsIcon />
          Налаштування
        </MenuItemWrapper>
        <MenuItemWrapper onClick={handleLogout}>
          <ExitToAppIcon />
          Вийти
        </MenuItemWrapper>
      </Menu>
    </HeaderToolbar>
  </HeaderAppBar>
</>
  );
};

export default Header;

```

packages/frontend/src/components/Inputs/AddUserSearchInput.jsx:

```

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

```

```

import {
  CircularProgress,
  Avatar,
  Typography,
  Box,
  TextField,
} from "@mui/material";
import SearchIcon from "@mui/icons-material/Search";
import { styled } from "@mui/system";
import axios from "../../utils/axios.js";
import { useAuth } from "../../contexts/AuthContext";

const SearchContainer = styled("div")({
  position: "relative",
  width: "100%",
  display: "flex",
  alignItems: "center",
  mt: 2,
});

const SearchIconContainer = styled("div")(({ theme }) => ({
  padding: theme.spacing(0, 2),
  height: "100%",
  position: "absolute",
  pointerEvents: "none",
  display: "flex",
  alignItems: "center",
  justifyContent: "center",
  color: theme.palette.primary.main,
})));

const SearchInputBase = styled(TextField)(({ theme }) => ({
  padding: "2px",
  paddingLeft: `calc(1em + ${theme.spacing(4)})`,
  transition: theme.transitions.create("width"),
  width: "100%",
  color: theme.palette.secondary.main,
})));

const LoadingSpinner = styled(CircularProgress)(({ theme }) => ({
  position: "absolute",
  right: 0,
  top: "50%",
  transform: "translateY(-50%)",
  marginRight: theme.spacing(2),
  color: theme.palette.primary.main,
  marginTop: "-10px",
})));

const MenuItem = styled("div")({
  display: "flex",
  alignItems: "center",
  padding: "8px",
  backgroundColor: "white",
  boxShadow: "0px 4px 10px rgba(0, 0, 0, 0.1)",
  borderRadius: "4px",
  marginBottom: "4px",
  color: "#000000",
});

const AddUserSearchInput = ({ onSubmit, selectedContacts }) => {
  const [searchQuery, setSearchQuery] = useState("");
  const [loading, setLoading] = useState(false);
  const [searchResults, setSearchResults] = useState([]);
  const [isOpen, setIsOpen] = useState(false);
  const { token } = useAuth();
  const inputRef = useRef(null);

  useEffect(() => {

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				106
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const delayDebounceFn = setTimeout(async () => {
  if (searchQuery.trim() !== "") {
    setLoading(true);
    try {
      const response = await axios.get(`/user/${searchQuery}`, {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      });

      const filteredResults = response.data.filter(
        (result) =>
          !selectedContacts.some((contact) => contact._id === result._id)
      );

      setSearchResults(filteredResults);
      setLoading(false);
      setIsOpen(true);
    } catch (error) {
      console.error("Error fetching search results", error);
      setLoading(false);
    }
  } else {
    setSearchResults([]);
    setLoading(false);
    setIsOpen(false);
  }
}, 1000);

return () => clearTimeout(delayDebounceFn);
}, [searchQuery, token, selectedContacts]);

const handleInputChange = (event) => {
  setLoading(!event.target.value);
  setSearchQuery(event.target.value);
  setSearchResults([]);
};

const handleClick = (contact) => {
  setIsOpen(false);
  setSearchQuery("");
  onSubmit(contact);
};

return (
  <SearchContainer>
    <SearchIconContainer>
      <SearchIcon />
    </SearchIconContainer>

    <SearchInputBase
      autoComplete="off"
      placeholder="Пошук контактів..."
      inputProps={{ "aria-label": "search" }}
      onChange={handleInputChange}
      inputRef={inputRef}
    />

    {loading && <LoadingSpinner size={20} />}

    {isOpen && (
      <Box
        style={{
          position: "absolute",
          top: "100%",
          width: "100%",
          zIndex: 999,
          padding: "4px",
        }}
      >

```

```

    >
    {searchResults.length > 0
      ? searchResults.map((user) => (
        <MenuItem key={user._id} onClick={() => handleClick(user)}>
          <Avatar
            alt={` ${user.firstName} ${user.lastName}`}
            src={user.avatar}
            sx={{
              marginRight: "8px",
              background: "gray",
              color: "white",
            }}
          />
          <div>
            <Typography>`${user.firstName} ${user.lastName}`</Typography>
            <Typography>{user.email}</Typography>
          </div>
        </MenuItem>
      ))
      : !loading &&
        searchQuery && (
          <MenuItem>
            <Typography>Не знайдено такого контакту</Typography>
          </MenuItem>
        )
    }
  </Box>
}
</SearchContainer>
);
};

```

export default AddUserSearchInput;

packages/frontend/src/components/Inputs/SearchInput.jsx:

```

import React, { useState, useEffect, useRef } from "react";
import {
  InputBase,
  CircularProgress,
  Avatar,
  Typography,
  Box,
} from "@mui/material";
import SearchIcon from "@mui/icons-material/Search";
import { styled } from "@mui/system";
import axios from "../../../utils/axios.js";
import { useAuth } from "../../../contexts/AuthContext";

```

```

const SearchContainer = styled("div")(({ theme }) => ({
  position: "relative",
  borderRadius: theme.shape.borderRadius * 3,
  backgroundColor: theme.palette.primary.main,
  border: `2px solid ${theme.palette.secondary.main}`,
  "&:hover": {
    backgroundColor: theme.palette.primary.main,
  },
  marginLeft: theme.spacing(2),
  marginRight: theme.spacing(1),
  width: "100%",
  [theme.breakpoints.up("sm")]: {
    width: "300px",
    marginLeft: 0,
    marginRight: 0,
  },
  display: "flex",
  alignItems: "center",
}));

```

```

const SearchIconContainer = styled("div")(({ theme }) => ({

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				108
Змн.	Арк.	№ докум.	Підпис	Дата		

```

padding: theme.spacing(0, 2),
height: "100%",
position: "absolute",
pointerEvents: "none",
display: "flex",
alignItems: "center",
justifyContent: "center",
color: theme.palette.secondary.main,
));

const SearchInputBase = styled(InputBase)(({ theme }) => ({
padding: "2px",
paddingLeft: `calc(1em + ${theme.spacing(4)})`,
transition: theme.transitions.create("width"),
width: "100%",
color: theme.palette.secondary.main,
}));

const LoadingSpinner = styled(CircularProgress)(({ theme }) => ({
position: "absolute",
right: 0,
top: "50%",
transform: "translateY(-50%)",
marginRight: theme.spacing(2),
color: theme.palette.secondary.main,
marginTop: "-10px",
}));

const MenuItem = styled("div")({
display: "flex",
alignItems: "center",
padding: "8px",
backgroundColor: "white",
boxShadow: "0px 4px 10px rgba(0, 0, 0, 0.1)",
borderRadius: "4px",
marginBottom: "4px",
color: "#000000",
zIndex: "200",
});

const SearchInput = ({ setCurrentChat, setUpdateChats }) => {
const [searchQuery, setSearchQuery] = useState("");
const [loading, setLoading] = useState(false);
const [searchResults, setSearchResults] = useState([]);
const [isOpen, setIsOpen] = useState(false);
const { token } = useAuth();
const inputRef = useRef(null);

useEffect(() => {
const delayDebounceFn = setTimeout(async () => {
if (searchQuery.trim() !== "") {
setLoading(true);
try {
const response = await axios.get(`/user/${searchQuery}`, {
headers: {
Authorization: `Bearer ${token}`,
},
});
setSearchResults(response.data);
setLoading(false);
setIsOpen(true);
} catch (error) {
console.error("Error fetching search results", error);
setLoading(false);
}
} else {
setSearchResults([]);
setLoading(false);
setIsOpen(false);
}
});
}, [searchQuery, token]);

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				109
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }, 1000);

    return () => clearTimeout(delayDebounceFn);
  }, [searchQuery, token]);

const handleInputChange = (event) => {
  setLoading(!event.target.value);
  setSearchQuery(event.target.value);
  setSearchResults([]);
};

const createChat = async (userId) => {
  try {
    const response = await axios.post(
      "/chat",
      {
        userId,
        headers: {
          Authorization: `Bearer ${token}`,
        },
      },
    );
    setIsOpen(false);
    setCurrentChat(response.data);
    setUpdateChats(response.data);
  } catch (error) {}
};

return (
  <SearchContainer>
    <SearchIconContainer>
      <SearchIcon />
    </SearchIconContainer>

    <SearchInputBase
      placeholder="Пошук контактів..."
      inputProps={{ "aria-label": "search" }}
      onChange={handleInputChange}
      inputRef={inputRef}
    />
    {loading && <LoadingSpinner size={20} />}
    {isOpen && (
      <Box
        style={{
          position: "absolute",
          top: "100%",
          width: "100%",
          zIndex: 999,
          padding: "4px",
        }}
      >
        {searchResults.length > 0
          ? searchResults.map((user) => (
              <MenuItem key={user._id} onClick={() => createChat(user._id)}>
                <Avatar
                  alt={` ${user.firstName} ${user.lastName}`}
                  src={user.avatar}
                  sx={{
                    marginRight: "8px",
                    background: "gray",
                    color: "white",
                  }}
                />
                <div>
                  <Typography>` ${user.firstName} ${user.lastName}`</Typography>
                  <Typography>{user.email}</Typography>
                </div>
              </MenuItem>
            ))
          : null}
      )}
    )}
  )
);

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				110
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    ))
    : !loading &&
      searchQuery && (
        <MenuItem>
          <Typography>Не знайдено такого контакту</Typography>
        </MenuItem>
      )
    </Box>
  )
</SearchContainer>
);
};

export default SearchInput;

```

packages/frontend/src/components/MessageStatus/MessageStatus.jsx:

```

import React from "react";
import AccessTimeIcon from "@mui/icons-material/AccessTime";
import DoneIcon from "@mui/icons-material/Done";
import DoneAllIcon from "@mui/icons-material/DoneAll";

const MessageStatus = ({ status }) => {
  const style = {
    width: "16px",
    height: "16px",
    ml: 1,
    opacity: 0.5,
  };
  switch (status) {
    case "sending":
      return <AccessTimeIcon sx={style} />;
    case "sent":
      return <DoneIcon sx={style} />;
    case "read":
      return <DoneAllIcon sx={style} />;
  }
};

export default MessageStatus;

```

packages/frontend/src/components/Modals/ModalBox.jsx:

```

import styled from "@emotion/styled";
import { Box } from "@mui/material";

const ModalBox = styled(Box)(({ theme }) => ({
  position: "absolute",
  top: "50%",
  left: "50%",
  transform: "translate(-50%, -50%)",
  backgroundColor: theme.palette.background.paper,
  boxShadow: theme.shadows[24],
  padding: theme.spacing(4),
  minWidth: 420,
  borderRadius: "24px",
  maxHeight: 600,
  overflowY: "scroll",
  "&::-webkit-scrollbar": {
    display: "none",
  },
})));

export default ModalBox;

```

packages/frontend/src/components/Modals/Chat/ChatInfoModal.jsx:

```

import React from "react";
import axios from "../../../utils/axios.js";
import styled from "@emotion/styled";
import { Avatar, Box, Button, Modal, Typography } from "@mui/material";

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		111

```

import DeleteOutlineIcon from "@mui/icons-material/DeleteOutline";

import { getChatInfo } from "../../utils/chat";
import ModalBox from "../../ModalBox";
import { useAuth } from "../../contexts/AuthContext";

const Column = styled("div")(({ theme }) => ({
  padding: theme.spacing(2),
  display: "flex",
  justifyContent: "center",
  flexDirection: "column",
  alignItems: "center",
})));

const ChatInfoModal = ({
  open,
  close,
  chat,
  user,
  setCurrentChat,
  setUpdateChats,
}) => {
  const info = getChatInfo(chat, user);
  const { token } = useAuth();

  const deleteChat = async () => {
    try {
      const response = await axios.delete(`/chat/${chat._id}`, {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      });
      chat._id === response.data._id && setCurrentChat({});
      setUpdateChats(response);
      close();
    } catch (error) {}
  };

  return (
    <Modal open={open} onClose={close}>
      <ModalBox>
        <Column>
          <Avatar
            alt={info.title}
            src={info.avatar}
            sx={{
              width: 100,
              height: 100,
              background: "gray",
              color: "white",
            }}
          />
          <Typography variant="h6" sx={{ mt: 1 }}>
            {info.title}
          </Typography>
          <Typography variant="subtitle1">{info.email}</Typography>
          <Box
            sx={{
              display: "flex",
              justifyContent: "flex-end",
              mt: 2,
              width: "100%",
            }}
          >
            <Button onClick={close} variant="outlined" color="primary">
              Закрити
            </Button>
            <Button
              variant="contained"

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				112
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        color="error"
        onClick={deleteChat}
        startIcon={<DeleteOutlineIcon />}
        sx={{ ml: 2 }}
      >
        Видалити
      </Button>
    </Box>
  </Column>
</ModalBox>
</Modal>
);
};

export default ChatInfoModal;

```

packages/frontend/src/components/Modals/Group/CreateGroupModal.jsx:

```

import React, { useState } from "react";
import axios from "../../../utils/axios.js";
import { useAuth } from "../../../contexts/AuthContext";
import { useForm, Controller } from "react-hook-form";
import {
  TextField,
  Button,
  Typography,
  Stack,
  Alert,
  Avatar,
  Modal,
  Box,
  CircularProgress,
  Menu,
  MenuItem,
} from "@mui/material";
import GroupsIcon from "@mui/icons-material/Groups";
import CloseIcon from "@mui/icons-material/Close";
import MoreVertIcon from "@mui/icons-material/MoreVert";
import ModalBox from "../../../ModalBox";
import AddUserSearchInput from "../../../Inputs/AddUserSearchInput";

const CreateGroupModal = ({
  isOpen,
  handleClose,
  contacts,
  user,
  setUpdateChats,
}) => {
  const { control, handleSubmit } = useForm();
  const [errorMessage, setErrorMessage] = useState("");
  const [avatarUrl, setAvatarUrl] = useState(null);
  const [isLoading, setIsLoading] = useState(false);
  const [groupAdmins, setGroupAdmins] = useState([user]);
  const [selectedContacts, setSelectedContacts] = useState([]);
  const [anchorEl, setAnchorEl] = useState(null);
  const [anchorElTitle, setAnchorElTitle] = useState(null);
  const { token } = useAuth();

  const handleCreateGroup = async (data) => {
    setIsLoading(true);
    try {
      const response = await axios.post(
        "/chat/group",
        {
          ...data,
          groupAdmins: groupAdmins,
          avatar: avatarUrl,
          users: selectedContacts,
        },
      ),

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		113

```

    {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    }
  );
  setUpdateChats(response.data);
  handleClose();
} catch (error) {
  setErrorMessage("Помилка створення групи. Спробуй пізніше");
  console.error("Error creating group:", error);
}
setIsLoading(false);
};

const handleAvatarUpload = async (event) => {
  setErrorMessage("");
  const file = event.target.files[0];
  if (file.type === "image/jpeg" || file.type === "image/png") {
    if (file.size > 10485760) {
      setErrorMessage("Файл є за великий. Максимальний розмір - 10MB");
    } else {
      const formData = new FormData();
      formData.append("file", file);

      try {
        const response = await axios.post("/upload", formData);
        setAvatarUrl(response.data.url);
      } catch (error) {
        setErrorMessage(
          "Помилка завантаження файлу. Спробуйте пізніше"
        );
      }
    }
  } else {
    setErrorMessage("Це має бути зображення");
  }
};

const addUser = (contact) => {
  setSelectedContacts((prevContacts) => [...prevContacts, contact]);
};

const deleteUser = (userId) => {
  setSelectedContacts((prevContacts) =>
    prevContacts.filter((contact) => contact._id !== userId)
  );
};

const handleMakeAdmin = (user) => {
  setGroupAdmins((prevAdmins) => [...prevAdmins, user]);
  setSelectedContacts((prevContacts) =>
    prevContacts.filter((contact) => contact._id !== user._id)
  );
  setAnchorEl(null);
  setAnchorElTitle(null);
};

const handleRemoveAdmin = (user) => {
  setGroupAdmins((prevAdmins) =>
    prevAdmins.filter((admin) => admin._id !== user._id)
  );
  setSelectedContacts((prevContacts) => [...prevContacts, user]);
  setAnchorEl(null);
  setAnchorElTitle(null);
};

const handleMenuOpen = (event, isAdmin) => {
  setAnchorEl(event.currentTarget);
};

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				114
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    setAnchorElTitle(isAdmin ? "admin" : "user");
  };

  const handleMenuClose = () => {
    setAnchorEl(null);
    setAnchorElTitle(null);
  };

  return (
    <Modal open={isOpen} onClose={handleClose}>
      <ModalBox>
        <Typography component="h2" variant="h5">
          Створити групи
        </Typography>
        <form onSubmit={handleSubmit(handleCreateGroup)}>
          <Stack direction="column" alignItems="center" spacing={2}>
            <label htmlFor="avatar-upload" style={{ display: "none" }}>
              <input
                id="avatar-upload"
                name="avatar-upload"
                type="file"
                accept="image/png, image/jpeg"
                style={{ display: "none" }}
                onChange={handleAvatarUpload}
              />
            </label>
            <Avatar
              alt="Group Avatar"
              src={avatarUrl || ""}
              sx={{
                width: 60,
                height: 60,
                cursor: "pointer",
                background: "gray",
                color: "white",
              }}
              onClick={() => document.getElementById("avatar-upload").click()}
            >
              <GroupsIcon sx={{ width: 40, height: 40 }} />
            </Avatar>
            <input
              type="file"
              accept="image/png, image/jpeg"
              onChange={handleAvatarUpload}
              style={{ display: "none" }}
            />
          </Stack>
          <Controller
            name="title"
            control={control}
            defaultValue=""
            rules={{ required: "Назва групи є обов'язкова" }}
            render={({ field, fieldState }) => (
              <TextField
                {...field}
                variant="outlined"
                label="Назва групи"
                fullWidth
                margin="normal"
                error={!fieldState.error}
                helperText={fieldState.error ? fieldState.error.message : null}
              />
            )
          )
        />

        <Controller
          name="description"
          control={control}
          defaultValue=""

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		115

```

render={({ field }) => (
  <TextField
    {...field}
    variant="outlined"
    label="Опис групи"
    fullWidth
    multiline
    rows={3}
    margin="normal"
  />
)}
/>

<AddUserSearchInput
  onSubmit={addUser}
  selectedContacts={selectedContacts.concat(groupAdmins)}
/>

{groupAdmins.map((admin) => (
  <Box
    key={admin._id}
    sx={{ display: "flex", alignItems: "center", my: 1 }}
  >
    <Avatar
      alt={admin.email}
      src={admin.avatar}
      sx={{ mr: 2, background: "gray", color: "white" }}
    />
    <Box sx={{ flexGrow: 1 }}>
      <Typography>`${admin.firstName} ${admin.lastName}`</Typography>
      <Typography>{admin.email}</Typography>
    </Box>
    <Box sx={{ ml: 2 }}>
      <Typography variant="subtitle1" color="primary">
        Адмін
      </Typography>
    </Box>
    {user._id !== admin._id && (
      <>
        <Button onClick={(e) => handleMenuOpen(e, true)}>
          <MoreVertIcon />
        </Button>
        <Menu
          anchorEl={anchorEl}
          open={Boolean(anchorEl) && anchorElTitle === "admin"}
          onClose={handleMenuClose}
        >
          <MenuItem onClick={() => handleRemoveAdmin(admin)}>
            Звернути права Адміна
          </MenuItem>
        </Menu>
      </>
    )}
  </Box>
)}
)}}

{selectedContacts.map((contact) => (
  <Box
    key={contact._id}
    sx={{ display: "flex", alignItems: "center", my: 1 }}
  >
    <Avatar
      alt={contact.email}
      src={contact.avatar}
      sx={{ mr: 2, background: "gray", color: "white" }}
    />
    <Box sx={{ flexGrow: 1 }}>
      <Typography>`${contact.firstName} ${contact.lastName}`</Typography>
      <Typography>{contact.email}</Typography>
    </Box>
  </Box>
)}

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				116
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        </Box>
        <Button onClick={() => deleteUser(contact._id)}>
          <CloseIcon />
        </Button>
        <Button onClick={(e) => handleMenuOpen(e, false)}>
          <MoreVertIcon />
        </Button>
        <Menu
          anchorEl={anchorEl}
          open={Boolean(anchorEl) && anchorElTitle === "user"}
          onClose={handleMenuClose}
        >
          <MenuItem onClick={() => handleMakeAdmin(contact)}>
            Зробити Адміном
          </MenuItem>
        </Menu>
      </Box>
    )}

    {errorMessage && (
      <Alert severity="error" sx={{ mt: 2 }}>
        {errorMessage}
      </Alert>
    )}
    <Box sx={{ display: "flex", justifyContent: "flex-end", mt: 2 }}>
      <Button
        onClick={handleClose}
        variant="outlined"
        color="primary"
        sx={{ ml: 2 }}
        disabled={isLoading}
      >
        Відмінити
      </Button>
      <Button
        type="submit"
        variant="contained"
        color="primary"
        disabled={isLoading}
        sx={{ ml: 2 }}
      >
        {isLoading ? <CircularProgress size={24} /> : "Створити"}
      </Button>
    </Box>
  </form>
</ModalBox>
</Modal>
);
};

```

export default CreateGroupModal;

packages/frontend/src/components/Modals/Group/GroupInfoModal.jsx:

```

import React from "react";
import { Avatar, Modal, Typography, Button, Box } from "@mui/material";
import GroupsIcon from "@mui/icons-material/Groups";
import axios from "../../../utils/axios.js";
import ModalBox from "../../../ModalBox";
import { getChatInfo } from "../../../utils/chat";
import styled from "@emotion/styled";
import UpdateGroup from "../../../UpdateGroup";
import { useAuth } from "../../../contexts/AuthContext";

```

```

const Column = styled("div")(({ theme }) => ({
  padding: theme.spacing(2),
  display: "flex",
  justifyContent: "center",
  flexDirection: "column",

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		117

```

    alignItems: "center",
  }));

const UserBox = styled("div")(({ theme }) => ({
  display: "flex",
  alignItems: "center",
  marginBottom: theme.spacing(1),
  marginTop: theme.spacing(1),
  width: "100%",
}));

const GroupAvatar = styled(Avatar)(({ theme }) => ({
  marginRight: theme.spacing(2),
  background: "gray",
  color: "white",
  width: 100,
  height: 100,
}));

const GroupInfoModal = ({
  open,
  close,
  group,
  user,
  setUpdateChats,
  setCurrentChat,
}) => {
  const info = getChatInfo(group, user);
  const isAdmin = group.groupAdmins.some((admin) => admin._id === user._id);
  const { token } = useAuth();

  const leaveGroup = async () => {
    try {
      const response = await axios.put(
        "/chat/group/leave",
        { chatId: group._id },
        {
          headers: {
            Authorization: `Bearer ${token}`,
          },
        }
      );
      setUpdateChats(response);
      group._id === response.data._id && setCurrentChat({});
      close();
    } catch (error) {}
  };

  const deleteGroup = async () => {
    try {
      const response = await axios.delete(`/chat/${group._id}`, {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      });
      group._id === response.data._id && setCurrentChat({});
      setUpdateChats(response);
      close();
    } catch (error) {}
  };

  return (
    <Modal open={open} onClose={close}>
      <ModalBox>
        <Column>
          {!isAdmin ? (
            <>
              <GroupAvatar alt={info.title} src={info.avatar}>

```

		Іщенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				118
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <GroupsIcon sx={{ width: 60, height: 60 }} />
      </GroupAvatar>
      <Typography variant="h6" sx={{ mt: 1 }}>
        {info.title}
      </Typography>
      <Typography variant="subtitle1">{info.description}</Typography>
      <Typography variant="h5" sx={{ mt: 2 }}>
        Учасники
      </Typography>
      {group.groupAdmins.map((admin) => (
        <UserBox key={admin._id}>
          <Avatar
            alt={admin.email}
            src={admin.avatar}
            sx={{ mr: 2, background: "gray", color: "white" }}
          />
          <Box sx={{ flexGrow: 1 }}>
            <Typography>`${admin.firstName} ${admin.lastName}`</Typography>
            <Typography>{admin.email}</Typography>
          </Box>
          <Box sx={{ ml: 2 }}>
            <Typography variant="subtitle1" color="primary">
              Адмін
            </Typography>
          </Box>
        </UserBox>
      )))
      {group.users.map((contact) => (
        <UserBox key={contact._id}>
          <Avatar
            alt={contact.email}
            src={contact.avatar}
            sx={{ mr: 2, background: "gray", color: "white" }}
          />
          <Box sx={{ flexGrow: 1 }}>
            <Typography>`${contact.firstName}
${contact.lastName}`</Typography>
            <Typography>{contact.email}</Typography>
          </Box>
        </UserBox>
      )))
    <Box
      sx={{
        display: "flex",
        width: "100%",
        justifyContent: "flex-end",
        mt: 2,
      }}
    >
      <Button
        onClick={close}
        variant="outlined"
        color="primary"
        sx={{ ml: 2 }}
      >
        Закрити
      </Button>
      <Button
        variant="outlined"
        color="error"
        sx={{ ml: 2 }}
        onClick={leaveGroup}
      >
        Вийти з чату
      </Button>
    </Box>
  </>
) : (
  <UpdateGroup

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				119
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        handleClose={close}
        user={user}
        currentGroup={group}
        setUpdateChats={setUpdateChats}
        setCurrentChat={setCurrentChat}
        onLeave={leaveGroup}
        onDelete={deleteGroup}
      />
    )}
  </Column>
</ModalBox>
</Modal>
);
};

```

export default GroupInfoModal;

packages/frontend/src/components/Modals/Group/UpdateGroup.jsx:

```

import React, { useEffect, useState } from "react";
import axios from "../../utils/axios.js";
import { useAuth } from "../../contexts/AuthContext";
import { useForm, Controller } from "react-hook-form";
import {
  TextField,
  Button,
  Typography,
  Stack,
  Alert,
  Avatar,
  Box,
  CircularProgress,
  Menu,
  MenuItem,
} from "@mui/material";
import GroupsIcon from "@mui/icons-material/Groups";
import DeleteOutlineIcon from "@mui/icons-material/DeleteOutline";
import CloseIcon from "@mui/icons-material/Close";
import AddUserSearchInput from "../../Inputs/AddUserSearchInput";
import MoreVertIcon from "@mui/icons-material/MoreVert";

const UpdateGroup = ({
  handleClose,
  contacts,
  user,
  setUpdateChats,
  currentGroup,
  onLeave,
  onDelete,
  setCurrentChat,
}) => {
  const { control, handleSubmit } = useForm();
  const [errorMessage, setErrorMessage] = useState("");
  const [avatarUrl, setAvatarUrl] = useState(currentGroup.avatar);
  const [isLoading, setIsLoading] = useState(false);
  const [groupAdmins, setGroupAdmins] = useState(currentGroup.groupAdmins);
  const [selectedContacts, setSelectedContacts] = useState(currentGroup.users);
  const [anchorEl, setAnchorEl] = useState(null);
  const [anchorElTitle, setAnchorElTitle] = useState(null);

  const { token } = useAuth();

  const updateFields = (data) => {
    setAvatarUrl(data.avatar);
    setSelectedContacts(data.users);
    setGroupAdmins(data.groupAdmins);
  };

  useEffect(() => {

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		120

```

    updateFields(currentGroup);
  }, [currentGroup]);

const handleUpdateGroup = async (data) => {
  setIsLoading(true);
  try {
    const response = await axios.put(
      "/chat/group",
      {
        ...data,
        chatId: currentGroup._id,
        avatar: avatarUrl,
        users: selectedContacts,
        groupAdmins: groupAdmins,
      },
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      },
    );
    setUpdateChats(response.data);
    setCurrentChat(response.data);
    handleClose();
  } catch (error) {
    setErrorMessage("Помилка оновлення даних групи. Спробуйте пізніше");
    console.error("Error creating group:", error);
  }
  setIsLoading(false);
};

const handleAvatarUpload = async (event) => {
  setErrorMessage("");
  const file = event.target.files[0];
  if (file.type === "image/jpeg" || file.type === "image/png") {
    if (file.size > 10485760) {
      setErrorMessage("Файл є за великий. Максимальний розмір - 10MB");
    } else {
      const formData = new FormData();
      formData.append("file", file);

      try {
        const response = await axios.post("/upload", formData);
        setAvatarUrl(response.data.url);
      } catch (error) {
        setErrorMessage(
          "Помилка завантаження файлу. Спробуйте пізніше"
        );
      }
    }
  } else {
    setErrorMessage("Це має бути зображення");
  }
};

const addUser = (contact) => {
  setSelectedContacts((prevContacts) => [...prevContacts, contact]);
};

const deleteUser = (userId) => {
  setSelectedContacts((prevContacts) =>
    prevContacts.filter((contact) => contact._id !== userId)
  );
};

const handleMakeAdmin = (user) => {
  setGroupAdmins((prevAdmins) => [...prevAdmins, user]);
  setSelectedContacts((prevContacts) =>
    prevContacts.filter((contact) => contact._id !== user._id)
  );
};

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				121
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    );
    setAnchorEl(null);
    setAnchorElTitle(null);
  };

  const handleRemoveAdmin = (user) => {
    setGroupAdmins((prevAdmins) =>
      prevAdmins.filter((admin) => admin._id !== user._id)
    );
    setSelectedContacts((prevContacts) => [...prevContacts, user]);
    setAnchorEl(null);
    setAnchorElTitle(null);
  };

  const handleMenuOpen = (event, isAdmin) => {
    setAnchorEl(event.currentTarget);
    setAnchorElTitle(isAdmin ? "admin" : "user");
  };

  const handleMenuClose = () => {
    setAnchorEl(null);
    setAnchorElTitle(null);
  };

  return (
    <form onSubmit={handleSubmit(handleUpdateGroup)}>
      <Stack direction="column" alignItems="center" spacing={2}>
        <label htmlFor="avatar-upload" style={{ display: "none" }}>
          <input
            id="avatar-upload"
            name="avatar-upload"
            type="file"
            accept="image/png, image/jpeg"
            style={{ display: "none" }}
            onChange={handleAvatarUpload}
          />
        </label>
        <Avatar
          alt="Group Avatar"
          src={avatarUrl || ""}
          sx={{
            width: 60,
            height: 60,
            cursor: "pointer",
            background: "gray",
            color: "white",
          }}
          onClick={() => document.getElementById("avatar-upload").click()}
        >
          <GroupsIcon sx={{ width: 40, height: 40 }} />
        </Avatar>
        <input
          type="file"
          accept="image/png, image/jpeg"
          onChange={handleAvatarUpload}
          style={{ display: "none" }}
        />
      </Stack>
      <Controller
        name="title"
        control={control}
        defaultValue={currentGroup.chatName}
        rules={{ required: "Назва групи є обов'язкова" }}
        render={({ field, fieldState }) => (
          <TextField
            {...field}
            variant="outlined"
            label="Назва групи"
            fullWidth

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				122
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        margin="normal"
        error={!fieldState.error}
        helperText={fieldState.error ? fieldState.error.message : null}
      />
    )}
  />

<Controller
  name="description"
  control={control}
  defaultValue={currentGroup.chatDescription}
  render={({ field }) => (
    <TextField
      {...field}
      variant="outlined"
      label="Опис групи"
      fullWidth
      multiline
      rows={3}
      margin="normal"
    />
  )}
/>

<AddUserSearchInput
  onSubmit={addUser}
  selectedContacts={selectedContacts.concat(groupAdmins)}
/>

{groupAdmins.map((admin) => (
  <Box
    key={admin._id}
    sx={{ display: "flex", alignItems: "center", my: 1 }}
  >
    <Avatar
      alt={admin.email}
      src={admin.avatar}
      sx={{ mr: 2, background: "gray", color: "white" }}
    />
    <Box sx={{ flexGrow: 1 }}>
      <Typography>`${admin.firstName} ${admin.lastName}`</Typography>
      <Typography>{admin.email}</Typography>
    </Box>
    <Box sx={{ ml: 2 }}>
      <Typography variant="subtitle1" color="primary">
        Адмін
      </Typography>
    </Box>
    {user._id !== admin._id && (
      <>
        <Button onClick={(e) => handleMenuOpen(e, true)}>
          <MoreVertIcon />
        </Button>
        <Menu
          anchorEl={anchorEl}
          open={Boolean(anchorEl) && anchorElTitle === "admin"}
          onClose={handleMenuClose}
        >
          <MenuItem onClick={() => handleRemoveAdmin(admin)}>
            Збрати права Адміна
          </MenuItem>
        </Menu>
      </>
    )}
  </Box>
)}}

{selectedContacts.map((contact) => (
  <Box

```

		Іщенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				123
Змн.	Арк.	№ докум.	Підпис	Дата		

```

key={contact._id}
sx={{ display: "flex", alignItems: "center", my: 1 }}
>
<Avatar
  alt={contact.email}
  src={contact.avatar}
  sx={{ mr: 2, background: "gray", color: "white" }}
/>
<Box sx={{ flexGrow: 1 }}>
  <Typography>`${contact.firstName} ${contact.lastName}`</Typography>
  <Typography>{contact.email}</Typography>
</Box>
<Button onClick={() => deleteUser(contact._id)}>
  <CloseIcon />
</Button>
<Button onClick={(e) => handleMenuOpen(e, false)}>
  <MoreVertIcon />
</Button>
<Menu
  anchorEl={anchorEl}
  open={Boolean(anchorEl) && anchorElTitle === "user"}
  onClose={handleMenuClose}
>
  <MenuItem onClick={() => handleMakeAdmin(contact)}>
    Зробити Адміном
  </MenuItem>
</Menu>
</Box>
)))
{errorMessage && (
  <Alert severity="error" sx={{ mt: 2 }}>
    {errorMessage}
  </Alert>
)}
<Box sx={{ display: "flex", justifyContent: "flex-end", mt: 2 }}>
  <Button
    variant="contained"
    color="error"
    onClick={onDelete}
    startIcon={<DeleteOutlineIcon />}
  >
    Видалити групу
  </Button>
  <Button
    variant="outlined"
    color="error"
    sx={{ ml: 2 }}
    onClick={onLeave}
  >
    Покинути групу
  </Button>
  <Button
    onClick={handleClose}
    variant="outlined"
    color="primary"
    sx={{ ml: 2 }}
    disabled={isLoading}
  >
    Відмінити
  </Button>
  <Button
    type="submit"
    variant="contained"
    color="primary"
    disabled={isLoading}
    sx={{ ml: 2 }}
  >
    {isLoading ? <CircularProgress size={24} /> : "Зберегти"}

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				124
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        </Button>
      </Box>
    </form>
  );
};

export default UpdateGroup;

```

packages/frontend/src/components/Modals/Settings/ChangePasswordModal.jsx:

```

import React, { useState } from "react";
import { Box, Button, Modal, Typography, TextField } from "@mui/material";
import { useForm, Controller } from "react-hook-form";
import ModalBox from "../ModalBox";
import axios from "../../../utils/axios.js";
import { useAuth } from "../../../contexts/AuthContext";

const ChangePasswordModal = ({ open, onClose }) => {
  const { control, handleSubmit, reset } = useForm();
  const [successMessage, setSuccessMessage] = useState("");
  const [errorMessage, setErrorMessage] = useState("");
  const { token } = useAuth();

  const handleFormSubmit = async (data) => {
    if (data.oldPassword !== data.newPassword) {
      try {
        await axios.put("/user/password", data, {
          headers: {
            Authorization: `Bearer ${token}`,
          },
        });
      } catch (error) {
        setErrorMessage("Сталася помилка. Пароль не змінено. Спробуйте пізніше");
        console.error("Change password error: ", error);
      }
    } else {
      setErrorMessage("Попередній пароль є такий самий, як новий");
    }
  };

  const handleClose = () => {
    reset();
    setSuccessMessage("");
    setErrorMessage("");
    onClose();
  };

  return (
    <Modal open={open} onClose={handleClose}>
      <ModalBox>
        <Typography variant="h6" gutterBottom>
          Зміна паролю
        </Typography>
        <form onSubmit={handleSubmit(handleFormSubmit)}>
          <Box display="flex" flexDirection="column" gap={2}>
            <Controller
              name="oldPassword"
              control={control}
              defaultValue=""
              render={({ field, fieldState }) => (
                <TextField
                  {...field}
                  label="Старий пароль"
                  type="password"
                  variant="outlined"
                  fullWidth
                  error={!!fieldState.error}

```

```

        helperText={
          fieldState.error ? fieldState.error.message : null
        }
      />
    ))
    rules={{
      required: "Поле є обов'язковим",
    }}
  />
<Controller
  name="newPassword"
  control={control}
  defaultValue=""
  render={({ field, fieldState }) => (
    <TextField
      {...field}
      label="Новий пароль"
      type="password"
      variant="outlined"
      fullWidth
      error={!!fieldState.error}
      helperText={
        fieldState.error ? fieldState.error.message : null
      }
    />
  )
  rules={{
    required: "Поле є обов'язкове",
    minLength: {
      value: 8,
      message: "Пароль за короткий",
    },
  }}
/>
</Box>
{errorMessage && (
  <Typography color="error" mt={2}>
    {errorMessage}
  </Typography>
)}
{successMessage && (
  <Typography color="primary" mt={2}>
    {successMessage}
  </Typography>
)}
<Box mt={2} display="flex" justifyContent="flex-end">
  <Button onClick={handleClose} variant="outlined">
    Відміна
  </Button>
  <Button
    type="submit"
    color="primary"
    variant="contained"
    style={{ marginLeft: 10 }}
  >
    Змінити пароль
  </Button>
</Box>
</form>
</ModalBox>
</Modal>
);
};

```

export default ChangePasswordModal;

packages/frontend/src/components/Modals/Settings/SettingsModal.jsx:
import React, { useState } from "react";

		Іщенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				126
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import axios from "../../utils/axios.js";
import styled from "@emotion/styled";
import {
  Avatar,
  Button,
  Modal,
  Typography,
  TextField,
  Stack,
} from "@mui/material";

import ModalBox from "../ModalBox";
import { useAuth } from "../../contexts/AuthContext";
import { useForm, Controller } from "react-hook-form";
import ChangePasswordModal from "../ChangePasswordModal";

const Column = styled("div")(({ theme }) => ({
  padding: theme.spacing(2),
  display: "flex",
  justifyContent: "center",
  flexDirection: "column",
  alignItems: "center",
})));

const SettingsModal = ({ open, onClose, user, setUpdateChats }) => {
  const { token, login } = useAuth();
  const { control, handleSubmit } = useForm();
  const [errorMessage, setErrorMessage] = useState("");
  const [successMessage, setSuccessMessage] = useState("");
  const [avatarUrl, setAvatarUrl] = useState(user.avatar);
  const [changePasswordModalIsOpen, setChangePasswordModalIsOpen] =
    useState(false);

  const handleClose = () => {
    setSuccessMessage("");
    setErrorMessage("");
    onClose();
  };

  const onSubmit = async (data) => {
    try {
      if (
        data.firstName !== user.firstName ||
        data.lastName !== user.lastName ||
        data.email !== user.email ||
        avatarUrl !== user.avatar
      ) {
        data.avatar = avatarUrl;
        const response = await axios.put(`/user/me`, data, {
          headers: {
            Authorization: `Bearer ${token}`,
          },
        });
        login(response.data.token);
        setSuccessMessage("Зміни збережено");
        setUpdateChats(response.data);
      }
      handleClose();
    } catch (error) {
      console.error(error);
      setErrorMessage("Помилка оновлення даних. Спробуйте пізніше");
    }
  };

  const handleAvatarUpload = async (event) => {
    setErrorMessage("");
    const file = event.target.files[0];
    if (file.type === "image/jpeg" || file.type === "image/png") {
      if (file.size > 10485760) {

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				127
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    setErrorMessage("Файл є за великий. Максимальний розмір - 10MB");
  } else {
    const formData = new FormData();
    formData.append("file", file);

    try {
      const response = await axios.post("/upload", formData);
      setAvatarUrl(response.data.url);
    } catch (error) {
      setErrorMessage(
        "Помилка завантаження файлу. Спробуйте пізніше"
      );
    }
  }
} else {
  setErrorMessage("Це має бути зображення");
}
};

return (
  user && (
    <>
    <ChangePasswordModal
      open={changePasswordModalIsOpen}
      onClose={() => setChangePasswordModalIsOpen(false)}
    />
    <Modal open={open} onClose={handleClose}>
      <ModalBox>
        <Column>
          <Typography component="h2" variant="h5">
            Налаштування
          </Typography>
          <form onSubmit={handleSubmit(onSubmit)} style={{ width: "100%" }}>
            <Stack spacing={2} mt={2} width="100%">
              <Stack
                direction="column"
                alignItems="center"
                justifyContent="center"
              >
                <Controller
                  name="avatar"
                  control={control}
                  defaultValue=""
                  render={({ field }) => (
                    <input
                      type="file"
                      accept="image/png, image/jpeg"
                      onChange={handleAvatarUpload}
                      {...field}
                      style={{ display: "none" }}
                    />
                  )}
                />
                {avatarUrl ? (
                  <Avatar
                    alt="Avatar"
                    src={avatarUrl}
                    sx={{
                      width: 100,
                      height: 100,
                      my: 1,
                      cursor: "pointer",
                    }}
                    onClick={() =>
                      document.getElementById("avatar-upload").click()
                    }
                  />
                ) : (
                  <Avatar

```

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				128
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        alt="Avatar"
        sx={{
          width: 100,
          height: 100,
          my: 1,
          cursor: "pointer",
          background: "gray",
          color: "white",
        }}
        onClick={() =>
          document.getElementById("avatar-upload").click()
        }
      />
    )}
    <label htmlFor="avatar-upload" style={{ display: "none" }}>
      <input
        id="avatar-upload"
        name="avatar-upload"
        type="file"
        accept="image/png, image/jpeg"
        style={{ display: "none" }}
        onChange={handleAvatarUpload}
      />
    </label>
    {avatarUrl && (
      <Button
        variant="outlined"
        color="error"
        fullWidth
        onClick={() => setAvatarUrl(null)}
      >
        Видалити зображення профілю
      </Button>
    )}
  </Stack>
  <Stack direction="row" spacing={2}>
    <Controller
      name="firstName"
      control={control}
      defaultValue={user.firstName}
      render={({ field, fieldState }) => (
        <TextField
          {...field}
          label="Ім'я"
          variant="outlined"
          fullWidth
          error={!!fieldState.error}
          helperText={
            fieldState.error ? fieldState.error.message : null
          }
        />
      )}
      rules={{ required: "Ім'я є обов'язковим" }}
    />
    <Controller
      name="lastName"
      control={control}
      defaultValue={user.lastName}
      render={({ field, fieldState }) => (
        <TextField
          {...field}
          label="Прізвище"
          variant="outlined"
          fullWidth
          error={!!fieldState.error}
          helperText={
            fieldState.error ? fieldState.error.message : null
          }
        />
      )}
    />
  </Stack>

```

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				129
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    ))
    rules={{ required: "Прізвище є обов'язковим" }}
  />
</Stack>
<Controller
  name="email"
  control={control}
  defaultValue={user.email}
  render={({ field, fieldState }) => (
    <TextField
      {...field}
      label="Email"
      variant="outlined"
      fullWidth
      error={!!fieldState.error}
      helperText={
        fieldState.error ? fieldState.error.message : null
      }
    />
  )}
  rules={{
    required: "Email є обов'язковим",
    pattern: {
      value:
        /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/ ,
      message: "Email не відповідає формату",
    },
  }}
/>
{errorMessage && (
  <Typography color="error">{errorMessage}</Typography>
)}
{successMessage && (
  <Typography color="primary">{successMessage}</Typography>
)}
<Button
  variant="outlined"
  fullWidth
  onClick={() => setChangePasswordModalIsOpen(true)}
>
  Змінити пароль
</Button>
<Stack direction="row" spacing={2}>
  <Button variant="outlined" onClick={handleClose} fullWidth>
    Закрити
  </Button>
  <Button
    type="submit"
    variant="contained"
    color="primary"
    fullWidth
  >
    Зберегти
  </Button>
</Stack>
</Stack>
</form>
</Column>
</ModalBox>
</Modal>
</>
)
);
};

```

export default SettingsModal;

packages/frontend/src/components/Route/PrivateRoute.jsx:

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				130
Змн.	Арк.	№ докум.	Підпис	Дата		

```
import React from "react";
import { Navigate } from "react-router-dom";
import { useAuth } from "../../contexts/AuthContext";
import { SocketProvider } from "../../contexts/SocketContext";
```

```
const PrivateRoute = ({ element: Element, ...rest }) => {
  const { isAuthenticated } = useAuth();

  return isAuthenticated ? (
    <SocketProvider>
      <Element {...rest} />{" "}
    </SocketProvider>
  ) : (
    <Navigate to="/sign-in" />
  );
};
```

```
export default PrivateRoute;
```

packages/frontend/src/components/Route/PublicRoute.jsx:

```
import React from 'react';
import { Navigate } from 'react-router-dom';
import { useAuth } from '../../contexts/AuthContext';
```

```
const PublicRoute = ({ element: Element, ...rest }) => {
  const { isAuthenticated } = useAuth();

  return isAuthenticated ? <Navigate to="/" replace /> : <Element {...rest} /> ;
};
```

```
export default PublicRoute;
```

packages/frontend/src/components/Sidebar/Notification.jsx:

```
import React, { useEffect, useState } from "react";
import { useSocket } from "../../contexts/SocketContext";
import { Box, Typography } from "@mui/material";
```

```
const NotificationBox = ({ chatId }) => {
  const { notification } = useSocket();
  const [notifCount, setNotifCount] = useState(0);

  useEffect(() => {
    setNotifCount(
      notification?.filter((notif) => notif.chat._id === chatId)?.length || 0
    );
  }, [notification, chatId]);
```

```
  return (
    notifCount > 0 && (
      <Box
        sx={{
          width: "30px",
          height: "30px",
          display: "flex",
          justifyContent: "center",
          alignItems: "center",
          backgroundColor: "primary.main",
          color: "secondary.main",
          borderRadius: "50%",
        }}
      >
        <Typography
          component="h6"
          variant="subtitle1"
          sx={{
            width: "100%",
            height: "100%",
            display: "flex",
```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		131

```

        justifyContent: "center",
        alignItems: "center",
      }}
    >
      {notifCount}
    </Typography>
  </Box>
)
);
};

```

export default NotificationBox;

packages/frontend/src/components/Sidebar/Sidebar.jsx:

```

import React, { useEffect, useState } from "react";
import Box from "@mui/material/Box";
import List from "@mui/material/List";
import ListItem from "@mui/material/ListItem";
import Typography from "@mui/material/Typography";
import LinearProgress from "@mui/material/LinearProgress";
import Avatar from "@mui/material/Avatar";
import { styled } from "@mui/system";
import { useAuth } from "../../../contexts/AuthContext";
import axios from "../../../utils/axios.js";
import { getChatInfo } from "../../../utils/chat";
import GroupsIcon from "@mui/icons-material/Groups";
import NotificationBox from "../NotificationBox";
import { decryptMessage } from "../../../utils/encryption";

```

```

const StyledSidebar = styled(Box)({
  width: "100%",
  height: "calc(100vh - 64px)",
  display: "flex",
  flexDirection: "column",
  borderRight: "1px solid #ccc",
  overflowY: "scroll",
  "&::-webkit-scrollbar": {
    width: "8px",
  },
  "&::-webkit-scrollbar-thumb": {
    backgroundColor: "#bbb",
    borderRadius: "4px",
  },
  "&::-webkit-scrollbar-thumb:hover": {
    backgroundColor: "#888",
  },
});

```

```

const Sidebar = ({
  user,
  currentChat,
  setCurrentChat,
  updateChats,
  chats,
  setChats,
}) => {
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState("");
  const { token } = useAuth();

  useEffect(() => {
    const fetchChats = async () => {
      try {
        if (!chats) setLoading(true);
        const response = await axios.get("/chat", {
          headers: {
            Authorization: `Bearer ${token}`,
          },
        });

```

```

    });
    setChats(response.data);
  } catch (err) {
    setError("Помилка завантаження чатів");
  } finally {
    setLoading(false);
  }
};

fetchChats();
}, [token, updateChats, setChats]));

return (
  <StyledSidebar>
    {loading ? (
      <LinearProgress />
    ) : error ? (
      <Typography variant="body1" color="error">
        {error}
      </Typography>
    ) : (
      <List sx={{ flexGrow: 1, px: 1 }}>
        {chats && chats.length > 0 ? (
          chats.map((chat, index) => {
            const chatInfo = getChatInfo(chat, user);
            return (
              <React.Fragment key={index}>
                <ListItem>
                  button
                  onClick={() => setCurrentChat(chat)}
                  sx={{
                    borderRadius: 4,
                    backgroundColor:
                      chat._id === currentChat._id ? "#31473A30" : "inherit",
                  }}
                >
                  <Box>
                    sx={{
                      width: "100%",
                      display: "flex",
                      alignItems: "center",
                    }}
                  >
                    <Box>
                      sx={{
                        width: "90%",
                        display: "flex",
                        alignItems: "center",
                        overflowX: "hidden",
                      }}
                    >
                      <Avatar>
                        alt={chatInfo.title}
                        src={chatInfo.avatar}
                        sx={{
                          marginRight: "8px",
                          background: "gray",
                          color: "white",
                        }}
                      >
                        {chat.isGroupChat && !chat.avatar && <GroupsIcon />}
                      </Avatar>
                      <Box>
                        <Typography variant="body1">
                          {chatInfo.title}
                        </Typography>
                        {chat.latestMessage &&
                          (chat.latestMessage.text ? (
                            <Typography variant="body2" noWrap={true}>

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				133
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {decryptMessage(
          chat.latestMessage.text,
          chat._id
        )}
      </Typography>
    ) : (
      chat.latestMessage.attachments &&
      chat.latestMessage.attachments.length > 0 && (
        <Typography variant="body2">
          {
            chat.latestMessage.attachments[0]
              .originalname
          }
        </Typography>
      )
    )
  </Box>
</Box>
<Box sx={{ width: "10%" }}>
  <NotificationBox chatId={chat._id} />
</Box>
</Box>
</ListItem>
</React.Fragment>
);
})
) : (
  <Typography variant="body1">
    Ви ще не маєте чатів. Скористайтеся пошуком контактів, щоб ропочати
    спілкування
  </Typography>
)
</List>
))
</StyledSidebar>
);
};

export default Sidebar;

```

packages/frontend/src/contexts/AuthContext.jsx:

```

import React, {
  createContext,
  useContext,
  useState,
  useEffect,
  useCallback,
} from "react";
import axios from "../utils/axios.js";

const AuthContext = createContext();

const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);
  const [token, setToken] = useState(localStorage.getItem("token") || null);
  const [isAuthenticated, setIsAuthenticated] = useState(!!token);

  const login = (newToken) => {
    setToken(newToken);
    localStorage.setItem("token", newToken);
  };

  const logout = () => {
    setToken(null);
    setIsAuthenticated(false);
    localStorage.removeItem("token");
  };
};

```

```

const fetchUserData = useCallback(async () => {
  if (token && token !== "undefined") {
    try {
      const response = await axios.get("/user/me", {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      });
      setUser(response.data);
      setIsAuthenticated(true);
    } catch (error) {
      console.error("Error fetching user data", error);
      setIsAuthenticated(false);
      logout();
    }
  }
}, [token]);

useEffect(() => {
  if (token) {
    fetchUserData();
  }
}, [token, fetchUserData]);

return (
  <AuthContext.Provider
    value={{ user, token, login, logout, isAuthenticated }}
  >
    {children}
  </AuthContext.Provider>
);
};

const useAuth = () => {
  const context = useContext(AuthContext);
  if (!context) {
    throw new Error("useAuth must be used within an AuthProvider");
  }
  return context;
};

export { AuthProvider, useAuth };

```

packages/frontend/src/context/SocketContext.jsx:

```

import React, { createContext, useContext, useEffect, useState } from "react";
import io from "socket.io-client";
import axios from "../utils/axios.js";
import { useAuth } from "../AuthContext";
import { latestActivityFormatDateTime } from "../utils/datetime";

const SOCKET_SERVER_URL = process.env.REACT_APP_API_BASE_URL;

const SocketContext = createContext();

const SocketProvider = ({ children }) => {
  const [socket, setSocket] = useState(null);
  const [socketConnected, setSocketConnected] = useState(false);
  const [usersStatuses, setUsersStatuses] = useState({});
  const [selectedChatCompare, setSelectedChatCompare] = useState(null);
  const [notification, setNotification] = useState([]);

  const { user, token } = useAuth();

  useEffect(() => {
    const getNotification = async () => {
      const response = await axios.get("/message/notif", {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      });
    };
  }, [token]);

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				135
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    },
  });
  setNotification(response.data);
};
getNotification();
}, []);

useEffect(() => {
  if (user) {
    const newSocket = io(SOCKET_SERVER_URL);

    newSocket.emit("setup", user);
    newSocket.on("connected", () => setSocketConnected(true));
    newSocket.on("userStatus", (status) => {
      setUsersStatuses(status);
    });

    setSocket(newSocket);

    return () => {
      newSocket.off("userStatus");
    };
  }
}, [user]);

const isUserOnline = (userId) => {
  return usersStatuses[userId].status === "online";
};

const getUserStatus = (userId) => {
  if (usersStatuses[userId]) {
    if (isUserOnline(userId)) {
      return "Online";
    } else {
      return latestActivityFormatDateTime(
        usersStatuses[userId].latestActivity
      );
    }
  }
};

return (
  <SocketContext.Provider
    value={{
      socket,
      socketConnected,
      usersStatuses,
      isUserOnline,
      getUserStatus,
      selectedChatCompare,
      setSelectedChatCompare,
      notification,
      setNotification,
    }}
  >
    {children}
  </SocketContext.Provider>
);
};

const useSocket = () => {
  const context = useContext(SocketContext);
  if (!context) {
    throw new Error("useSocket must be used within an SocketProvider");
  }
  return context;
};

export { SocketProvider, useSocket };

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				136
Змн.	Арк.	№ докум.	Підпис	Дата		

packages/frontend/src/pages/Chat.jsx:

```
import React, { useState, useEffect } from "react";
import { Box, CircularProgress, Grid, Typography } from "@mui/material";

import { getContacts } from "../../utils/contacts";
import { useAuth } from "../../contexts/AuthContext";
import ChatBox from "../../components/ChatBox/ChatBox";
import Header from "../../components/Header/Header";
import Sidebar from "../../components/Sidebar/Sidebar";
import { useSocket } from "../../contexts/SocketContext";

function Chat() {
  const { user } = useAuth();
  const [messages, setMessages] = useState([]);
  const [currentChat, setCurrentChat] = useState({});
  const [isSmallScreen, setIsSmallScreen] = useState(window.innerWidth < 600);
  const [updateChats, setUpdateChats] = useState();
  const [chats, setChats] = useState([]);
  const [contacts, setContacts] = useState(getContacts(chats, user));
  const {
    socket,
    selectedChatCompare,
    setSelectedChatCompare,
    notification,
    setNotification,
  } = useSocket();

  useEffect(() => {
    setSelectedChatCompare(currentChat._id ? currentChat : null);
  }, [currentChat]);

  useEffect(() => {
    setContacts(getContacts(chats, user));
  }, [chats, user]);

  useEffect(() => {
    const handleResize = () => {
      setIsSmallScreen(window.innerWidth < 600);
    };

    window.addEventListener("resize", handleResize);

    return () => {
      window.removeEventListener("resize", handleResize);
    };
  }, []);

  useEffect(() => {
    if (socket) {
      socket.on("received", (message) => {
        if (
          !selectedChatCompare ||
          selectedChatCompare._id !== message.chat._id
        ) {
          if (!notification.includes(message)) {
            setNotification((prev) => [...prev, message]);
            setUpdateChats(message);
          }
        } else {
          if (selectedChatCompare._id === message.chat._id) {
            setMessages((prevMessages) => [...prevMessages, message]);
          }
        }
      });
    }

    return () => {
      socket.off("received");
    };
  });
}
```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		137

```

    }
  }, [socket, selectedChatCompare]));

return user ? (
  <>
    <Header
      user={user}
      setCurrentChat={setCurrentChat}
      setUpdateChats={setUpdateChats}
      contacts={contacts}
    />
    <Box
      sx={{ flexGrow: 1, display: "flex", overflow: "hidden", mt: "64px" }}
    >
      <Grid container>
        {((isSmallScreen && Object.keys(currentChat).length === 0) ||
        !isSmallScreen) && (
          <Grid item xs={12} sm={4} md={3}>
            <Sidebar
              user={user}
              currentChat={currentChat}
              setCurrentChat={setCurrentChat}
              updateChats={updateChats}
              chats={chats}
              setChats={setChats}
            />
          </Grid>
        )}
        {((isSmallScreen && Object.keys(currentChat).length > 0) ||
        !isSmallScreen) && (
          <Grid item xs={12} sm={8} md={9}>
            {Object.keys(currentChat).length > 0 ? (
              <ChatBox
                isSmallScreen={isSmallScreen}
                user={user}
                currentChat={currentChat}
                setCurrentChat={setCurrentChat}
                setUpdateChats={setUpdateChats}
                messages={messages}
                setMessages={setMessages}
              />
            ) : (
              <Box
                sx={{
                  display: "flex",
                  justifyContent: "center",
                  alignItems: "center",
                  height: "100%",
                }}
              >
                <Typography variant="h6">
                  Обери чат щоб розпочати спілкування
                </Typography>
              </Box>
            )}
          </Grid>
        )}
      </Grid>
    </Box>
  </>
) : (
  <CircularProgress />
);
}

export default Chat;

```

packages/frontend/src/pages/SignIn.jsx:

		Иценко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		138

```

import React, { useState } from "react";
import axios from "../utils/axios.js";
import { useAuth } from "../contexts/AuthContext";
import { useForm, Controller } from "react-hook-form";
import {
  TextField,
  Button,
  Container,
  Typography,
  Link,
  Stack,
  Alert,
} from "@mui/material";

function SignIn() {
  const { control, handleSubmit } = useForm();
  const { login } = useAuth();
  const [errorMessage, setErrorMessage] = useState("");

  const onSubmit = (formData) => {
    axios
      .post("/user/sign-in", formData)
      .then(({ data }) => {
        setErrorMessage("");
        login(data.token);
      })
      .catch((err) => {
        if (err.response && err.response.status === 401) {
          setErrorMessage("Email або пароль невірні. Спробуйте ще раз");
        } else {
          setErrorMessage("Сталася помилка. Спробуйте пізніше");
        }
      });
  };

  return (
    <Stack
      direction="row"
      justifyContent="center"
      alignItems="center"
      sx={{ width: 1, height: "100vh" }}
      backgroundColor="#EDF4F2"
    >
      <Container
        component="main"
        maxWidth="xs"
        style={{
          backgroundColor: "#fafafa",
          padding: "60px 40px",
          borderRadius: 10,
        }}
      >
        <div
          style={{
            display: "flex",
            flexDirection: "column",
            alignItems: "center",
          }}
        >
          <Typography component="h2" variant="h5">
            Вхід
          </Typography>
          <form
            style={{ width: "100%", marginTop: "1rem" }}
            onSubmit={handleSubmit(onSubmit)}
          >
            <Controller
              name="email"
              control={control}

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				139
Змн.	Арк.	№ докум.	Підпис	Дата		

```

defaultValue=""
render=(({ field, fieldState }) => (
  <TextField
    {...field}
    variant="outlined"
    margin="normal"
    fullWidth
    label="Email"
    error={!fieldState.error}
    helperText={
      fieldState.error ? fieldState.error.message : null
    }
  />
))
rules={{
  required: "Поле Email обов'язкове",
  pattern: {
    value: /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/ ,
    message: "Це не є Email. Введіть в правильній формі",
  },
}}
/>
<Controller
  name="password"
  control={control}
  defaultValue=""
  render=(({ field, fieldState }) => (
    <TextField
      {...field}
      variant="outlined"
      margin="normal"
      fullWidth
      label="Пароль"
      type="password"
      autoComplete="password"
      error={!fieldState.error}
      helperText={
        fieldState.error ? fieldState.error.message : null
      }
    />
  ))
  rules={{
    required: "Поле 'Пароль' є обов'язковим",
  }}
/>
{errorMessage && (
  <Alert severity="error" fullWidth="true">
    {errorMessage}
  </Alert>
)}
<Button
  type="submit"
  fullWidth
  variant="contained"
  color="primary"
  style={{ marginTop: "1rem", background: "#31473A" }}
>
  Увійти
</Button>
<Typography
  component="p"
  variant="body2"
  pt={2}
  style={{ textAlign: "center" }}
>
  Не маєте облікового запису? <Link href="/sign-up">Зареєструйтеся</Link>
</Typography>
</form>
</div>

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				140
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    </Container>
  </Stack>
);
}

export default SignIn;

```

packages/frontend/src/pages/SignUp.jsx:

```

import React, { useState } from "react";
import axios from "../utils/axios.js";
import { useAuth } from "../contexts/AuthContext";
import { useForm, Controller } from "react-hook-form";
import {
  TextField,
  Button,
  Container,
  Typography,
  Link,
  Stack,
  Alert,
  Avatar,
} from "@mui/material";

const SignUp = () => {
  const { control, handleSubmit, watch } = useForm();
  const { login } = useAuth();
  const [errorMessage, setErrorMessage] = useState("");
  const [avatarUrl, setAvatarUrl] = useState(null);

  const onSubmit = (formData) => {
    formData.avatar = avatarUrl; // Add avatarUrl to form data
    axios
      .post("/user/sign-up", formData)
      .then(({ data }) => {
        setErrorMessage("");
        login(data.token);
      })
      .catch((err) => {
        if (err.response) {
          setErrorMessage(err.response.data.error);
        } else {
          setErrorMessage("Сталася помилка. Спробуйте пізніше");
        }
      });
  };

  const watchPassword = watch("password", "");

  const handleAvatarUpload = async (event) => {
    setErrorMessage("");
    const file = event.target.files[0];
    if (file.type === "image/jpeg" || file.type === "image/png") {
      if (file.size > 10485760) {
        setErrorMessage("Ваша фотографія профілю за велика. Максимальний розмір - 10MB");
      } else {
        const formData = new FormData();
        formData.append("file", file);

        try {
          const response = await axios.post("/upload", formData);
          setAvatarUrl(response.data.url);
        } catch (error) {
          setErrorMessage(
            "Помилка завантаження файлу. Спробуйте пізніше"
          );
        }
      }
    }
  };
}

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		141

```

    } else {
      setErrorMessage("Зображення профілю не може бути цього типу");
    }
  };

return (
  <Stack
    direction="row"
    justifyContent="center"
    alignItems="center"
    sx={{ width: 1, height: "100vh" }}
    backgroundColor="#EDF4F2"
  >
    <Container
      component="main"
      maxWidth="xs"
      style={{
        backgroundColor: "#fafafa",
        padding: "60px 40px",
        borderRadius: 10,
      }}
    >
      <div
        style={{
          display: "flex",
          flexDirection: "column",
          alignItems: "center",
        }}
      >
        <Typography component="h2" variant="h5">
          Реєстрація
        </Typography>
        <form
          style={{ width: "100%", marginTop: "1rem" }}
          onSubmit={handleSubmit(onSubmit)}
        >
          <Stack direction="row" spacing={2}>
            <Controller
              name="firstName"
              control={control}
              defaultValue=""
              render={({ field, fieldState }) => (
                <TextField
                  {...field}
                  variant="outlined"
                  label="Ім'я"
                  error={!!fieldState.error}
                  helperText={
                    fieldState.error ? fieldState.error.message : null
                  }
                />
              )}
              rules={{ required: "Поле 'Ім'я' є обов'язковим" }}
            />
            <Controller
              name="lastName"
              control={control}
              defaultValue=""
              render={({ field, fieldState }) => (
                <TextField
                  {...field}
                  variant="outlined"
                  label="Прізвище"
                  error={!!fieldState.error}
                  helperText={
                    fieldState.error ? fieldState.error.message : null
                  }
                />
              )}
            />
          </Stack>
        </form>
      </div>
    </Container>
  </Stack>
);

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				142
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        rules={{ required: "Поле 'Прізвище' є обов'язковим" }}
      />
    </Stack>
    <Controller
      name="email"
      control={control}
      defaultValue=""
      render={({ field, fieldState }) => (
        <TextField
          {...field}
          variant="outlined"
          margin="normal"
          fullWidth
          label="Email"
          autoComplete="email"
          error={!fieldState.error}
          helperText={
            fieldState.error ? fieldState.error.message : null
          }
        />
      )}
      rules={{
        required: "Поле Email обов'язкове",
        pattern: {
          value: /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/ ,
          message: "Це не є Email. Введіть в правильній формі",
        },
      }}
    />
    <Controller
      name="password"
      control={control}
      defaultValue=""
      render={({ field, fieldState }) => (
        <TextField
          {...field}
          variant="outlined"
          margin="normal"
          fullWidth
          label="Пароль"
          type="password"
          autoComplete="new-password"
          error={!fieldState.error}
          helperText={
            fieldState.error ? fieldState.error.message : null
          }
        />
      )}
      rules={{
        required: "Поле 'Пароль' є обов'язковим",
        minLength: {
          value: 8,
          message: "Пароль є за короткий.",
        },
      }}
    />
    <Controller
      name="confirmPassword"
      control={control}
      defaultValue=""
      render={({ field, fieldState }) => (
        <TextField
          {...field}
          variant="outlined"
          margin="normal"
          fullWidth
          label="Підтвердіть пароль"
          type="password"
          autoComplete="new-password"

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				143
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        error={!fieldState.error}
        helperText={
          fieldState.error ? fieldState.error.message : null
        }
      />
    )}
    rules={{
      required: "Поле 'Підтвердіть пароль' є обов'язковим",
      validate: (value) =>
        value === watchPassword || "Паролі не однакові",
    }}
  />
<Stack direction="row" alignItems="center" justifyContent="center">
  <Controller
    name="avatar"
    control={control}
    defaultValue=""
    render={({ field }) => (
      <input
        type="file"
        accept="image/png, image/jpeg"
        onChange={handleAvatarUpload}
        {...field}
        style={{ display: "none" }}
      />
    )}
  />
  {avatarUrl ? (
    <Avatar
      alt="Avatar"
      src={avatarUrl}
      sx={{ width: 100, height: 100, my: 1, cursor: "pointer" }}
      onClick={() =>
        document.getElementById("avatar-upload").click()
      }
    />
  ) : (
    <Avatar
      alt="Avatar"
      sx={{
        width: 100,
        height: 100,
        my: 1,
        cursor: "pointer",
        background: "gray",
        color: "white",
      }}
      onClick={() =>
        document.getElementById("avatar-upload").click()
      }
    />
  )}
  <label htmlFor="avatar-upload" style={{ display: "none" }}>
    <input
      id="avatar-upload"
      name="avatar-upload"
      type="file"
      accept="image/png, image/jpeg"
      style={{ display: "none" }}
      onChange={handleAvatarUpload}
    />
  </label>
</Stack>
{errorMessage && (
  <Alert severity="error" fullWidth="true">
    {errorMessage}
  </Alert>
)}
</Button

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				144
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        type="submit"
        fullWidth
        variant="contained"
        color="primary"
        style={{ marginTop: "1rem" }}
      >
        Зареєструвати
      </Button>
    <Typography
      component="p"
      variant="body2"
      pt={2}
      style={{ textAlign: "center" }}
    >
      Уже маєте обліковий запис? <Link href="/sign-in">Увійдіть</Link>
    </Typography>
  </form>
</div>
</Container>
</Stack>
);
};

```

```
export default SignUp;
```

packages/frontend/src/utils/axios.js:

```

import axios from "axios";

const instance = axios.create({
  baseURL: process.env.REACT_APP_API_BASE_URL,
});

export default instance;

```

packages/frontend/src/utils/chat.js:

```

import { getContact } from "../contacts";

export const getChatInfo = (chat, user) => {
  if (chat.isGroupChat) {
    return {
      title: chat.chatName,
      description: chat.chatDescription,
      avatar: chat.avatar,
    };
  } else {
    const contact = getContact(chat, user);
    return {
      title: `${contact.firstName} ${contact.lastName}`,
      avatar: contact.avatar,
      email: contact.email,
    };
  }
};

```

packages/frontend/src/utils/contacts.js:

```

export const getContacts = (chats, user) => {
  const filteredChats = chats.filter((chat) => !chat.isGroupChat);

  const contacts = [];
  filteredChats.forEach((chat) => {
    chat.users.forEach((chatUser) => {
      if (chatUser._id !== user._id) {
        if (!contacts.find((contact) => contact._id === chatUser._id)) {
          contacts.push(chatUser);
        }
      }
    });
  });
};

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		145

```

    return contacts;
};

export const getContact = (chat, user) => {
    return chat.users.filter((chatUser) => chatUser._id !== user._id)[0];
};

packages/frontend/src/utils/datetime.js:
export const formatTime = (timestamp) => {
    const date = new Date(timestamp);
    const hours = date.getHours().toString().padStart(2, "0");
    const minutes = date.getMinutes().toString().padStart(2, "0");
    return `${hours}:${minutes}`;
};

export const formatDate = (date) => {
    const currentDate = new Date();
    const messageDate = new Date(date);

    const today = new Date(currentDate);
    today.setHours(0, 0, 0, 0);

    const yesterday = new Date(currentDate);
    yesterday.setDate(currentDate.getDate() - 1);
    yesterday.setHours(0, 0, 0, 0);

    if (messageDate >= today) {
        return "Сьогодні";
    } else if (messageDate >= yesterday) {
        return "Вчора";
    } else {
        const options = { year: "numeric", month: "long", day: "numeric" };
        return messageDate.toLocaleDateString("uk-UA", options);
    }
};

export const latestActivityFormatDateTime = (timestamp) => {
    const currentDate = new Date();
    const activityDate = new Date(timestamp);
    const timeDiff = currentDate - activityDate;

    const timeDiffInSeconds = Math.floor(timeDiff / 1000);

    if (timeDiffInSeconds < 60) {
        return "Few seconds ago";
    } else if (timeDiffInSeconds < 3600) {
        const minutes = Math.floor(timeDiffInSeconds / 60);
        return `${minutes} minute${minutes > 1 ? "s" : ""} ago`;
    } else if (timeDiffInSeconds < 21600) {
        const hours = Math.floor(timeDiffInSeconds / 3600);
        return `${hours} hour${hours > 1 ? "s" : ""} ago`;
    } else if (timeDiffInSeconds < 86400) {
        const hours = activityDate.getHours().toString().padStart(2, "0");
        const minutes = activityDate.getMinutes().toString().padStart(2, "0");
        return `${hours}:${minutes}`;
    } else if (timeDiffInSeconds < 172800) {
        return "Yesterday";
    } else {
        const options = { day: "numeric", month: "long", year: "numeric" };
        return activityDate.toLocaleDateString(undefined, options);
    }
};

```

packages/frontend/src/utils/encryption.js:

```

import CryptoJS from "crypto-js";

export const encryptMessage = (message, key) => {

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		146

```

    return CryptoJS.AES.encrypt(message, key).toString();
};

export const decryptMessage = (encryptedMessage, key) => {
  try {
    const bytes = CryptoJS.AES.decrypt(encryptedMessage, key);
    return bytes.toString(CryptoJS.enc.Utf8);
  } catch (error) {
    return encryptedMessage;
  }
};

```

packages/frontend/src/utlis/messages.js:

```

export const isSameSender = (messages, i) => {
  return (
    i > 0 &&
    messages[i] &&
    messages[i].sender &&
    messages[i - 1].sender &&
    messages[i - 1].sender._id === messages[i].sender._id
  );
};

export const isLastMessage = (messages, i) => {
  return (
    i === messages.length - 1 ||
    (i < messages.length - 1 &&
      messages[i + 1].sender._id !== messages[i].sender._id)
  );
};

export const isNewDayMessage = (messages, i) => {
  if (i === 0) {
    return true;
  }

  const prevMessageDate = new Date(messages[i - 1].createdAt).getDate();
  const currentMessageDate = new Date(messages[i].createdAt).getDate();

  return prevMessageDate !== currentMessageDate;
};

```

		Ищенко А.Ю.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		147