МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри, канд. техн. наук,
доцент_____Є. О. Давиденко
«___»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

«REAL-TIME ЧАТ-МЕСЕНДЖЕР»

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 408.22011024

Здобувач _____О. О. Шевченко
«___»_____2024 р.

Керівник завідувач кафедри ІПЗ, канд. техн. наук, доцент _____ Є. О. Давиденко
«___»_____2024 р.

Консультант канд. техн. наук, доцент _____А. О. Алексєєва
«___»_____2024 р.

м. Миколаїв – 2024 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ
Зав. кафедри
_____ Є. О. Давиденко
« » ___ 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано здобувачу групи 408 факультету комп'ютерних наук
Шевченко Олександру Олександровичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи
Real-time чат-месенджер

Затверджена наказом по ЧНУ від «22» грудня 2023 р. № 269
2. Строк представлення кваліфікаційної роботи «___» червня 2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є розробка та впровадження Real-time чат-месенджера з такими функціями, як:

- реєстрація та авторизація користувачів;
- створення особистих та групових чатів;
- надсилання текстових повідомлень в реальному часі;
- отримання та відображення повідомлень у реальному часі;
- управління профілем користувача (редагування інформації);
- безпека та конфіденційність даних користувачів

4. Перелік питань, що підлягають розробці:

- дослідження предметної галузі та аналіз існуючих аналогів на ринку чат-месенджерів;
- формування специфікації вимог до Real-time чат-месенджера з урахуванням функціональності обміну повідомленнями в реальному часі;
- визначення архітектури та інтеграція необхідних технологій для забезпечення обробки усіх дій користувача у реальному часі;
- моделювання та проєктування інтерфейсу користувача для зручного та інтуїтивного використання чат-месенджера;

– розробка програмного забезпечення для реалізації всіх функцій, визначених у специфікації вимог;

– тестування Real-time чат-месенджера з акцентом на перевірку коректності та швидкодії обміну повідомленнями у реальному часі;

5. Перелік графічних матеріалів

презентація

6. Завдання до спеціальної частини

Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алексєєва А. О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи: завідувач кафедри ІПЗ, канд. техн. наук,доцент Давиденко Євген Олександрович

(посада, прізвище, ім'я, по батькові)

(підпис)

Завдання прийнято до виконання

Шевченко Олександр Олександрович

(прізвище, ім'я, по батькові здобувача)

(підпис)

Дата видачі завдання «____» _____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема кваліфікаційної роботи: Real-time чат-месенджер

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	15.12.2023	22.12.2023	Виконано
2.	Огляд літератури за темою роботи	20.01.2024	30.01.2024	Виконано
3.	Складання календарного плану КРБ	01.02.2024	02.02.2024	Виконано
4.	Аналіз предметної області	05.02.2024	08.02.2024	Виконано
5.	Розробка проєктних рішень до КРБ	12.02.2024	15.02.2024	Виконано
6.	Моделювання та конструювання ПЗ	25.02.2024	29.03.2024	Виконано
7.	Кодування, тестування розробленого ПЗ, аналіз результатів тестування	30.03.2024	26.04.2024	Виконано
8.	Розробка спеціальної частини з охорони праці	16.05.2024	22.05.2024	Виконано
9.	Оформлення КРБ та презентації	24.05.2024	28.05.2024	Виконано
10.	Відгук керівника КРБ	29.05.2024	31.05.2024	Виконано
11.	Попередній захист	03.06.2024	05.06.2024	Виконано
12.	Завершення оформлення КРБ та презентації	06.06.2024	09.06.2024	Виконано
13.	Рецензування	15.06.2024	17.06.2024	Виконано
14.	Захист кваліфікаційної роботи	24.06.2024	27.06.2024	Виконано

Розробив здобувач Шевченко Олександр Олександрович _____

(прізвище, ім'я, по батькові) *(підпис)*

«1» лютого 2024 р.

Керівник роботи канд. техн. наук, доцент Давиденко Є. О. _

(посада, прізвище, ім'я, по батькові) *(підпис)*

«2» лютого 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Real-time чат-месенджер»

Здобувач 408 гр.: Шевченко Олександр Олександрович

Керівник: канд. техн. наук, доцент Давиденко Є. О.

Актуальність теми кваліфікаційної роботи визначається зростаючою потребою користувачів у миттєвому спілкуванні та обміні інформацією. Швидкий розвиток технологій вносить свої вимоги до месенджерів, зокрема у сфері електронної комунікації та інтерактивного спілкування.

Кваліфікаційна робота присвячена розробці програмного забезпечення-вебзастосунка real-time чат-месенджера для забезпечення зручної та ефективної комунікації користувачів у режимі реальному часі.

Об’єкт роботи: процес створення чат-месенджеру, що працює в реальному часі з використанням сучасних технологій.

Предмет роботи: програмні засоби та інструменти розробки сучасних вебзастосунків та технологій, що дозволяють реалізувати реально-часовий чат-месенджер.

Мета: розробка та впровадження вебзастосунка real-time чат-месенджера для забезпечення зручної та ефективної комунікації користувачів.

Кваліфікаційна робота бакалавра складається з вступу, чотирьох розділів, висновків та переліку джерел посилання.

У вступі визначається актуальність теми, що приймається за мету та невеликий огляд поставленої задачі, предмет дослідження та об’єкт роботи.

У першому розділі описується аналітична частина, тобто огляд чинних застосунків-аналогів вебмесенджерів, визначення функціонала, переваг та недоліків програмного забезпечення, обґрунтовується план виконання завдання. Наступною частиною розділу є формування та опис специфікації вимог до програмного забезпечення, що розробляється.

У другому розділі описується процес розробки проєктних рішень, що

забезпечують виконання пунктів специфікації вимог до програмного забезпечення, тобто моделювання об’єкта та предмету дослідження, а також функціональні та інформаційні моделі програмного забезпечення. В результаті процесу розробки складено детальний алгоритм вирішення поставленої задачі. У третьому розділі описується результат виконаної роботи з конструювання та моделювання програмного забезпечення, включаючи вибір технологій, мови програмування, та компонентів застосунку, тобто бібліотек, плагінів та ін, розробку UML-діаграм та опис інтерфейсу.

У четвертому розділі демонструється проведена робота з кодування та тестування вебзастосунка real-time чат-месенджера, крім того, описується аналіз отриманих результатів під час тестування програмного забезпечення.

У висновках проводиться аналіз роботи та отриманих результатів.

Кваліфікаційна робота бакалавра викладена на 68 сторінок, вона містить 4 розділи, 35 ілюстрацій, 12 таблиць, 24 джерел в переліку посилань.

Ключові слова: *real-time чат-месенджер, створення вебзастосунку, розробка програмного забезпечення, Вебзастосунок для спілкування , клієнт-серверна архітектура, вебтехнології, комунікація в онлайн режимі, месенджер для спілкування, Інтерфейс користувача* .

ABSTRACT

of the Bachelor's Thesis

"Real-time chat messenger" Student: Shevchenko Oleksandr

Supervisor: Candidate of Technical Sciences (Ph. D.), Associate Professor Davydenko Y. O.

The relevance of this qualification work is determined by the increasing need of users for instant communication and information exchange. The rapid development of technologies imposes its requirements on messengers, particularly in the field of electronic communication and interactive communication. This work is dedicated to developing web application software, a real-time chat messenger, to provide convenient and effective user communication in real-time mode.

The object of the work is the process of creating a real-time chat messenger using modern technologies.

The subject of the work is the software tools and development instruments for modern web applications and technologies that enable the implementation of a real-time chat messenger.

The aim is to develop and implement a web application for real-time chat messaging to ensure convenient and effective user communication.

The bachelor's qualification work consists of an introduction, four chapters, conclusions, and a list of references.

The introduction defines the relevance of the topic, the purpose, a brief overview of the set task, the subject of research, and the object of work.

The first chapter describes the analytical part, including a review of current analogs of web messengers, defining the functionality, advantages, and disadvantages of software, and justifying the execution plan. The next part of the chapter is the formation and description of the software requirements specification.

The second chapter describes the process of developing project solutions to fulfill the points of the software requirements specification, such as object and subject modeling, functional and informational software models. A detailed

algorithm for solving the set task is compiled as a result of the development process.

The third chapter describes the results of the construction and modeling of the software, including the selection of technologies, programming languages, application components such as libraries, plugins, etc., the development of UML diagrams, and the interface description.

The fourth chapter demonstrates the work of coding and testing the real-time chat messenger web application. Additionally, it describes the analysis of the obtained results during the software testing process.

The conclusions analyze the work and the obtained results.

The bachelor's qualification work is presented in 68 pages, it consists of 4 chapters, 35 illustrations, 12 tables, and 24 references in the list of citations.

Keywords: *real-time chat messenger, web application development, software development, web application for communication, client-server architecture, web technologies, online communication, messenger for communication, user interface.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД	–	база даних
КРБ	–	кваліфікаційна робота бакалавра
ОС	–	операційна система
ПЗ	–	програмне забезпечення
ПК	–	персональний комп’ютер
ЦА	–	цільова авдиторія
API	–	application programming interface
CSS	–	cascading style sheets
HTML	–	hypertext markup language
JS	–	JavaScript
React	–	JavaScript library for creating interactive UI
Next JS	–	React framework for server-side
Tailwind	–	CSS framework for rapid UI
SQL	–	structured query language
NoSQL	–	not only SQL
UI	–	user interface
UX	–	user experience
Web app	–	web application
JSON	–	JavaScript object notation
MongoDB	–	Scalable NoSQL database storing data in JSON-format
JWT		JSON Web Token

ВСТУП

Актуальність теми. В контексті сучасних технологій та змін у способах комунікації зростає потреба у реально-часових комунікаційних рішеннях для ефективної взаємодії та обміну інформацією між людьми та організаціями. Чат-месенджери дозволяють користувачам взаємодіяти у реальному часі, що є важливим для покращення ефективності та зручності комунікації, крім того, у сучасних реаліях воєнного стану дуже важливо тримати зв’язок зі своїми рідними та близькими людьми. Реалізація Real-time чат-месенджера відкриває можливості для ефективного та швидкого обміну повідомленнями, полегшує спілкування в онлайн-режимі та забезпечує доступ до інформації в реальному часі.

Об’єктом роботи є процес створення чат-месенджеру, що працює в реальному часі з використанням сучасних технологій, завдяки яким це можливо реалізувати.

Предметом кваліфікаційної роботи є інструменти розробки сучасних вебзастосунків та технологій, які дозволяють реалізувати реально-часовий чат-месенджер, їхні особливості, переваги та недоліки

Метою кваліфікаційної роботи є розробка та впровадження вебзастосунка real-time чат-месенджера для забезпечення зручної та ефективної комунікації користувачів.

Для досягнення поставленої мети необхідно виконати наступний перелік завдань:

1) дослідити предметну галузь, згідно з обраною темою КРБ;

2) аналіз переваг та недоліків існуючих чат-месенджерів;

3) розробка інтерфейсу, який буде одночасно зручним, звичним та функціональним;

4) спроєктувати та змоделювати архітектури вебзастосунка;

5) інтеграція сучасних технологій шифрування та хмарних сервісів для обробки та зберігання даних;

6) забезпечити безпеки та конфіденційність даних користувачів, які будуть використовувати застосунок;

7) провести тестування та вдосконалення застосунку згідно з отриманими результатами;

Сфера застосування: реально-часовий чат-месенджер може бути корисним для різноманітних сфер діяльності, таких як:

1) комунікація між членами сім'ї та друзями під час воєнного стану, коли зв'язок з рідними та близькими особами важливий;

2) комунікація між колегами та клієнтами в організаціях, щоб покращити ефективність та зручність комунікації;

3) комунікація між учасниками онлайн-громад, щоб обмінюватися інформацією та поділяти досвід;

4) комунікація між клієнтами та підтримкою клієнтів, щоб надавати ефективну та швидку підтримку;

5) комунікація між учасниками онлайн-навчання, щоб покращити взаємодію та обмін інформацією.

1 АНАЛІЗ ЗАСТОСУНКІВ РЕАЛЬНО ЧАСОВИХ ЧАТ МЕСЕНДЖЕРІВ

1.1 Огляд застосунків-аналогів Web-base real-time чат месенджерів

Для визначення ключових вимог до системи розробки було проведено ретельний аналіз існуючих подібних систем. Цей аналіз передбачає вивчення сильних та слабких сторін існуючих рішень для досягнення наступних цілей:

1) Розуміння позицій конкурентів на ринку: практично кожен застосунок має конкурентів. Тому дуже важливо розуміти їхні позиції і те, які функції вони пропонують. Аналізуючи конкурентів, можна визначити можливості для вдосконалення власного продукту та випередити конкурентів. Крім того, важливо постійно стежити за оновленнями та змінами на ринку.

2) Підвищення лояльності клієнтів: аналізуючи як близьких, так і далеких конкурентів (наприклад, іноземних або близьких за нішею), можна отримати ідеї для розвитку продукту. В результаті буде складено список нових функцій, які можна додати, щоб покращити продукт, зробити його максимально цікавим для потенційних користувачів системи. і підвищити лояльність клієнтів.

3) Аналіз подібних систем дає уявлення про сильні та слабкі сторони продукту, що дозволяє отримати повне уявлення про його можливості та сфери для вдосконалення. Ця інформація може бути використана для розробки інноваційних рішень і розширення функціональності продукту, що потенційно дасть йому конкурентну перевагу. Крім того, аналіз може виявити пропозиції щодо покращення застосування продукту та впровадження нових функцій [1].

LiveChat

LiveChat – це програмне забезпечення для онлайн-чату, яке використовується для покращення обслуговування клієнтів на вебсайтах (рис. 1.1).

Таблиця 1.1 – Опис «LiveChat»

Назва	**LiveChat**
Архітектура	Web-based
Виробник	LiveChat Inc.
Мова реалізації	JavaScript, PHP, Python

Кінець таблиці 1.1

Функції	Чат в реальному часі; підтримка клієнтів; інтеграція з електронною комерцією; інтеграція чат-бота; інтеграція довідкового центру; гейміфікація для агентів; можливості кастомізації; підтримка інтеграції зі Slack та Teams; інтеграція з Social Intents;
Переваги	Повнофункціональний живий чат; швидке та відносно просте налаштування; масштабованість для зростаючого бізнесу; підтримка 24/7 та велика база знань; підвищення конверсії для бізнесу; зручний інтерфейс;
Недоліки	Платний; складність налаштування та використання без відповідних технічних знань.
Вебсайт	https://www.livechat.com/

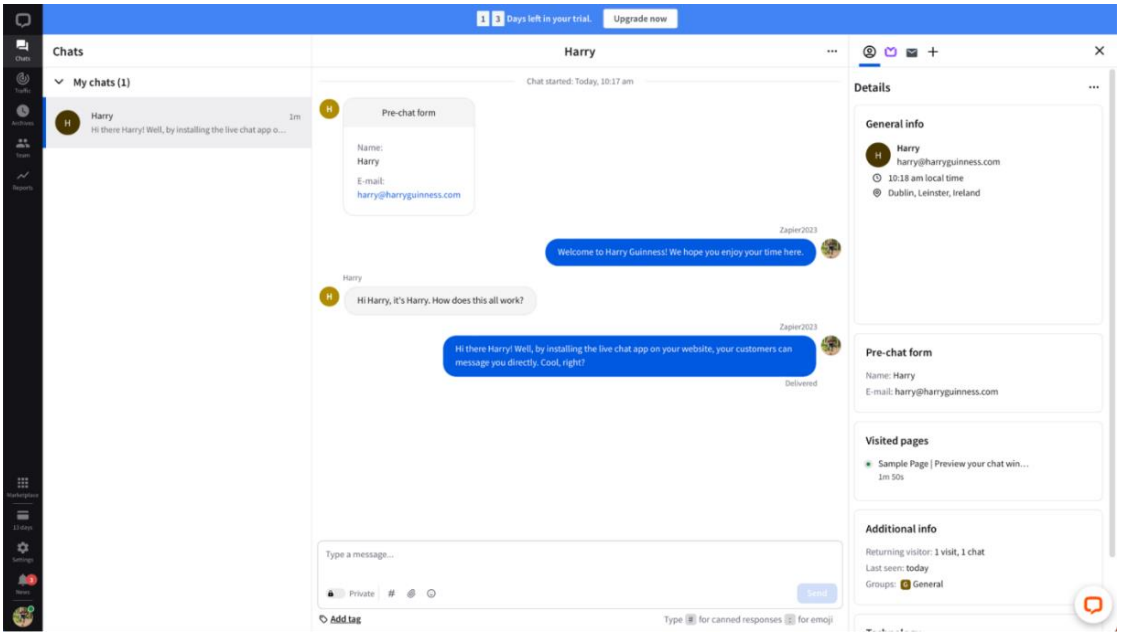

Рисунок 1.1 – Вигляд інтерфейсу застосунку «LiveChat»

Tawk.to

Tawk.to – це безкоштовний застосунок для онлайн-чату, який дозволяє спілкуватися з відвідувачами вебсайту (рис. 1.2). Він пропонує необмежену кількість агентів і чатів, що робить його гарним рішенням для малого бізнесу.

Таблиця 1.2 – Опис «Tawk.to»

Назва	**Tawk.to**
Архітектура	Web-based
Виробник	Tawk.to
Мова реалізації	JavaScript
Функції	Чат в реальному часі; груповий обмін повідомленнями; відео та голосова підтримка; керування тікетами клієнтської підтримки; інтеграція з CRM; створення довідкового центру; налаштовуваний віджет чату; mobile SDK для вбудовування підтримки в мобільні застосунки; прямий доступ до RESTful JSON API;
Переваги	Безкоштовно для необмеженої кількості агентів; цілодобова підтримка в чаті; глобально розподілена архітектура для швидшої доставки повідомлень; інтеграція з різними платформами та системами автентифікації
Недоліки	Відсутність можливості локального хостингу; брендування, за видалення якого потрібно платити, є більш нав'язливим, ніж більшість інших; менше інтеграцій, ніж у багатьох інших рішеннях
Вебсайт	https://www.tawk.to/

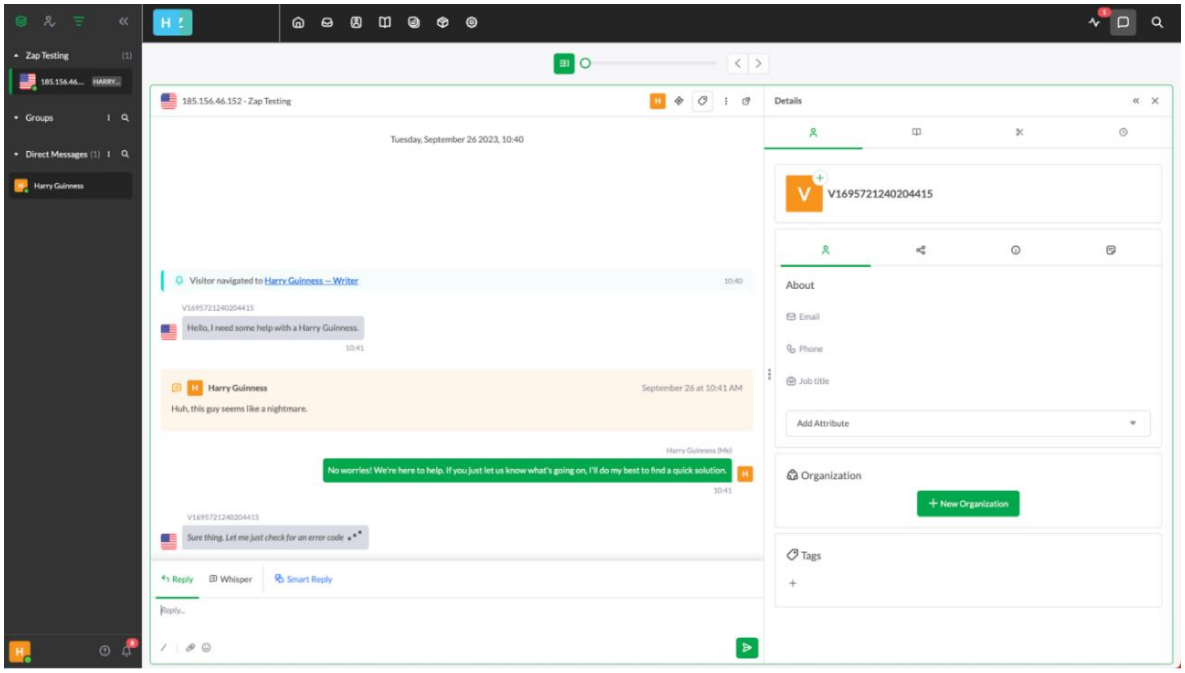

Рисунок 1.2 – Вигляд інтерфейсу застосунку «Tawk.to»

Re:amaze

Re:amaze – хмарний сервіс для підтримки клієнтів, що поєднує в собі функції чату, управління повідомленнями та бази знань (рис. 1.3). Він допомагає бізнесу підтримувати зв'язок з клієнтами через різні канали.

Таблиця 1.3 – Опис «Re:amaze»

Назва	**Re:amaze**
Архітектура	Web-based
Виробник	Re:amaze
Мова реалізації	JavaScript
Функції	Чат в реальному часі; різноманітні методи інтеграції (SDK, RESTful JSON API); настроювані лайтбокси підтримки; вбудовані бази знань; контактні форми, що випадають; мобільний SDK для вбудовування підтримки в мобільні застосунки; API кастомних модулів для вбудовування користувацького інтерфейсу підтримки

Кінець таблиці 13

Переваги	Гнучкі можливості інтеграції для різних вебсайтів і застосунків; простота використання для вбудовування підтримки в сайти та застосунки; налаштовуваний інтерфейс підтримки
Недоліки	Складність налаштування та використання без відповідних технічних знань.
Вебсайт	https://www.reamaze.com/

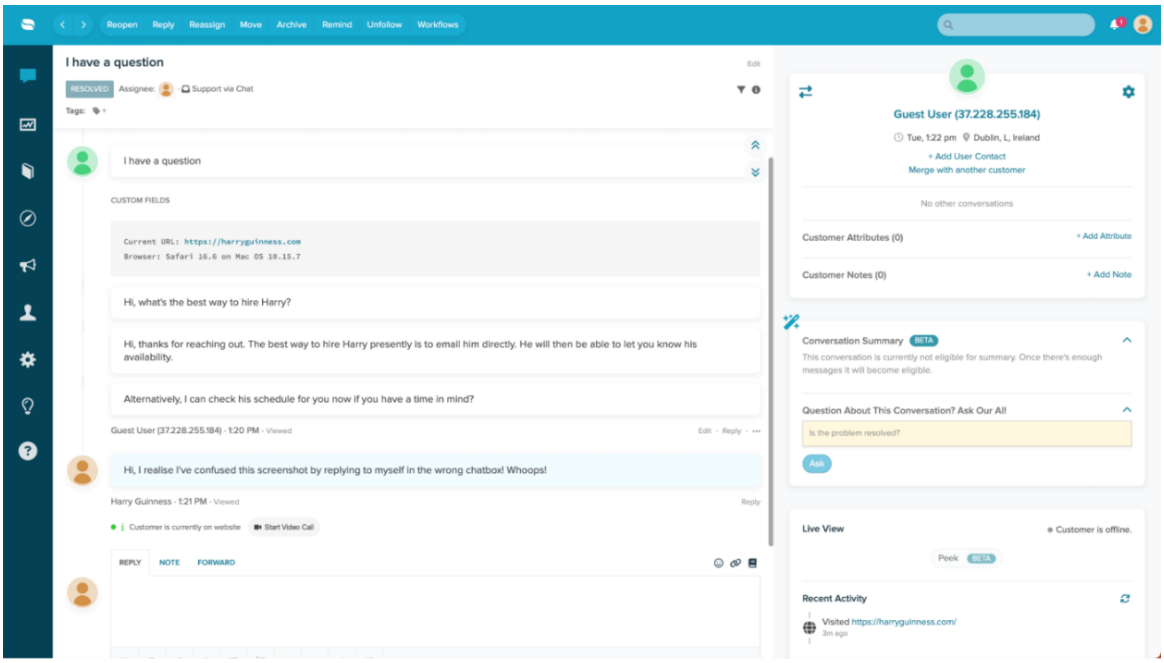

Рисунок 1.3 – Вигляд інтерфейсу застосунку «Tawk.to»

Zendesk

Zendesk – це хмарна платформа для обслуговування клієнтів, яка допомагає компаніям налагодити зв'язок з клієнтами через різні канали (рис. 1.4).

Таблиця 1.4 – Опис «Zendesk»

Назва	**Zendesk**
Архітектура	Web-based
Виробник	Zendesk
Мова реалізації	Не вказано

Кінець таблиці 1.4

Функції	Чат в реальному часі; керування тикетами; інтеграція з базою знань; аналітика клієнтської підтримки; автоматизовані відповіді чат-ботів; інструменти для спільної роботи агентів; інтеграція з різними платформами;
Переваги	Гнучкі можливості інтеграції для різних вебсайтів і застосунків простота використання для вбудовування підтримки в сайти; налаштовуваний інтерфейс підтримки
Недоліки	Платний; відносна складність налаштування та використання.
Вебсайт	https://www.zendesk.com/

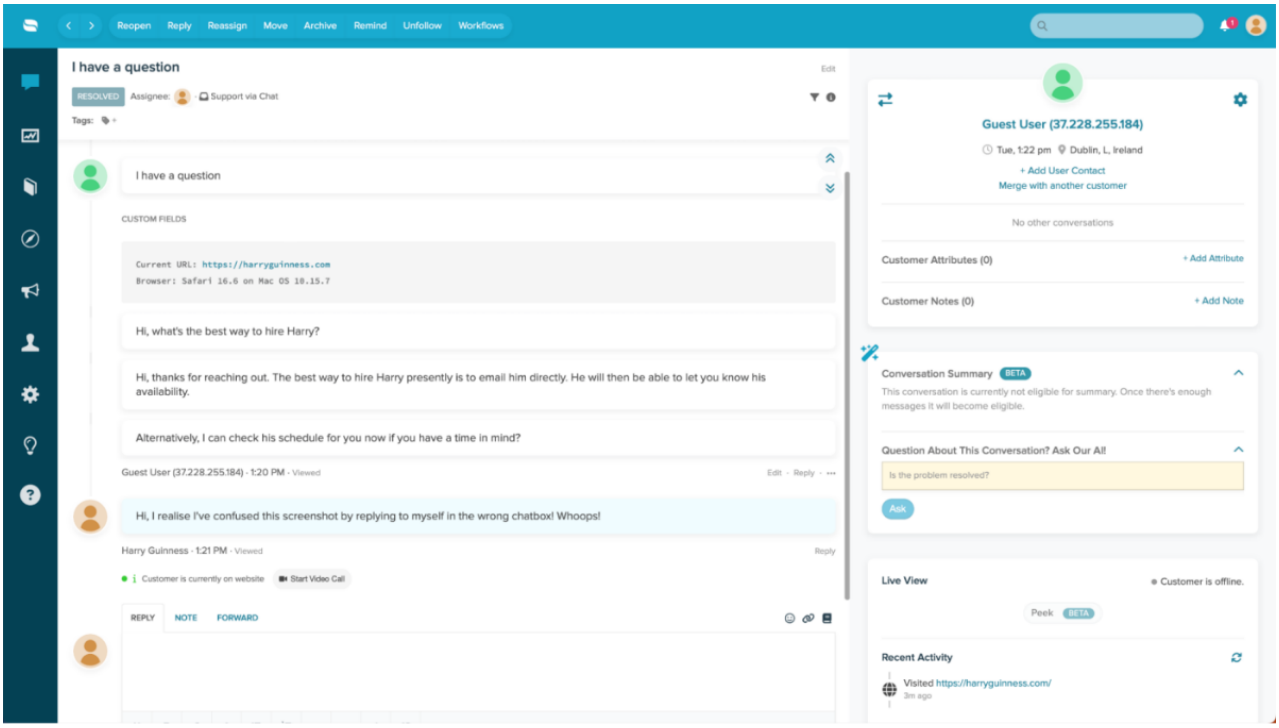

Рисунок 1.4 – Вигляд інтерфейсу застосунку «Zendesk»

1.2 Аналіз системи, що розробляється

Розробка реально-часового чат-месенджеру з використанням новітніх вебтехнологій призначена для ефективної взаємодії та обміну інформацією між користувачами. Користувачі мають потребу в швидкій та зручній комунікації, мати можливість бути на зв’язку, додавати нових учасників у чат-месенджер [2].

Призначення системи-застосунку, для якої розробляється програмне забезпечення:

Система є чат-застосунком в реальному часі для спілкування, який використовує найсучасніші технології для забезпечення швидкого, візуально привабливого, безпечного та інтерактивного досвіду для користувачів (рис. 1.5).

Система повинна забезпечувати багатокористувацький режим роботи. Інтерфейс має бути сумісним з ОС Windows (вище XP версії) та із смартфонами з ОС iOS та Android (вище версії 7.1). Система повинна бути в працездатному стані 24 години на добу, 7 днів на тиждень.

Таблиця 1.5 – Опис системи, що розробляється

Функції	1) реєстрація користувачів; 2) авторизація користувачів; 3) надсилання та отримання текстових повідомлень; 4) обмін файлами (фото, відео, документи); 5) створення групових чатів; 6) видалення чатів; 7) зміна особистих даних користувача; 8) шифрування повідомлень (з кінця в кінець); 9) перегляд інформації про інших користувачів
Користувацький інтерфейс	простий і інтуїтивно зрозумілий інтерфейс; повністю адаптивна вебверсія;
Користувачі	користувач; адміністратор групового чату

Кінець таблиці 1.5

Сценарії роботи	1) користувач авторизується або реєструється в застосунку; 2) користувач додає чи обирає іншого користувача для спілкування; 3) користувач створює новий чат з іншим користувачем; 4) користувач надсилає повідомлення – фотографію іншому користувачу; 5) користувач видаляє чат для обох користувачів; 6) користувач обирає ще двох інших користувачів і створює груповий чат з ними; 7) користувач змінює інформацію про себе в вікні налаштування (ім'я користувача, аватар); 8) користувач продивляється повідомлення в групі, про що сповіщають інших користувачів; 9) користувач видаляє груповий чат для всіх його учасників.

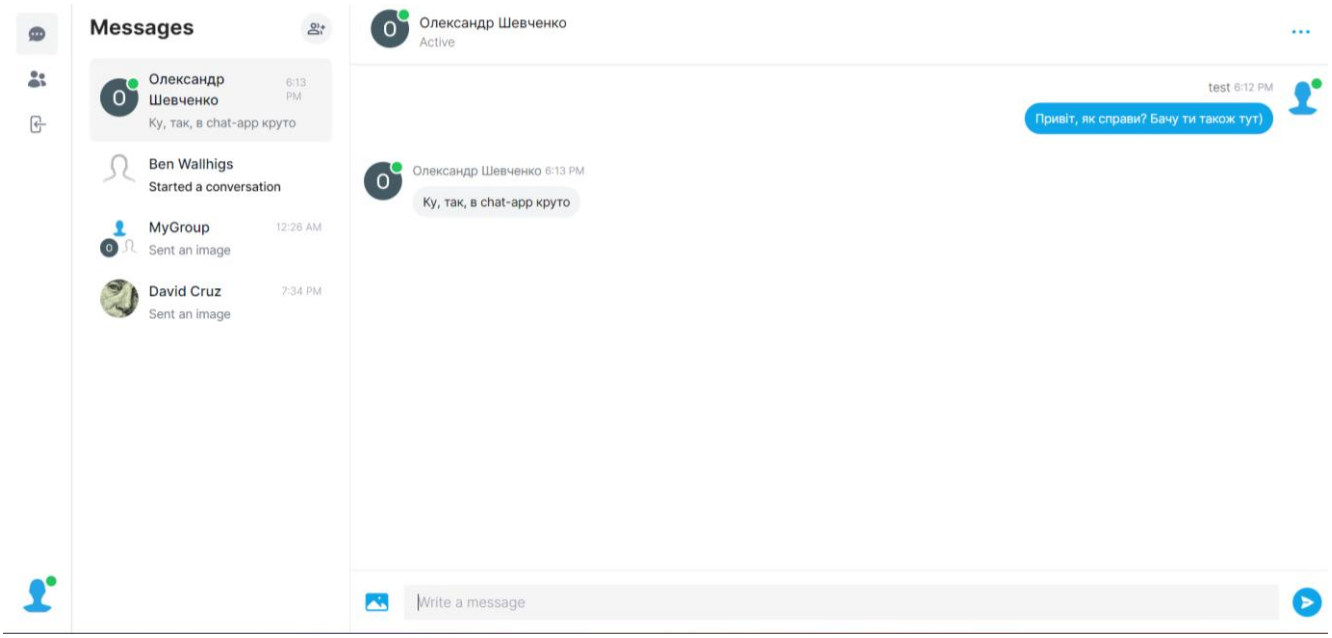

Рисунок 1.5 – Інтерфейсу застосунку «Chat-app»

1.3 Специфікація вимог до програмного забезпечення «Chat-app»

Призначення системи (застосунку), для якої розробляється програмне забезпечення

Призначенням застосунку є забезпечення ефективного і зручного спілкування між користувачами у режимі реального часу через вебзастосунок. Система дозволяє користувачам створювати чати, обмінюватися повідомленнями, додавати учасників до групових чатів, отримувати сповіщення та керувати своїм профілем.

Погодження, що ухвалені в програмній документації

Було погоджено, що для створення та злагодженої роботи системи роботи буде використовуватися допоміжний фреймворк – Next.js. Для фронтенду обрано бібліотеку React, Tailwind для стилізації, для баз даних Prisma та MongoDB, NextAuth для автентифікації користувачів та Pusher для реалізації реального часу обміну повідомленнями.

Межі проєкту ПЗ

Крайня дата завершення роботи над ПЗ – 22.06.2024р.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Система реалізована для сприяння комунікації між користувачами в режимі реального часу, зокрема у форматі чат-месенджера, який дозволяє створювати особисті та групові чати, обмінюватися повідомленнями, отримувати сповіщення та керувати профілем користувача.

Розроблений чат-месенджер в реальному часі може знайти своє застосування в різних сферах:

1. **Особисте життя:** користувачі можуть використовувати месенджер для спілкування з родиною та друзями, обміну фотографіями та відеозаписами, планування зустрічей тощо.

2. **Робоче середовище:** у бізнесі цей месенджер може використовуватися для комунікації внутрішнього колективу, організації робочих проєктів та обміну документами а також для підтримки клієнтів в чаті.

2. **Освіта:** у навчальних закладах чат може бути використаний для комунікації між викладачами та студентами, обговорення лекцій, викладання домашніх завдань тощо.

Характеристики користувачів

Основні характеристики користувачів цієї системи: наявність смартфону, планшету або ПК з доступом до мережі Інтернет. Та бажання спілкуватися та обмінюватися інформацією в режимі реального часу.

Загальна структура і склад системи

Основні частини системи включають:

1. **Сервер:** де централізовано зберігається і обробляється інформація.

2. **База даних (MongoDB з використанням Prisma):** де зберігається інформація про користувачів, чати, повідомлення тощо [5].

3. **Вебзастосунок (Next.js та React):** де користувачі можуть взаємодіяти з системою через інтерфейс користувача.

4. **API (NextAuth):** Де здійснюється автентифікація користувачів та обмін даними між клієнтською і серверною частинами.

Загальні обмеження

Основне обмеження для роботи з системою - наявність доступу до мережі Інтернет для користувачів.

ФУНКЦІЇ СИСТЕМИ ВЕБЗАСТОСУНКУ

Функція обміну повідомленнями в реальному часі

Опис функції.

Користувачі мають можливість обмінюватися повідомленнями в особистих та групових чатах в реальному часі без необхідності оновлення сторінки.

Вхідна і вихідна інформація

Вхідна інформація – текстове повідомлення, ідентифікатор чату.

Вихідна інформація – відправлене та отримане повідомлення.

Функціональні вимоги

Відправлення та отримання повідомлень без перезавантаження сторінки.

Синхронізація повідомлень між усіма учасниками чату в реальному часі.

Збереження історії повідомлень у базі даних.

Функція створення та керування чатами

Опис функції

Функція дозволяє користувачам створювати нові чати, додавати учасників та керувати чатом (видаляти для всіх учасників).

Вхідна і вихідна інформація

Вхідна інформація – дані про учасників чату та налаштування.

Вихідна інформація – створений чат та його параметри та учасники.

Функціональні вимоги

Збереження даних про чати та учасників у базі даних. Можливість додавання та видалення чату. Відображення списку чатів для користувача з належними правами доступу.

Функція реєстрації та авторизації користувачів

Опис функції

Функція дозволяє користувачам створювати облікові записи та входити в систему, забезпечуючи безпеку та доступ до основних функцій чат-месенджера.

Вхідна і вихідна інформація

Вхідна інформація – дані про учасників чату та налаштування.

Вихідна інформація – створений чат та його параметри та учасники.

Функціональні вимоги

Збереження даних про чати та учасників у базі даних. Можливість додавання та видалення чату. Відображення списку чатів для користувача з належними правами доступу.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела та зміст вхідної інформації (даних)

Джерела даних:

– користувачі чат-месенджера, які надсилають текстові повідомлення, фотографії, відео тощо;

– системні повідомлення та дані, які генеруються месенджером, такі як статуси користувачів, повідомлення про надходження повідомлень;

– зовнішні джерела, такі як інтеграція з іншими застосунками або сервісами, які можуть надсилати або отримувати дані через API.

Зміст вхідної інформації:

– текстові повідомлення, які надсилають користувачі;

– медіафайли, такі як зображення, відео, аудіо, які користувачі можуть надсилати один одному;

– метадані, такі як час надходження повідомлень, статуси користувачів.

Вимоги до способів організації, збереження та ведення інформації

Організація даних:

– дані про користувачів, їх повідомлення та інші дані повинні бути організовані у відповідності до логічних структур, таких як таблиці або колекції;

– класифікація даних за категоріями, такими як повідомлення, контакти, групи, налаштування.

Збереження даних:

– використання бази даних MongoDB для зберігання інформації про користувачів, повідомлення, медіафайли та інші дані;

– забезпечення резервного копіювання та відновлення даних для підтримки надійності та безпеки.

Ведення інформації:

– запис та оновлення даних про користувачів, їх повідомлення, статуси у реальному часі для забезпечення актуальності інформації;

– відстеження та логування дій користувачів для аналізу активності та виявлення проблем.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Для застосунку не передбачено жорстких обмежень. Він повинен працювати у будь-якому сучасному браузеру зі стабільним інтернет з’єднанням.

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

– База даних: Нереляційна база даних MongoDB.

– Браузерний клієнт: Вебзастосунок.

Системне програмне забезпечення

– Платформа: Node.js для вебзастосунку.

– Фреймворки: Next.js 14 та React для реалізації фронтенду .

– Бібліотеки: Tailwind CSS для стилізації інтерфейсу, Prisma для зв'язку з базою даних MongoDB.

Мережне програмне забезпечення

– Протоколи: HTTPS для захищеної передачі даних між клієнтом і сервером.

– Обмін даними: використання RESTful API для взаємодії між клієнтом і сервером для обміну повідомленнями в реальному часі.

Програмне забезпечення для забезпечення роботи в реальному часі

– Інтеграція з Pusher для забезпечення реально-часової спілкування та оновлення інтерфейсу без перезавантаження сторінки.

Технічні вимоги

– Система повинна підтримувати сучасні версії браузерів Chrome, Firefox, Safari.

– Мінімальні вимоги до апаратного забезпечення: процесор Intel Core i5 або еквівалент, ОЗП не менше 4 ГБ, мінімум 20 ГБ вільного місця на диску, доступ до Інтернету з мінімальною швидкістю зв'язку 10 Мбіт/с.

ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

Інтерфейс користувача

Інтерфейс повинен бути інтуїтивно зрозумілим і зручним, забезпечувати легке використання функцій.

Апаратний інтерфейс

Відповідність принципам UX/UI для забезпечення узгодженого користувацького досвіду, з використанням сучасних візуально привабливих компонентів з бібліотеки Tailwind CSS.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

Забезпечення підтримки широкого спектру пристроїв за рахунок оптимізації інтерфейсу та функціоналу для зручного використання на різних розмірах екранів та типах пристроїв.

Продуктивність

Забезпечення швидкого часу відгуку системи для миттєвої взаємодії з користувачем. Оптимізація завантаження сторінок та обробки запитів для мінімізації затримок.

Надійність

Надійне збереження даних користувачів та інформації про замовлення в базі

даних MongoDB з використанням технологій Prisma для ORM. Резервне копіювання та відновлення даних для уникнення втрат інформації у випадку непередбачуваних ситуацій.

Безпека

Шифрування конфіденційних даних з використанням сучасних алгоритмів шифрування. Використання JWT токенів для авторизації та управління сесіями користувачів з метою захисту від несанкціонованого доступу до даних.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи проведено докладний огляд існуючих реально-часових web-based чат-месенджерів та аналіз їх функцій. Аналіз цих застосунків дав можливість визначити сильні та слабкі сторони існуючих рішень, а також визначити потреби користувачів у таких застосунках. Також було проведено аналіз системи, що розробляється, в якому визначено типи користувачів, основний функціонал та сценарії використання вебзастосунку. В цьому розділі сформовано вимоги до програмного забезпечення «chat-app», детально описано його призначення та функціонал, визначено архітектуру та технології, необхідні для розробки застосунку. Розділ містить специфікацію вимог до програмного забезпечення, включаючи детальний опис як функціональних, так і нефункціональних вимог, для визначення взаємодії користувачів з програмним забезпеченням.

2 МОДЕЛЮВАННЯ ФУНЦІОНАЛЬНОСТІ ЗАСТОСУНКУ

2.1 Етапи реалізації проєкту

Розробка вебзастосунку real-time чат-месенджеру «Chat-app» включає розробку дизайну, створення інтерфейсу та реалізацію функціоналу серверної частини, що може виконуватися як одним розробником, так і командою фахівців. Тривалість розробки залежить від складності проєкту, розміру команди та досвіду фахівців.

Етапи розробки включають:

1. Формування концепції продукту: Збір інформації про продукт та формування переліку вимог і задач (product backlog). В цьому етапі важливо відповісти на такі питання:

1) Які проблеми має вирішувати застосунок?

2) Який результат очікується від розробки?

3) Який відведено час на розробку?

4) Який бюджет на розробку?

5) Який функціонал планується?

6) Чи існують які-небудь обмеження у використанні?

7) Які технології використовуватимуться?

8) Хто є цільовою аудиторією застосунку?

2. Проєктування архітектури: проєктування архітектури системи з розробкою UML-діаграм, таких як діаграми класів, варіантів використання, послідовності та розгортання. Цей етап визначає структуру системи, що дає чітке уявлення про компоненти та їх взаємодію [6].

3. Розробка макетів: створення дизайну інтерфейсу відповідно до вимог. Розробка макетів включає розробку логотипу, вибір шрифтів та колірної гами. Макети допомагають візуалізувати користувацький досвід та інтерфейс застосунку.

4. Розробка front-end та back-end частин: цей етап включає реалізацію інтерфейсу адаптивного вебзастосунку та серверної частини з необхнідним функціоналом.

5. Тестування та запуск: проведення тестування для виявлення помилок та покращення стабільності застосунку. Після успішного тестування продукт готовий до запуску на ринок.

6. Видання та технічне обслуговування: піісля запуску необхідно регулярно оновлювати застосунок для підтримки актуального функціоналу, виправлення помилок та покращення продуктивності, крім того важливо покращувати функціонал застосунку.

2.2 Створення Use Case

Use Case – це опис послідовності дій, відповідно до якого користувач взаємодіє з системою для виконання певного завдання залежно від функціоналу застосунку. Use Case описує, що система повинна робити, а не як вона це робить.

Короткий Use Case: Надсилання повідомлень

Користувач обирає іншого користувача, якому хоче написати. Система відкриває вікно чату з контактами. Користувач обирає отримувача та вводить текст повідомлення. Після натискання кнопки «Надіслати» система доставляє повідомлення отримувачеві.

Поверхневий Use Case: Створення групового чату

Головний сценарій (успішний):

1. Користувач відкриває вебзастосунок та авторизується в обліковому записі.
2. Користувач натискає на опцію створення нового чату.
3. Система відображає форму створення чату, де користувач може вказати назву чату та обрати учасників яких хоче додати.
4. Користувач заповнює форму, вказуючи назву чату та обираючи учасників зі списку контактів.
5. Користувач натискає кнопку «Створити».
6. Система створює новий груповий чат з вказаними учасниками.
7. Усі учасники отримують сповіщення про створення чату та автоматично додаються до нього.

Альтернативні сценарії:

1. Користувач не вказує назву чату. Система повідомляє про необхідність вказати назву та пропонує заповнити це поле.

2. Користувач не обирає жодного учасника, або меньше аніж два. Система повідомляє про необхідність додати хоча б одного учасника та пропонує обрати учасників зі списку контактів, повідомляє користувачу про помилку.

3. Користувач натискає кнопку «Скасувати» на формі створення чату, або натискає за межами форми. Система відміняє створення чату та повертає користувача до попереднього екрану.

Повний Use Case: Видалення чату

Таблиця 2.1 – Сценарій видалення чату

Use Case Name	Видалення чату
Scope	System
Level	User-goal
Primary Actor	Зареєстрований користувач
Stakeholders and interests	Користувач хоче видалити чат з групою користувачів
Preconditions	Користувач повинен бути авторизований у системі та мати відповідні права (бути його адміністратором або засновником)
Success guarantee	Груповий чат успішно видалено і учасники більше не можуть обмінюватися повідомленнями в ньому.
Main Success Scenario	1. Користувач відкриває список чатів. 2. Користувач обирає груповий чат, який хоче видалити. 3. Користувач натискає кнопку «Видалити чат». 4. Система показує підтвердження видалення. 5. Користувач підтверджує видалення. 6. Система видаляє груповий чат.
Extensions:	
1a.	Якщо користувач не авторизований, система показує повідомлення про необхідність авторизації.
2a	Якщо в користувача недостатньо прав на видалення чату система приховує цю функцію від нього.
Special Requirements	Немає

Кінець таблиці 2.1

Technology and Data Variations List	Використання Pusher для взаємодії в реальному часі.
Frequency of Occurrence	Часто

Повний Use Case: Створення групового чату

Таблиця 2.2 – Сценарій створення групового чату

Use Case Name	Створення групового чату
Scope	System
Level	User-goal
Primary Actor	Зареєстрований користувач
Stakeholders and interests	Користувач хоче створити новий груповий чат з вказаними учасниками.
Preconditions	Користувач повинен бути авторизований у системі.
Success guarantee	Груповий чат успішно створено, сповіщено про це його учасників і усі вони можуть обмінюватися повідомленнями
Main Success Scenario	1. Користувач натискає на опцію створення нового чату. 2. Система відображає форму створення чату, де користувач може вказати назву чату та додати учасників. 3. Користувач вводить назву чату та обирає учасників зі списку контактів. 4. Після вибору учасників користувач натискає кнопку «Створити». 5. Система перевіряє, чи вказані учасники є користувачами системи та чи не вже є у них груповий чат з такою ж назвою. 6. Якщо усі умови виконані, система створює новий груповий чат та додає усіх учасників до нього. 7. Усі учасники отримують сповіщення про створення чату та автоматично додаються до нього. 8. Система відображає повідомлення про успішне створення чату.
Extensions:	
1a.	Якщо користувач не авторизований, система показує повідомлення про необхідність авторизації.
Special Requirements	Немає
Technology and Data Variations List	Використання Pusher для взаємодії в реальному часі.
Frequency of Occurrence	Дуже часто

Повний Use Case: Зміна інформації про користувача

Таблиця 2.3 – Сценарій зміни інформації про користувача

Use Case Name	Зміна інформації користувача
Scope	System
Level	User-goal
Primary Actor	Зареєстрований користувач
Stakeholders and interests	Користувач хоче змінити своє ім'я та фото.
Preconditions	Користувач повинен бути авторизований у системі.
Success guarantee	Інформація про користувача успішно змінена, всі учасники можуть бачити ці зміни
Main Success Scenario	1. Користувач відкриває свій профіль. 2. Користувач натискає кнопку «Редагувати профіль». 3. Користувач вводить нове ім'я та вибирає нове фото. 4. Користувач натискає кнопку «Зберегти зміни». 5. Система зберігає нову інформацію про користувача.
Extensions:	
1a.	Якщо користувач не авторизований, система показує повідомлення про необхідність авторизації.
Special Requirements	Немає
Technology and Data Variations List	Використання Pusher для взаємодії в реальному часі, MongoDB для зберігання інформації.
Frequency of Occurrence	Дуже висока

2.3 Алгоритм роботи програмного забезпечення

Вебзастосунок має бути зручним та зрозумілим для широкого кола користувачів. Користувач повинен мати змогу легко скористатися застосунком на смартфоні та ПК. Програмне забезпечення покликане спростити процес спілкування в реальному часі та зробити його зручним.

Функціонал повинен залишатися актуальним і корисним для всіх користувачів. Для наочного відображення алгоритму роботи програмного забезпечення створено чотири діаграми діяльності, які демонструють послідовність дій, що виконуються під час взаємодії користувача із застосунком:

1) *Надсилання повідомлень* (рис. 2.1). Діаграма показує основні етапи відправки повідомлення.

Діаграма показує основні етапи процесу надсилання повідомлень.

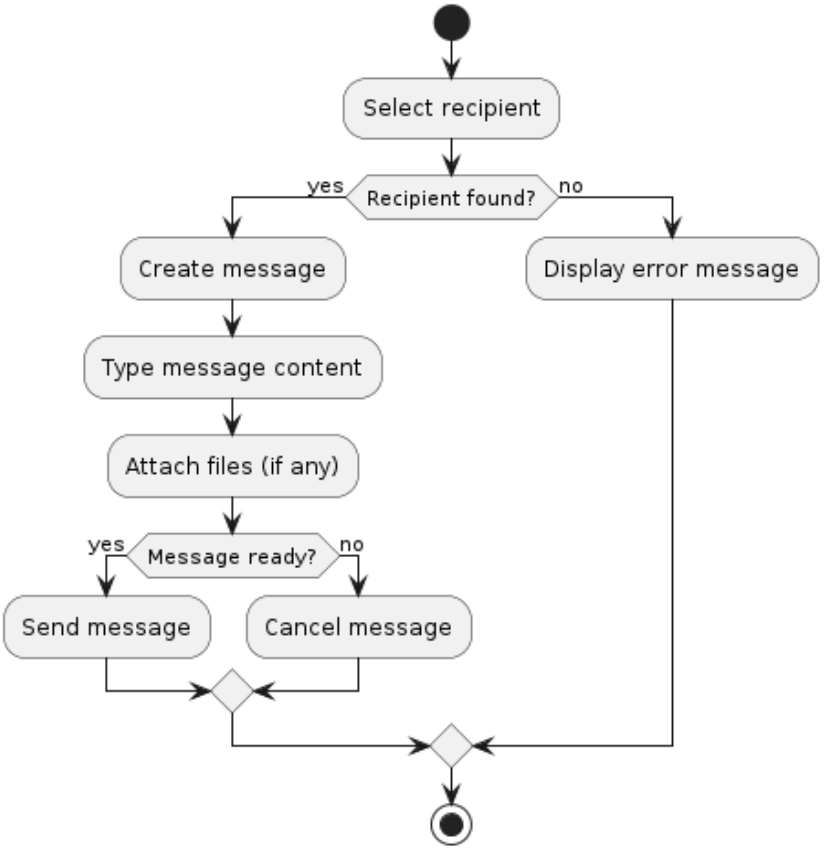

Рисунок 2.1 – Діаграма діяльності для функції надсилання повідомлення

2) *Створення групового чату* (рис. 2.2). Діаграма деталізує процес створення групового чату у вебзастосунку

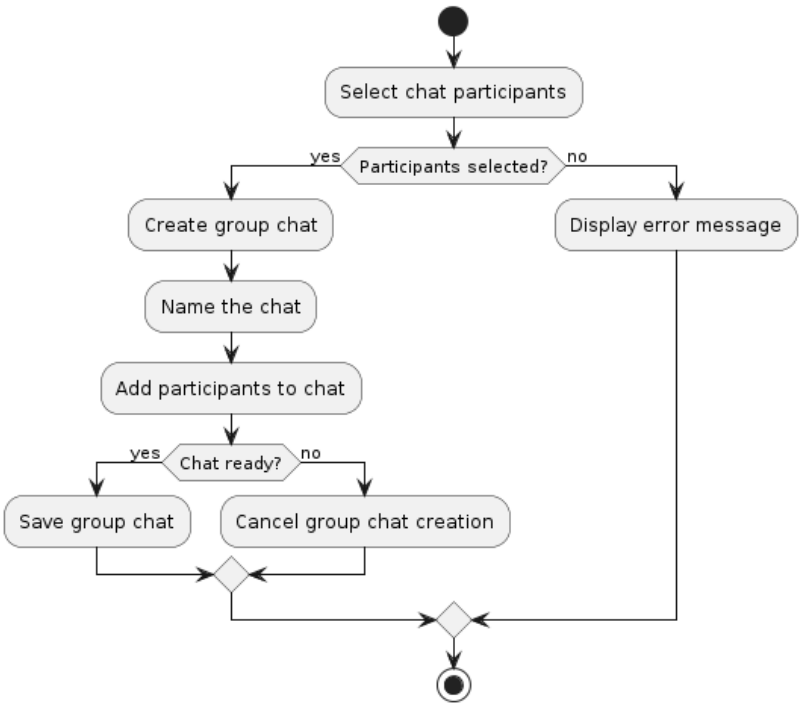

Рисунок 2.2 – Діаграма діяльності для створення групового чату

3) *Видалення створеного групового чату* (рис. 2.3). Діаграма деталізує процес видалення групового чату у вебзастосунку

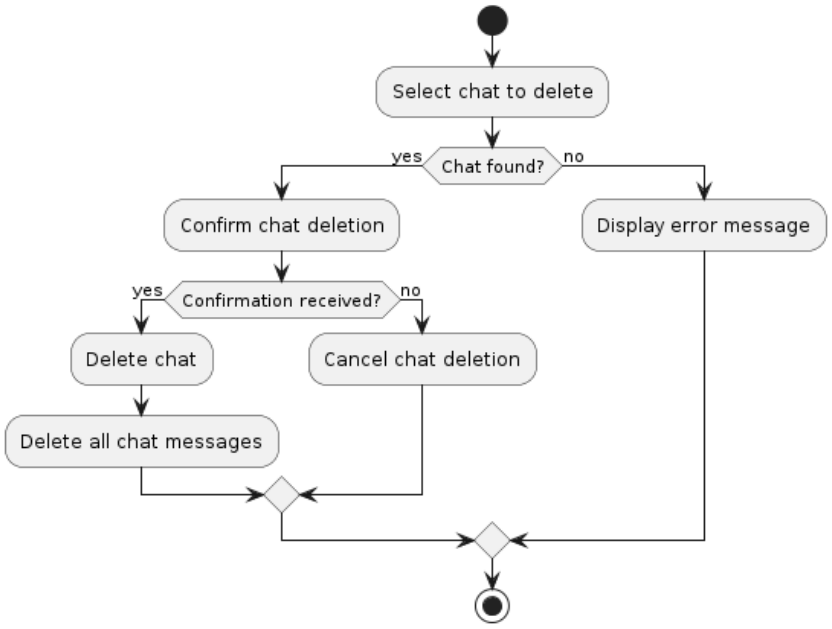

Рисунок 2.3 – Діаграма діяльності для видалення групового чату

4) *Узагальнена спрощена діаграма діяльності для користування застосункому* (рис. 2.4). Діаграма деталізує процес взаємодії користувача з Real-Time месенджером «Chat-app» у вебзастосунку.

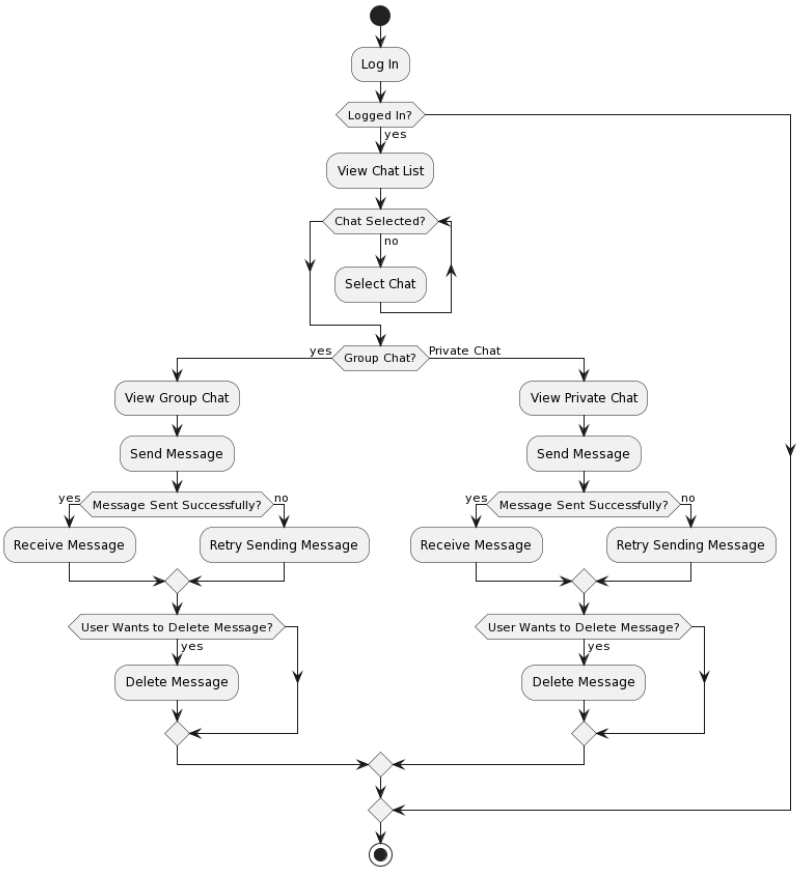

Рисунок 2.4 – Загальна діаграма діяльності для Real-Time месенджеру

2.4 Розробка діаграми класів

Розробка діаграми класів (рис. 2.5) є ключовим етапом у моделюванні архітектури програмного забезпечення. Вона надає графічне представлення структури системи, описуючи класи, їхні атрибути, методи та взаємозв'язки між ними. Це дозволяє визначити основні компоненти програмного забезпечення та їхні взаємодії ще на етапі проєктування [7].

Основні класи, представлені на діаграмі:

1. User (Користувач): Клас представляє користувача системи чат-месенджера. Включає в себе основну інформацію про користувача, таку як ім'я, електронну пошту, зображення профілю, хешований пароль тощо.

2. Account (Обліковий запис): Клас Account представляє зв'язок між користувачем (User) та його обліковим записом в системі. Включає інформацію про тип облікового запису, постачальника, токени доступу тощо.

3. Conversation (Розмова): Клас Conversation представляє розмову або чат між користувачами. Включає в себе назву розмови, дату створення, список повідомлень та учасників.

4. Message (Повідомлення): Клас Message представляє окреме повідомлення в месенджері. Містить текст повідомлення, зображення, інформацію про час створення та інших учасників.

5. Group (Група): Клас Group представляє груповий чат в системі. Містить назву групи, дату створення, список учасників тощо.

6. GroupMessage (Повідомлення групи): Клас GroupMessage представляє повідомлення, яке було відправлене в груповому чаті. Містить текст повідомлення, зображення, інформацію про час створення та відправника.

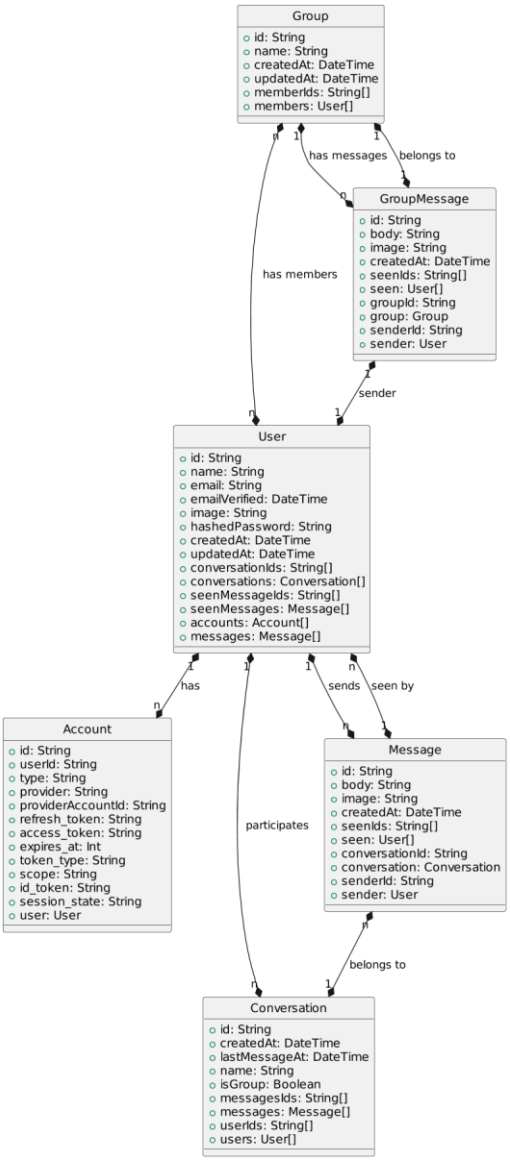

Рисунок 2.5 – Діаграма класів

Діаграма класів «Chat-app» демонструє, як різні компоненти системи співпрацюють між собою, щоб забезпечити основні функції застосунку [8].

Висновки до розділу 2

У другому розділі було розглянуто процес моделювання вебзастосунку чат-масенджеру «Chat-app», описано етапи реалізації проєкту, алгоритми роботи програмного забезпечення та розроблено діаграму класів. Це дозволило зміцнити навички в моделюванні, плануванні та проєктуванні архітектури програмного забезпечення для веббраузерів.

Розробка Use Case допомогла визначити основні сценарії використання застосунку, включаючи варіанти взаємодії користувача з функціоналом. На основі коротких і детальних сценаріїв було розроблено повні Use Case, які описують усі деталі використання програмного забезпечення.

Алгоритм роботи програмного забезпечення описав основні кроки взаємодії користувача із застосунком. Створено діаграму класів, яка дозволяє наочно побачити структуру застосунку, зрозуміти, як різні компоненти взаємодіють між собою та які функції вони виконують.

Використані UML-діаграми в цьому розділі чітко демонструють структуру застосунку, вимоги до нього, а також архітектуру програмного комплексу.

3 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ТА ОГЛЯД СТЕКУ ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

3.1Створення UML-діаграм

У процесі моделювання та проєктування програмного забезпечення часто використовують UML-діаграми. Вони є одними з найпоширеніших і найбільш зручних інструментів, оскільки дозволяють описувати бізнес-процеси, системне проєктування, а також різні організаційні структури [7, 8].

У рамках розробки вебзастосунка реально часового месенджера, було використано наступні UML-діаграми для моделювання різних аспектів системи:

– Діаграма варіантів використання (Use Case Diagram). Визначення функціональних вимог до застосунку, включаючи взаємодію користувача з функціями розробленої системи.

– Діаграма класів (Class Diagram). Моделювання структури класів та об'єктів застосунку, обробку запитів та взаємодію з базою даних.

– Діаграма послідовностей (Sequence Diagram). Відображення послідовності взаємодій між користувачем, застосунком та зовнішніми системами (наприклад, сервером) під час виконання ключових сценаріїв використання.

– Діаграма діяльності (Activity Diagram). Моделювання основних процесів, пов'язаних з роботою вебзастосунка, таких як створення та видалення групових чатів, надсилання повідомлення та узагальнена спрощена діаграма діяльності.

– Діаграма компонентів (Component Diagram). Відображення архітектури застосунку, включаючи компоненти, відповідальні за інтерфейс користувача, обробку даних та зв'язок з сервером.

– Діаграма розгортання (Deployment Diagram). Моделювання фізичного розгортання компонентів застосунку на пристроях та серверах, включаючи взаємодію з сервісами.

На етапі моделювання вебзастосунка для онлайн спілкування використано діаграму варіантів використання системи, діаграми діяльності та діаграму класів. Ці

діаграми чітко відображають структуру та вимоги до застосунку. На етапі проєктування застосувань станів та переходів, компонентів та розгортання, пакетів, які відображають поведінку, архітектуру ПЗ та його апаратну частину.

3.1.1 Діаграми взаємодії

Діаграми взаємодії використовуються для моделювання динамічних аспектів системи, показуючи, як об'єкти взаємодіють один з одним у певній послідовності дій.

Для вебзастосунка реально часового чат-месенджеру було розроблено три діаграми взаємодії.

Перша діаграма (рис. 3.1) ілюструє процес створення групового чату:

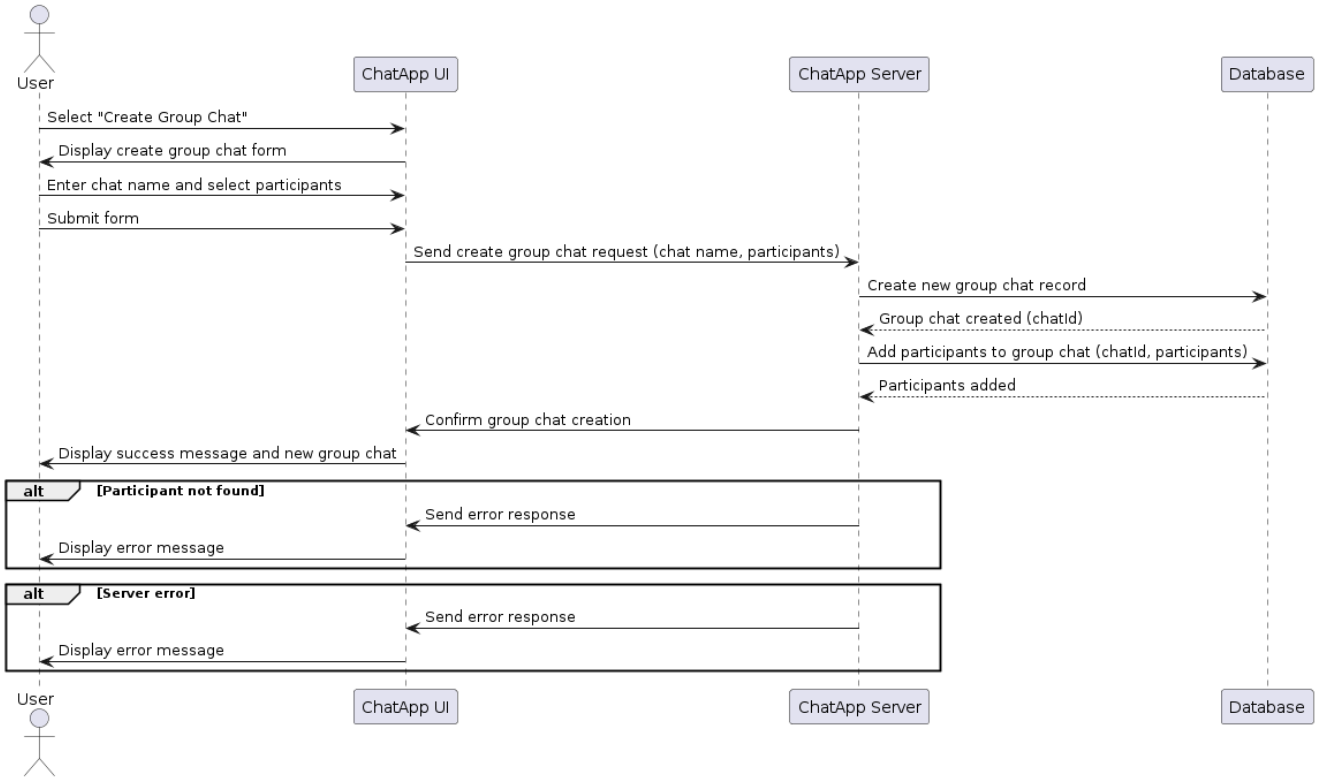

Рисунок 3.1 – Діаграма взаємодії до Use Case «Створення групового чату»

Послідовність дій виглядає наступним чином:

1) користувач вибирає опцію «Створити груповий чат»;

2) інтерфейс користувача (UI) відображає форму створення групового чату;

3) користувач вводить назву чату та додає учасників;

4) користувач надсилає форму;

5) UI надсилає запит на створення групового чату на сервер;

6) сервер створює новий запис групового чату у базі даних;

7) база даних підтверджує створення групового чату та повертає його ідентифікатор (chatId);

8) сервер додає учасників до групового чату у базі даних;

9) база даних підтверджує додавання учасників;

10) сервер підтверджує створення групового чату UI;

11) UI відображає повідомлення про успішне створення чату та новий груповий чат користувачу.'

Друга діаграма (рис. 3.2) описує процес видалення групового чату:

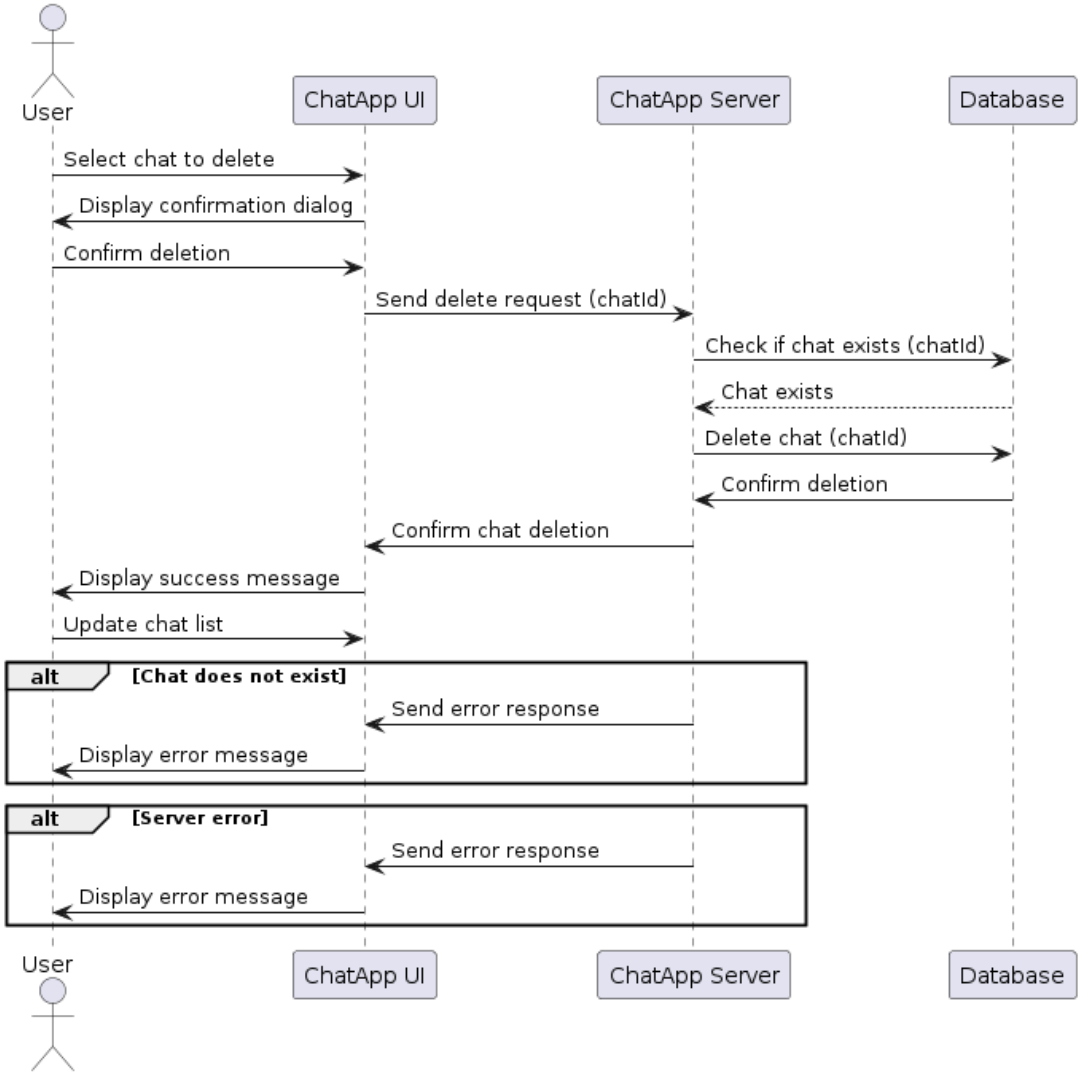

Рисунок 3.2 – Діаграма взаємодії до Use Case «Видалення чату»

Послідовність дій виглядає наступним чином:

1) користувач обирає чат для видалення;

2) інтерфейс користувача (UI) відображає діалог підтвердження;

3) користувач підтверджує видалення;

4) UI надсилає запит на видалення чату на сервер;

5) сервер перевіряє існування чату у базі даних;

6) якщо чат існує, сервер видаляє його з бази даних;

7) сервер підтверджує видалення чату UI;

8) UI відображає повідомлення про успішне видалення чату користувачу;

9) чат з усіма повідомленнями видаляється для всіх його учасників;

10) сиситема оновлює список чатів.

Третя діаграма (рис. 3.3) ілюструє процес редагування інформації про користувача:

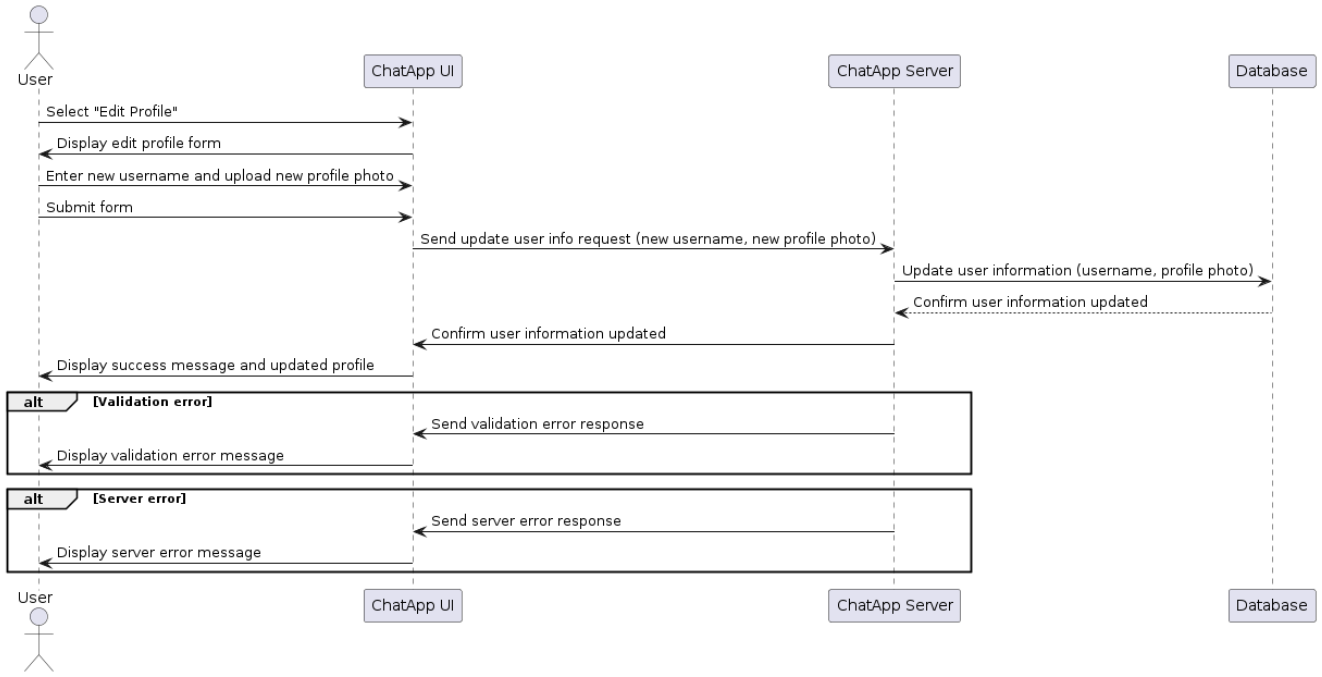

Рисунок 3.2 – Діаграма взаємодії до Use Case «Зміна інформації про користувача»

Послідовність дій виглядає наступним чином:

1) користувач вибирає опцію «Редагувати профіль»;

2) інтерфейс користувача (UI) відображає форму редагування профілю;

3) користувач вводить нову інформацію про профіль;

4) користувач надсилає форму;

5) UI надсилає запит на зміну інформації про користувача на сервер;

6) сервер оновлює інформацію про користувача у базі даних;

7) база даних підтверджує оновлення інформації про користувача;

8) сервер підтверджує оновлення інформації про користувача UI;

9) UI відображає повідомлення про успішне оновлення та оновлений профіль користувачу.

Альтернативні сценарії включають випадки, коли відбувається помилка валідації або помилка сервера.

3.1.2 Діаграма станів та переходів

Діаграми станів та переходів (Statechart Diagrams) є важливим інструментом для моделювання поведінки вебзастосунку чат-месенджеру. Вони дозволяють візуалізувати динаміку системи, показуючи, як застосунок реагує на різні події та дії користувача. рамках проєкту кваліфікаційної роботи бакалавра було розроблено 3 ключові діаграми станів та переходів:

1) загальна діаграма станів та переходів (рис. 3.4);

2) діаграма станів для usecase «Створення та видалення групового чату» (рис. 3.5);

3) діаграма станів для usecase «Управління профілем користувача» (рис. 3.6).

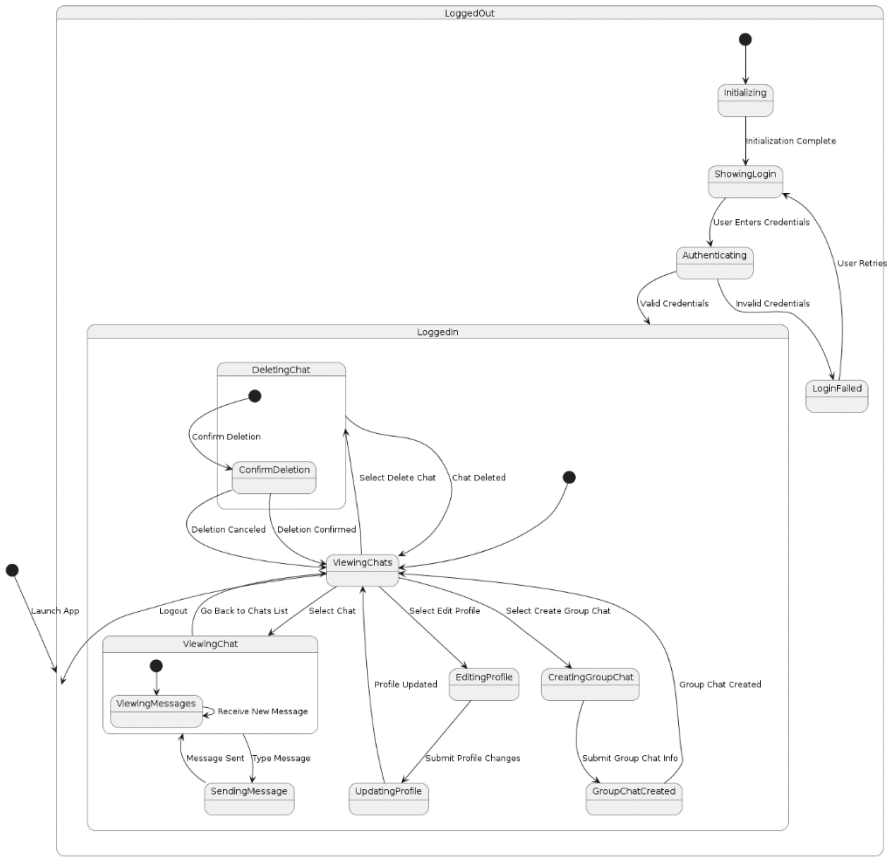

Рисунок 3.4 – Загальна діаграма станів

Ця діаграма відображає загальний життєвий цикл застосунку та можливі взаємодії користувача з ним. Вона демонструє наступні стани:

1. **LoggedOut**:

– Initializing: Початковий стан, коли застосунок запускається.

– ShowingLogin: Відображення форми входу (Реєстрація/Авторизація).

– Authenticating: Перевірка облікових даних користувача.

– LoginFailed: Невдалий вхід через неправильні облікові дані.

– LoggedIn: Успішний вхід, перехід до стану «LoggedIn».

2. **LoggedIn**:

– ViewingChats: Перегляд списку чатів.

– ViewingChat: Перегляд конкретного чату.

– SendingMessage: Відправлення повідомлення в чаті.

– EditingProfile: Редагування профілю користувача.

– UpdatingProfile: Оновлення профілю користувача.

– CreatingGroupChat: Створення групового чату.

– GroupChatCreated: Груповий чат створено.

– LoggedOut: Вихід з облікового запису.

3. **ViewingChat**:

– ViewingMessages: Перегляд повідомлень у чаті.

– ViewingMessages: Отримання нового повідомлення в чаті.

4. DeletingChat:

– ConfirmDeletion: Підтвердження видалення чату.

– ViewingChats: Повернення до списку чатів після підтвердження або скасування видалення.

Розглянемо детально діаграму станів для створення та видалення групового чату:

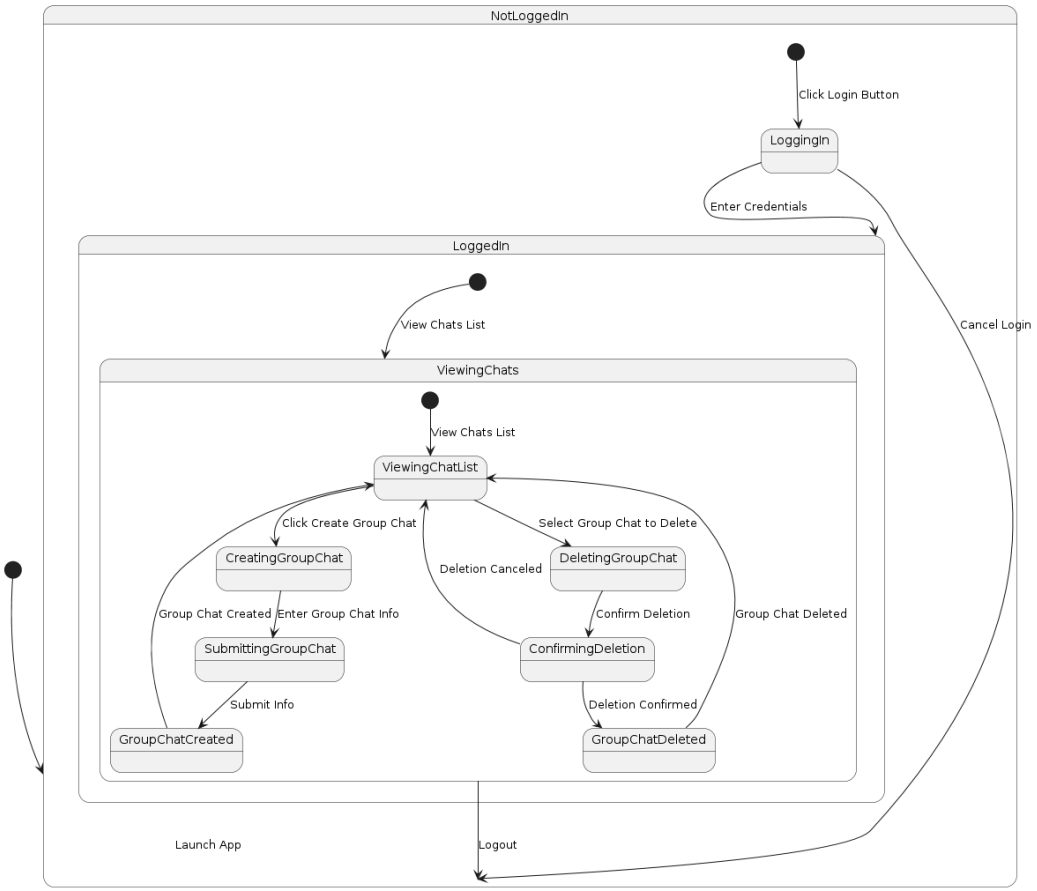

Рисунок 3.5 – Діаграма станів для usecase «Створення та видалення групового чату»

На діаграмі деталізовно процес створення та видалення групового чату починаючи з авторизації користувача:

ViewingChats: Загальний стан перегляду чатів.

– ViewingChatList: Перегляд списку чатів.

– CreatingGroupChat: Створення групового чату.

– SubmittingGroupChat: Введення інформації про груповий чат та подання форми.

– GroupChatCreated: Підтвердження створення групового чату.

– DeletingGroupChat: Видалення групового чату.

– ConfirmingDeletion: Підтвердження видалення групового чату.

– GroupChatDeleted: Підтвердження видалення групового чату.

Розглянемо детально діаграма станів для управління профілем користувача:

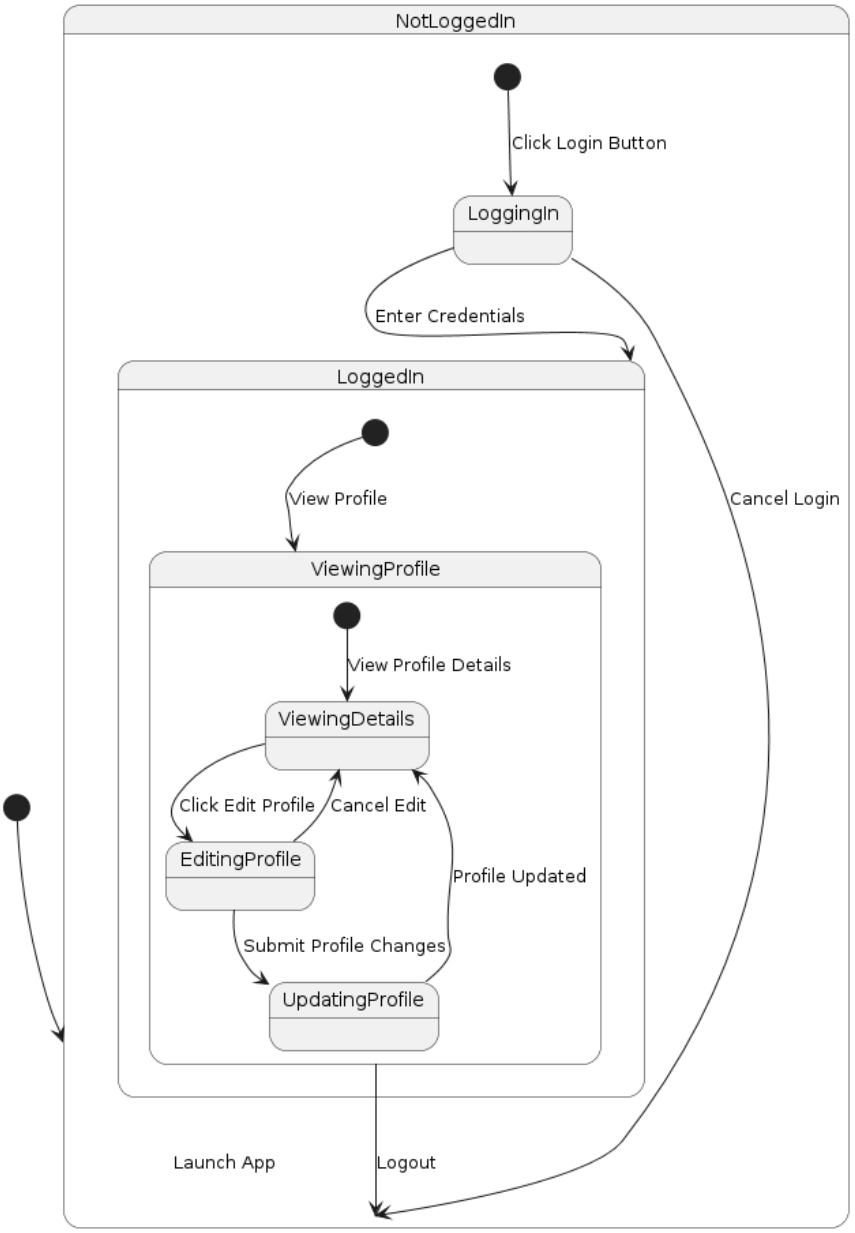

Рисунок 3.6 – Діаграма станів для usecase «Управління профілем користувача»

На діаграмі представлено:

ViewingProfile: Загальний стан перегляду профілю користувача.

– ViewingDetails: Перегляд деталей профілю.

– EditingProfile: Редагування профілю.

– UpdatingProfile: Оновлення профілю після внесення змін.

Ці діаграми станів та переходів служать важливим інструментом для розуміння та аналізу поведінки вебзастосунку, допомагаючи виявити потенційні проблеми та оптимізувати взаємодію з користувачем.

3.1.3 Діаграми компонентів та розгортання

Діаграми компонентів і розгортання використовуються для моделювання фізичних аспектів програмного забезпечення. Вони допомагають визначити структуру системи, взаємозв'язки між її частинами та відображають апаратні й програмні компоненти, необхідні для виконання застосунку [10].

Діаграма компонентів

На діаграмі компонентів (рис. 3.7) представлено структуру вебзастосунку чат меседжеру, діаграма відображає основні компоненти, що складають ChatApp, включаючи фронтенд і бекенд [22].

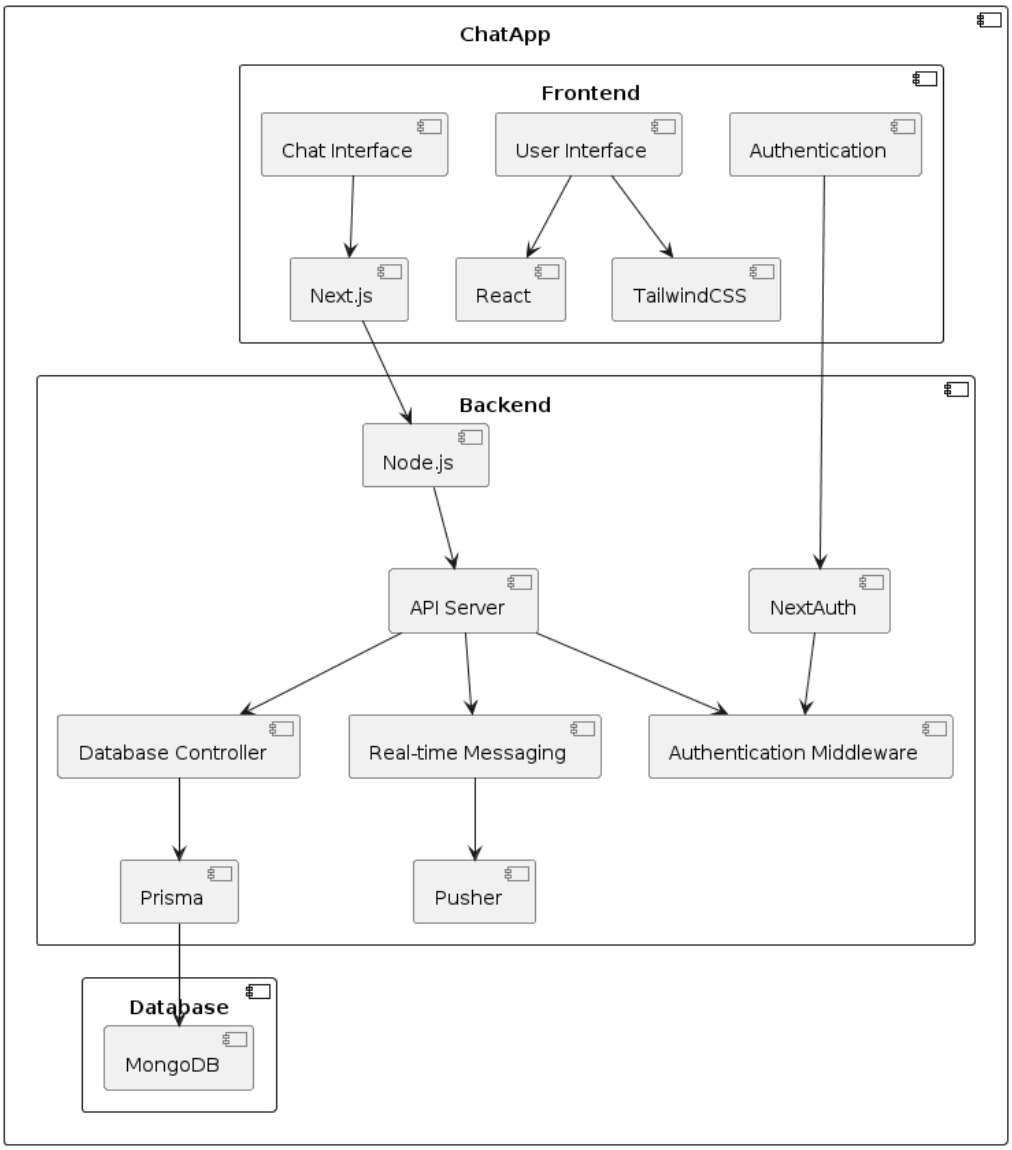

Рисунок 3.7 – Діаграма компонентів

Діаграма компонентів для системи ChatApp відображає взаємодію основних складових частин програми, які забезпечують її функціональність. Ось короткий опис компонентів:

1. Client (Клієнт):

– Browser (Браузер): Інтерфейс користувача для доступу до ChatApp через веббраузер.

2. Frontend (Фронтенд):

– Chat Interface: Компонент для взаємодії користувачів з чатами.

– User Interface: Загальний інтерфейс користувача для взаємодії з системою.

– Authentication: Компонент для реєстрації та входу користувачів.

– Next.js: Фреймворк для створення серверних рендерингів React-додатків.

– React: Бібліотека для створення інтерфейсів користувача.

– TailwindCSS: Утилітарний CSS-фреймворк для швидкого створення стилів.

3. Backend (Бекенд):

– Node.js: JavaScript середовище виконання, що дозволяє виконувати серверний код.

– API Server: Обробляє бізнес-логіку та запити до API.

– Authentication Middleware: Проміжне ПЗ для обробки аутентифікації користувачів.

– Database Controller: Контролер для взаємодії з базою даних.

– Real-time Messaging: Компонент для обміну повідомленнями в реальному часі.

– NextAuth: Система аутентифікації користувачів.

– Prisma: ORM для взаємодії з базою даних.

– Pusher: Служба для обміну повідомленнями в реальному часі через WebSocket-з'єднання.

4. Database (База даних):

– MongoDB: Нереляційна база даних для збереження всіх даних системи.

Діаграма розгортання

На діаграмі розгортання (рис. 3.8) представлено фізичну інфраструктуру вебзастосунку з використанням використантих технології.

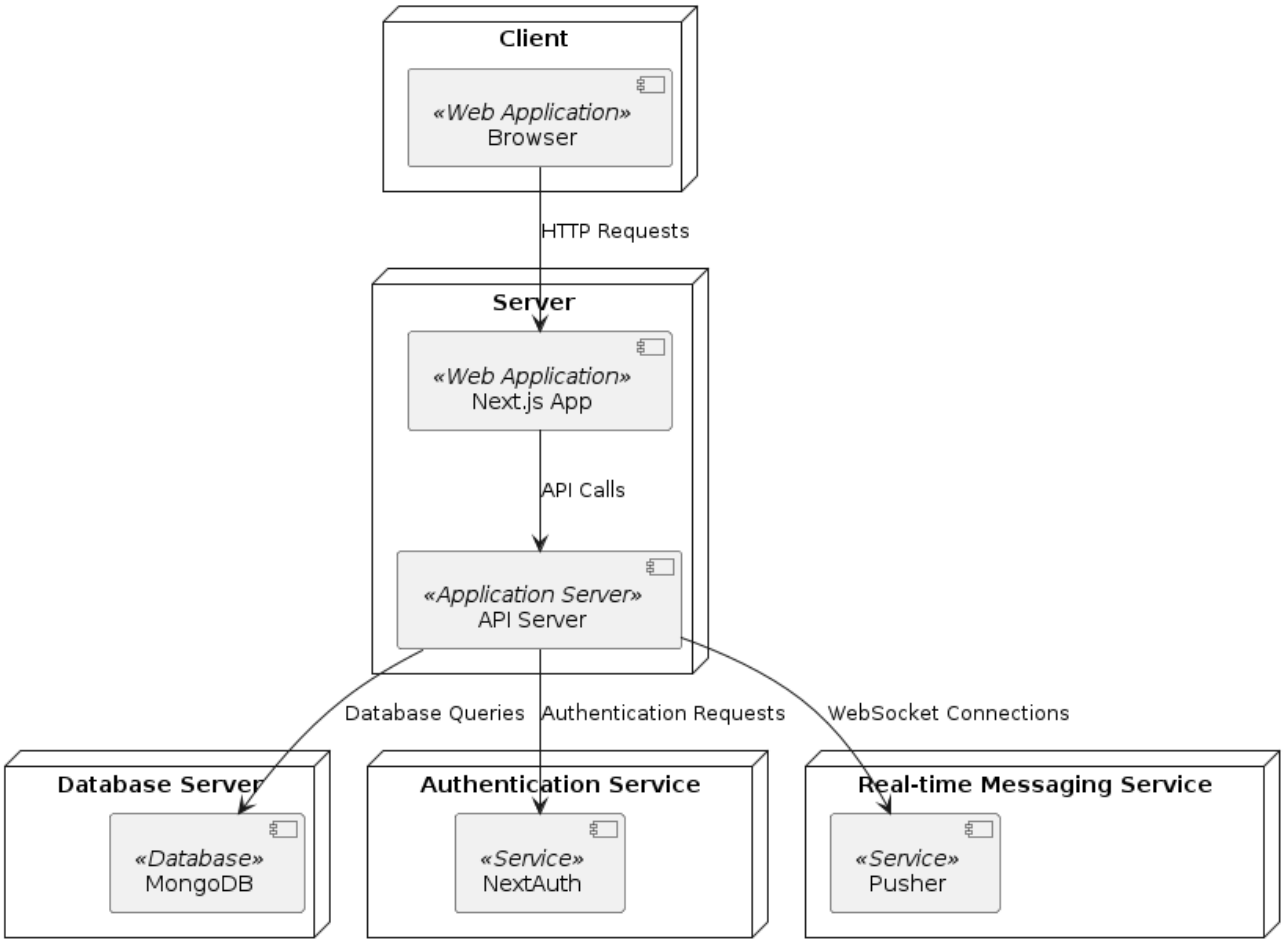

Рисунок 3.8 – Діаграма розгортання

Ця діаграма розгортання відображає архітектуру вебзастосунку реального часу для чат-месенджера ChatApp. Вона демонструє взаємодію між різними компонентами системи, включаючи клієнтську частину, серверну частину, базу даних, сервіс автентифікації та сервіс обміну повідомленнями в реальному часі [23].

Компоненти:

1. Client:

– Browser: клієнтська частина, через яку користувачі взаємодіють із застосунком. Відправляє HTTP-запити до серверної частини.

2. Server:

– Next.js App: вебзастосунок, що обробляє запити з браузера і викликає відповідні API [12].

– API Server: сервер застосунку, який обробляє бізнес-логіку і взаємодіє з базою даних, сервісом автентифікації та сервісом обміну повідомленнями.

3. Database Server:

– MongoDB: база даних, яка зберігає дані користувачів, повідомлень та інших сутностей.

4. Authentication Service:

– NextAuth: сервіс автентифікації, що обробляє запити на автентифікацію від API сервера.

5. Real-time Messaging Service:

– Pusher: сервіс обміну повідомленнями в реальному часі, що обробляє WebSocket-з'єднання для миттєвої доставки повідомлень.

3.1.4 Діаграма пакетів

Діаграма пакетів для системи ChatApp відображає логічну структуру основних модулів та їх взаємозв'язки (рис. 3.9). Вона поділена на три основні секції: Frontend, Backend та Database, а також містить інтеграцію з хмарними сервісами для завантаження зображень та соціальної аутентифікації [24].

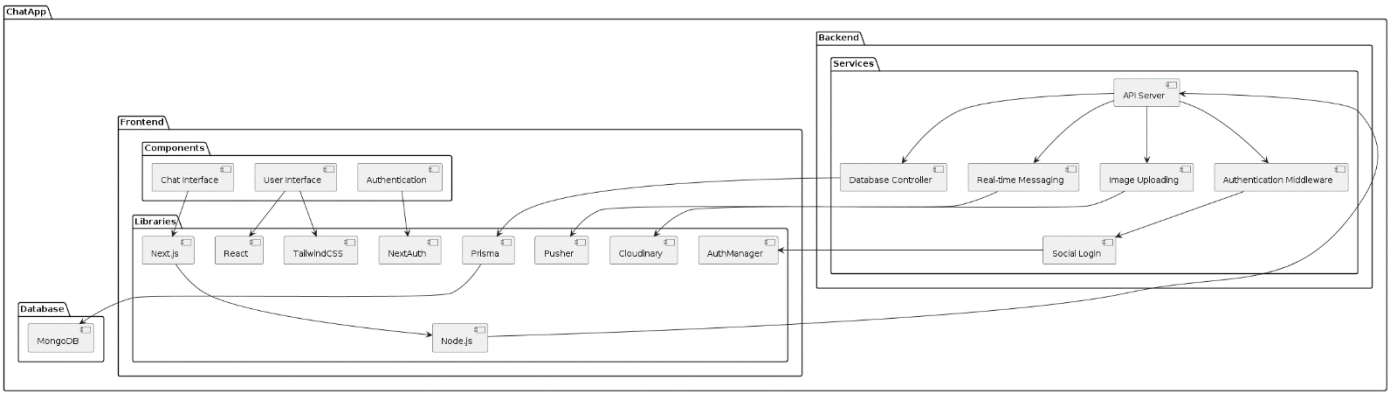

Рисунок 3.9 – Діаграма пакетів

Пакет Frontend

Frontend відповідає за інтерфейс користувача та взаємодію з користувачами. Включає такі пакети:

– Components: Вміщує окремі частини інтерфейсу, такі як Chat Interface (інтерфейс чату), User Interface (інтерфейс користувача) та Authentication (аутентифікація).

– Libraries: Містить зовнішні бібліотеки, що використовуються у фронтенді, включаючи Next.js (фреймворк для серверного рендерингу React-додатків), React (бібліотека для побудови інтерфейсів користувача) та TailwindCSS (CSS-фреймворк для швидкого створення стилів).

Пакет Backend

Backend відповідає за обробку бізнес-логіки та взаємодію з базою даних. Включає такі пакети:

– Services: Містить основні сервіси бекенду, включаючи API Server (сервер для обробки API-запитів), Authentication Middleware (проміжне ПЗ для аутентифікації), Database Controller (контролер бази даних) та Real-time Messaging (обмін повідомленнями в реальному часі).

– Libraries: Містить зовнішні бібліотеки, що використовуються у бекенді, включаючи NextAuth (система аутентифікації), Prisma (ORM для роботи з базою даних) та Pusher (служба для обміну повідомленнями в реальному часі).

Пакет Database

Database відповідає за зберігання даних системи. Включає наступний пакет:

– MongoDB: Нереляційна база даних для зберігання всіх даних системи.

Пакет Cloud Services

Cloud Services відповідає за інтеграцію з зовнішніми хмарними сервісами. Включає такі пакети:

– Cloudinary: Хмарний сервіс для завантаження та зберігання зображень.

– AuthManager: Система для соціальної аутентифікації користувачів.

Взаємодія пакетів

– Chat Interface, User Interface та Authentication у Frontend використовують Next.js, React та TailwindCSS.

– Next.js у Frontend взаємодіє з Node.js у Backend.

– Node.js взаємодіє з API Server у Backend.

– API Server у Backend взаємодіє з Authentication Middleware та Database Controller.

– Database Controller у Backend взаємодіє з Prisma, який підключається до MongoDB у Database.

– API Server також взаємодіє з Real-time Messaging, який використовує Pusher.

– API Server інтегрується з Cloudinary для завантаження зображень.

– NextAuth у Backend взаємодіє з Authentication Middleware для аутентифікації користувачів.

– Authentication Middleware інтегрується з AuthManager для соціальної аутентифікації.

Ця діаграма показує, як різні пакети взаємодіють один з одним, щоб забезпечити функціонування системи ChatApp.

3.2 Огляд стеку технологій

Під час розробки вебзастосунка ChatApp – месенджера в реальному часі було використано сучасний стек технологій, який забезпечує високу продуктивність, зручність у розробці, підтримку застосунку та безпеку. Огляд ключових технологій, що були використані [4]:

1. Html - основна мова для створення вебсторінок та вебзастосунків. Вона використовується для структурування та представлення вмісту в Інтернеті.

2. Next.js 14 – це реактивний фреймворк для створення вебзастосунків на основі React. Використання Next.js дозволяє покращити продуктивність та швидкість розробки завдяки можливості передового серверного рендерингу, оптимізації вебсторінок та простоті конфігурації [13].

3. React – це JavaScript-бібліотека для побудови інтерфейсів користувача. Використання React у поєднанні з Next.js дозволяє створювати ефективні та динамічні клієнтські інтерфейси [11].

4. Tailwind CSS – це сучасний CSS-фреймворк, який дозволяє швидко та зручно створювати стилізовані компоненти за допомогою класів [15].

5. Prisma – це ORM (Object-Relational Mapping), який спрощує взаємодію з базою даних MongoDB шляхом надання зручного API для створення, читання, оновлення та видалення даних [16].

6. MongoDB – це документ-орієнтована нереляційна база даних, яка використовується для зберігання структурованих та неструктурованих даних, таких як профілі користувачів, повідомлення тощо [17].

7. NextAuth – це бібліотека для автентифікації та авторизації користувачів у вебзастосунках на основі Next.js. Вона забезпечує різні методи автентифікації, такі як електронна пошта та пароль, соціальні мережі тощо.

8. Pusher – це хмарна платформа для реального часу, яка забезпечує миттєву передачу даних через WebSocket-з'єднання, що дозволяє реалізувати функціонал реального часу в чат-месенджері, такий як миттєве відображення повідомлень [18].

3.2.1 Мова програмування

Зазвичай при розробці Frontend частини вебзастосунків використовують мову програмування JavaScript. JavaScript є універсальною мовою програмування, яка дозволяє створювати динамічні та інтерактивні елементи на вебсторінках. За допомогою JavaScript можна використовувати різноманітні бібліотеки та фреймворки, такі як React, Vue.js, або Angular, що дозволяє розширити можливості вебзастосунка та полегшити розробку.

Рисунок 3.10 – Логотип JavaScript

Основною ж мовою програмування для розробки вебзастосунка Real-time ChatApp було обрано TypeScript. TypeScript є суворо типізованою мовою програмування, яка базується на JavaScript і дозволяє покращити якість і надійність коду завдяки використанню типів даних і статичному аналізу. TypeScript надає розширені функції, такі як інтерфейси, переліки, класи, модулі та інші, що дозволяє ефективно розробляти складні програми та застосунки та використовувати усі основні принципи ООП.

Рисунок 3.11 – Логотип TypeScript

Основні переваги використання TypeScript включають:

1. **Сувора типізація**: TypeScript дозволяє визначати типи даних для змінних, функцій та об'єктів, що покращує перевірку типів під час компіляції і запобігає потенційним помилкам.

2. **Підтримка сучасних функцій JavaScript**: TypeScript базується на стандарті ECMAScript і підтримує всі сучасні функції JavaScript, такі як стрілкові функції, розгалуження, асинхронний код тощо.

3. **Легий рефакторинг**: Завдяки суворій типізації, можна безпечно проводити рефакторинг коду, змінювати його структуру та функції, не хвилюючись про можливі проблеми з типами даних.

4. **Більш висока надійність**: TypeScript допомагає виявляти і виправляти помилки ще до виконання коду, що забезпечує більшу надійність і стабільність застосунку.

Таким чином, використання TypeScript у розробці застосунку чат месенджера дозволило покращити його якість, ефективність та надійність, забезпечуючи зручні інструменти для розробки та підтримки.

3.2.2 Допоміжні технології та бібліотеки

У процесі розробки Real-Time Chat App було використано ряд допоміжних технологій та бібліотек, які сприяли поліпшенню продуктивності, забезпеченню безпеки та стабільності застосунку, а також реалізації потрібного функціоналу. Нижче подано короткий огляд кожної з цих технологій та бібліотек [21].

1. **axios**: Бібліотека для виконання HTTP-запитів з JavaScript. axios надає зручний і простий інтерфейс для роботи з AJAX-запитами на клієнтській стороні.

2. **Prisma**: ORM-інструмент для Node.js та TypeScript, що дозволяє легко взаємодіяти з базами даних. Prisma автоматично генерує код для взаємодії з базою даних, полегшуючи розробку.

3. **bcrypt**: Бібліотека для хешування паролів у Node.js. Вона використовує сильний алгоритм хешування, що забезпечує безпеку паролів користувачів.

4. **tailwindcss**: CSS-фреймворк, який дозволяє швидко створювати стилізовані компоненти та макети за допомогою класів. Він дозволяє зосередитись на дизайні, не втрачаючи час на написання власного CSS.

5. **react-hook-form**: Бібліотека для React, що дозволяє легко керувати формами у застосунку. Вона надає зручні хуки для валідації, обробки подій та керування станом форм.

6. **npx**: Інструмент для виконання локальних пакетів Node.js. Він дозволяє запускати команди локально встановлених пакетів без встановлення їх глобально.

7. **react-hot-toast**: Бібліотека для створення та відображення сповіщень в React-застосунку. Має простий API та гнучкі налаштування.

8. **next-auth**: Бібліотека для автентифікації та авторизації в Next.js застосунках. Підтримує різні методи автентифікації, включаючи соціальні мережі та JWT.

9. **react-icons**: Колекція іконок для React, що містить великий набір різних іконок для використання в застосунках.

10. **next-superjson-plugin**: Плагін для Next.js, що допомагає оптимізувати передачу та серіалізацію даних між сервером і клієнтом.

11. **date-fns**: Бібліотека для роботи з датами у JavaScript, яка має різноманітні функції для операцій з датами та часом.

12. **next-cloudinary**: Бібліотека для інтеграції з Cloudinary у Next.js застосунках для роботи з медіафайлами.

13. **zustand**: Бібліотека для керування станом у React-застосунках, що надає простий API для створення та керування глобальним станом.

14. **headlessui/react**: Бібліотека з компонентами без UI для React, яка дозволяє створювати доступні інтерфейси без зайвого коду.

15. **react-select**: Бібліотека для створення випадаючих списків у React-застосунках.

16. **react-spinners: Колекція компонентів-завантажувачів для React.**

17. **pusher-js**: Клієнтська бібліотека для роботи з Pusher, яка дозволяє забезпечити реально-часову комунікацію між клієнтом і сервером [14].

18. **lodash**: Утилітарна бібліотека для роботи зі структурами даних та іншими операціями у JavaScript.

3.2.3 Технології зберігання та управління даними

Використання MongoDB

Однією з ключових технологій для зберігання даних у реально часовому чат-месенджері оборано MongoDB – нереляційна база даних, яка дозволяє зберігати як структуровані, так і неструктуровані дані, такі як профілі користувачів, повідомлення та інше. MongoDB відома своєю гнучкістю та швидкістю доступу до даних, що робить її ідеальним вибором для реалізації застосунків з великим обсягом даних та потребами у швидкій обробці запитів [19].

Рисунок 3.12 – Логотип MongoDB

Використання Prisma

Prisma використовувався для забезпечення Object-Relational Mapping (ORM) та спрощення взаємодії з базою даних MongoDB. Це дозволяє розробникам легко створювати, читати, оновлювати та видаляти дані, а також виконувати різноманітні запити до бази даних. Prisma допомагає знизити кількість ручного коду, необхідного для роботи з базою даних, і забезпечує зручний та ефективний спосіб керування даними.

Рисунок 3.13 – Логотип Prisma

Використання NextAuth

NextAuth використовувався для реалізації системи автентифікації та авторизації користувачів у вебзастосунках на основі Next.js. Це дозволяє

забезпечувати безпеку доступу до даних та функціональності застосунку, а також надає зручний і безпечний механізм управління правами доступу.

Рисунок 3.14 – Логотип NextAuth

Ці технології разом створюють потужний та ефективний стек технологій для розробки реально-часового чат-месенджера. Вони забезпечують високу продуктивність, швидкість реакції та безпеку даних користувачів, що є критичними аспектами для успішного функціонування вебзастосунка.

Висновки до розділу 3

У третьому розділі було детально розглянуто та проаналізовано основні аспекти розробки вебзастосунка реально часового чат месенджера. Зокрема, були створер різні типи UML-діаграм, які відобразили архітектуру застосунку, його компоненти та взаємодії між ними. Використання діаграм класів, компонентів, станів та переходів допомогло чітко визначити структуру та логіку роботи системи.

Проаналізовано основні технології, використані для розробки застосунку. Зокрема, обрану мова програмування JavaScript, а точніше, його строго типізовану версію TypeScript, яка гарантує високий рівень безпеки та зрозумілість коду. Були детально розглянуті технології front-end та full-stack розробки, включаючи використані бібліотеки та плагіни, і були визначені технології зберігання та управління даними.

Описано використання обраних інструментів для реалізації необхідного функціоналу. Результатом цього аналізу є створення ефективного та функціонального чат-месенджера, який відповідає вимогам сучасного вебзастосунку та забезпечує зручність та безпеку для його користувачів.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ТА ПРОВЕДЕННЯ ТЕСТУВАННЯ

Вебзастосунок містить кілька основних екранів, зокрема: екран авторизації/реєстрації, головний, екран користувачів, екран налаштувань та самого чату. Основним користувачем системи є клієнт, який має доступ до функціональних можливостей застосунку.

4.1 Реалізація дизайну вебзастосунку

Екран авторизації/реєстрації

Екран реєстрації (рис. 4.1) є одним з ключових екранів вебзастосунку чат-месенджеру. Він дозволяє користувачам зареєструватися, або скористатися стороннми сервісами, такими як Google Account або Github для входу.

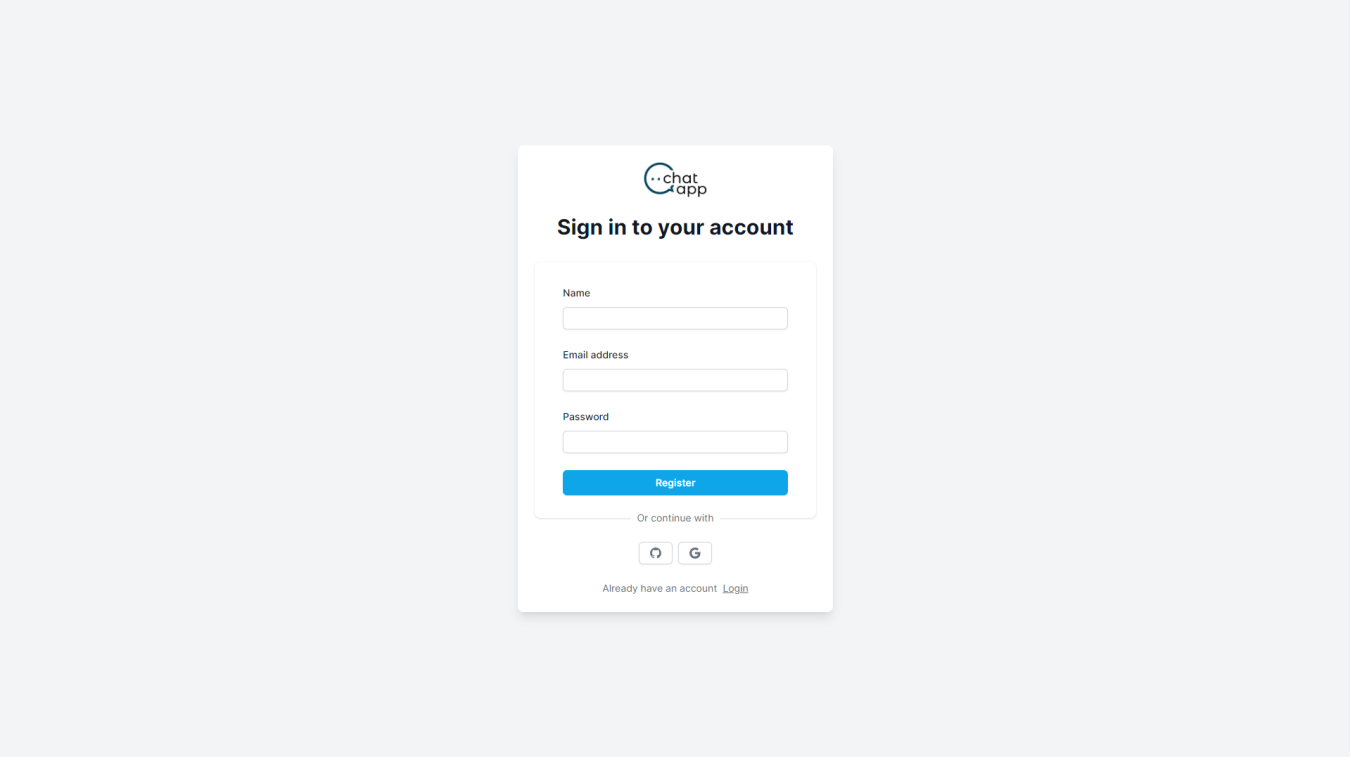

Рисунок 4.1 – Екран реєстрації

Екран авторизації дозволяє користувачеві увійти до системи, використовуючи свій обліковий запис. Крім того, оскільки це перше, що бачить користувач при вході до вебзастосунку, він має унікальний дизайн із логотипом для кращого запам'ятовування. Дизайн цього як і попереднього екрану був розроблений з

урахуванням принципів зручності та мінімалізму, що забезпечує простоту використання і швидкий доступ до функціоналу застосунку.

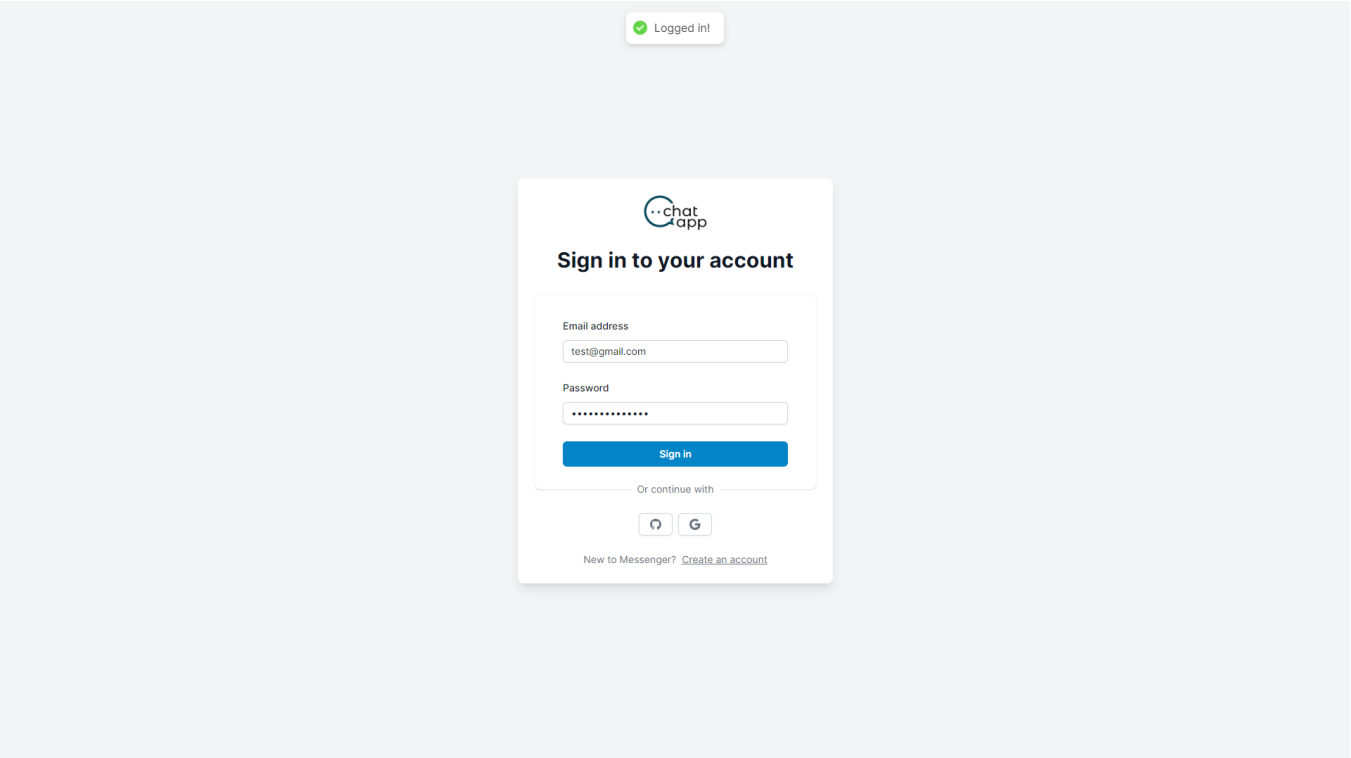

Рисунок 4.2 – Екран успішної авторизації

Екран користувачів

Екран користувачів вебзастосунка (рис. 4.3) дозволяє користувачеві переглядати список доступних користувачів та вибирати чати для спілкування. Він складається з трьох основних частин:

1. Панель навігації зліва:

– включає іконки для переходу між різними розділами застосунку;

– забезпечує легкий доступ до основних функцій, таких як список користувачів, список чатів та вихід із системи.

2. Список користувачів:

– відображається список користувачів із їхніми аватарами та іменами;

– дозволяє бачити статус користувачів (зелений індикатор свідчить про онлайн-статус).

3. Основна область:

– показує повідомлення з підказкою: «Select a chat or start a new conversation»,

яке нагадує користувачеві вибрати чат або розпочати нову розмову.

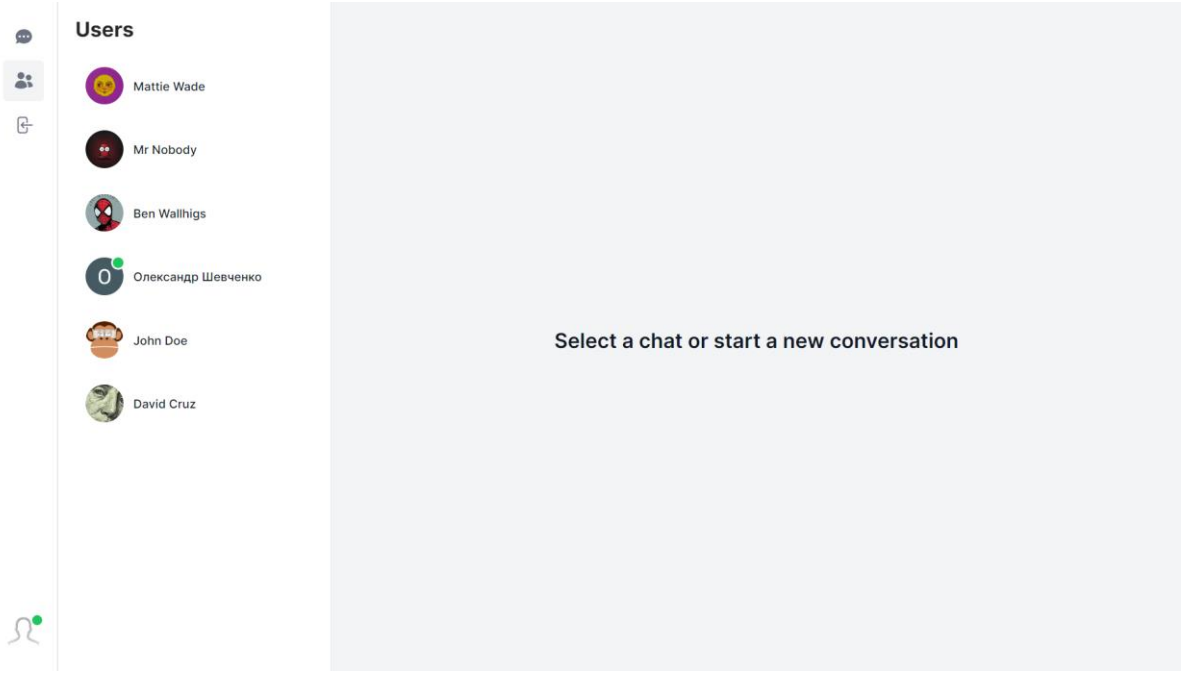

Рисунок 4.3 – Екран користувачів

Дизайн цього екрану, як і екрану авторизації, був розроблений з урахуванням принципів зручності та мінімалізму, що забезпечує простоту використання та швидкий доступ до функціонала застосунку.

Екран діалогу

Сторінка діалогу вебзастосунку (рис. 4.4) забезпечує зручний інтерфейс для спілкування між користувачами. Вона складається з кількох основних елементів:

1. Панель навігації зліва:

– містить іконки для переходу між різними розділами застосунку, такими як список користувачів, кнопку створення групових чатів та вихід із системи, що забезпечує користувачеві легкий доступ до основних функцій застосунку.

2. Список чатів:

– відображається перелік активних чатів з іменами та аватарами користувачів;

– показуються останні повідомлення та статуси чатів, що допомагає швидко орієнтуватися в нових і поточних розмовах.

3. Основна область діалогу:

– включає область для перегляду та відправлення повідомлень у вибраному чаті;

– відображає повідомлення з аватаром користувача, ім'ям та часом відправлення;

– повідомлення представлені в хронологічному порядку, що полегшує спілкування та перегляд історії розмови.

4. Поле вводу повідомлень:

– розташоване в нижній частині основної області діалогу;

– містить текстове поле для вводу повідомлень та кнопку для їх відправлення;

– підтримує відправлення тексту, емодзі та зображень, що робить спілкування більш інтерактивним.

5. Інформаційний рядок чату:

– показує активного користувача та його статус;

– включає інформацію про поточного користувача, з яким ведеться діалог, для кращої орієнтації.

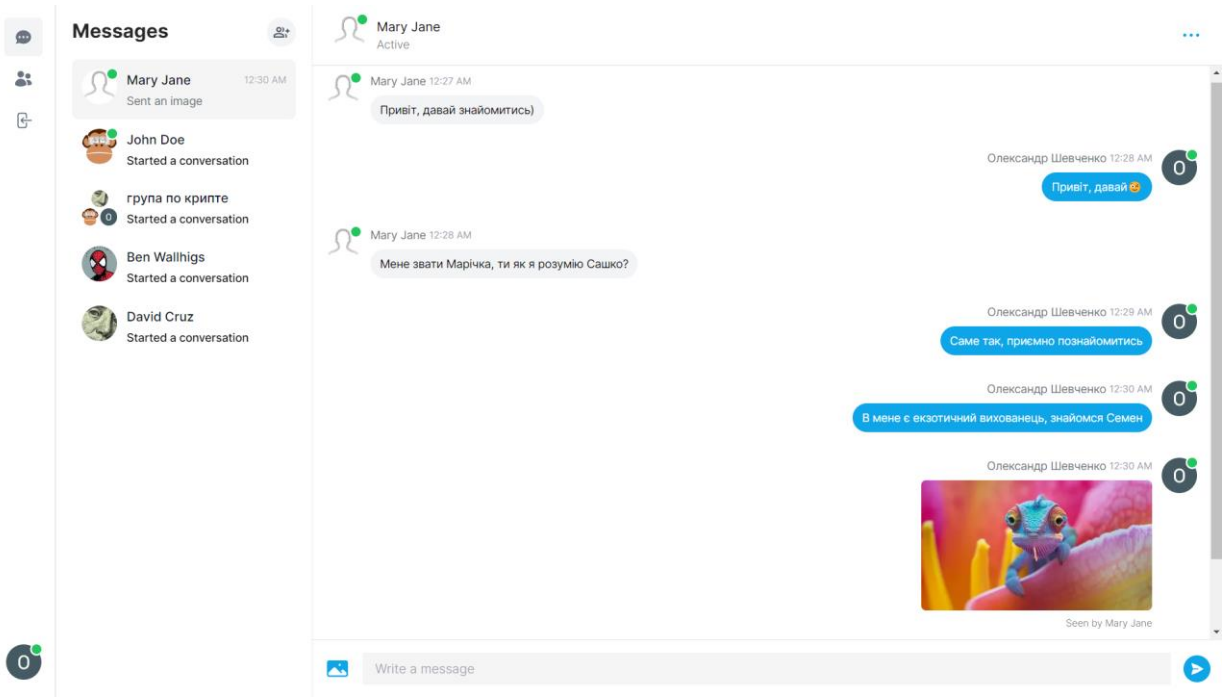

Рисунок 4.4 – Екран чату

Дизайн сторінки діалогу розроблений з урахуванням принципів зручності та мінімалізму, що забезпечує простоту використання та швидкий доступ до функціоналу застосунку. Він дозволяє користувачам легко спілкуватися в реальному часі, обмінюватися повідомленнями та медіа-файлами, що робить досвід

користування приємним і ефективним.

Екран редагування даних користувача

Екран редагування даних користувача (рис 4.5) забезпечує простий і зрозумілий інтерфейс для оновлення особистої інформації. Дизайн включає такі основні елементи:

1. Діалогове вікно редагування профілю:

2. Поле вводу імені:

3. Розділ для зміни фото:

4. Кнопки управління:

5. Дизайн та розміщення:

– Центральне розташування діалогового вікна.

– Простий, мінімалістичний дизайн.

– Значок закриття у правому верхньому куті.

Цей інтерфейс забезпечує зручність і простоту оновлення особистої інформації користувача

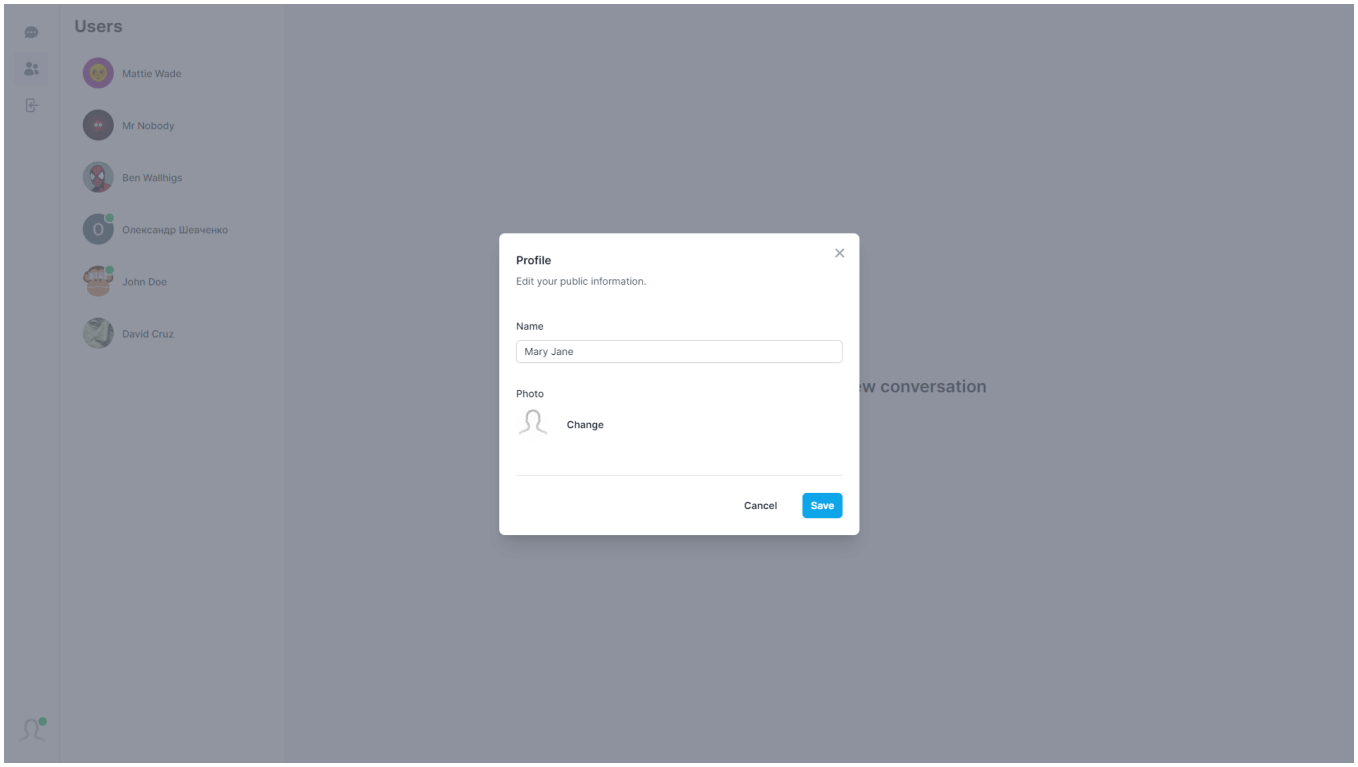

Рисунок 4.5 – Екран редагування даних користувача

Адаптивність вебзастосунка

Представлений скриншот (рис 4.6) демонструє екран користувачів у вебзастосунку, який було оптимізовано для мобільних пристроїв. Завдяки використанню Tailwind CSS, дизайн є повністю адаптивним, що забезпечує зручний і привабливий інтерфейс незалежно від розміру екрана.

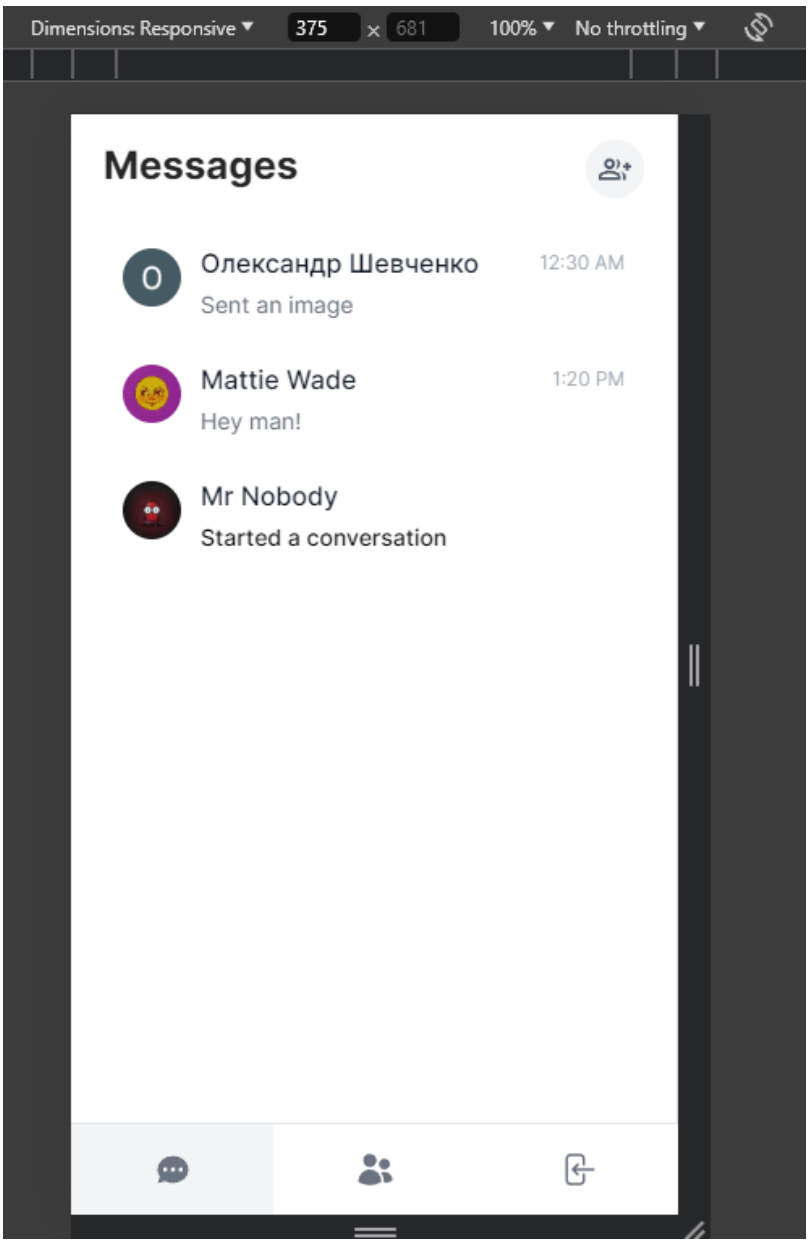

Рисунок 4.6 – Екран повідомлень на мобільних пристроях

Для забезпечення адаптивності було розташовано навігаційну панель в нижній частині екрана з трьома іконками для швидкого доступу до різних розділів застосунку:

Використання Tailwind CSS дозволило досягти високого рівня адаптивності:

– Responsive Design: Tailwind CSS надає можливість легко налаштовувати стилі для різних розмірів екранів за допомогою спеціальних утиліт, що дозволяє компонувати інтерфейс під мобільні пристрої, планшети та десктоп.

– Flexbox і Grid: Завдяки цим технологіям, компоненти інтерфейсу можуть гнучко змінювати свої розміри та позиціювання в залежності від розміру екрана.

– Простота налаштувань: Tailwind CSS дозволяє швидко та зручно налаштовувати стилі без необхідності писати великий обсяг CSS коду, що особливо корисно для адаптації інтерфейсу під різні пристрої.

4.2 Реалізація програмних компонентів та логіки чат-месенджеру

4.2.1 Реєстрація, авторизація та адаптивний інтерфейс

Реєстрація та авторизація є критично важливими компонентами для будь-якого чат-месенджеру, оскільки вони забезпечують безпеку та управління доступом до функціоналу системи. Авторизовані користувачі отримують доступ до більшого спектру можливостей, тоді як неавторизовані можуть лише переглядати обмежений вміст.

Для реалізації цих функцій у чат-месенджері були використані різні інструменти та бібліотеки, такі як NextAuth.js для управління аутентифікацією, react-hook-form для обробки форм, та axios для здійснення HTTP-запитів до сервера (рис. 4.7 – 4.8).

AuthForm.tsx

```tsx
const AuthForm = () => {
  const session = useSession();
  const router = useRouter();
  const [variant, setVariant] = useState<Variant>("LOGIN");
  const [isLoading, setIsLoading] = useState(false);

  useEffect(() => {
    if (session?.status === "authenticated") {
      router.push("/conversations");
    }
  }, [session?.status, router]);

  const toggleVariant = useCallback(() => {
    if (variant === "LOGIN") {
      setVariant("REGISTER");
    } else {
      setVariant("LOGIN");
    }
  }, [variant]);

  const {
    register,
    handleSubmit,
    formState: { errors },
  } = useForm<FieldValues>({
    defaultValues: {
      name: "",
      email: "",
      password: "",
    },
  });
```

Рисунок 4.7 – Код файлу AuthForm.tsx

Процес реєстрації та авторизації включає наступні етапи:

1. Форма реєстрації та авторизації:

– Користувач має можливість обрати між реєстрацією нового облікового запису або входом в існуючий.

– Форма для реєстрації включає поля для введення імені, електронної пошти та пароля.

– Форма для авторизації включає поля для введення електронної пошти та пароля.

2. Обробка даних форми:

– Використовується бібліотека react-hook-form для управління станом форми та валідації введених даних.

– При поданні форми відбувається перевірка введених даних, після чого дані надсилаються на сервер за допомогою axios.

3. Реєстрація нового користувача:

– Після отримання даних від форми реєстрації, вони передаються на сервер для створення нового облікового запису.

– Якщо реєстрація успішна, користувач отримує повідомлення про успішну реєстрацію та автоматично авторизується в системі.

4. Авторизація існуючого користувача:

– При поданні форми авторизації, введені дані перевіряються на сервері.

– Якщо дані введені правильно, користувач отримує доступ до свого облікового запису та відповідних функцій чат-месенджеру.

```
53  const onSubmit: SubmitHandler<FieldValues> = (data) => {
54    setIsLoading(true);
55
56    if (variant === "REGISTER") {
57      axios
58        .post("/api/register", data)
59        .then(() => {
60          toast.success("Registration successful!");
61          signIn("credentials", data);
62        })
63        .catch(() => toast.error("Something went wrong!"))
64        .finally(() => setIsLoading(false));
65    }
66
67    if (variant === "LOGIN") {
68      signIn("credentials", {
69        ...data,
70        redirect: false,
71      })
72        .then((callback) => {
73          if (callback?.error) {
74            toast.error("Invalid credentials");
75          }
76
77          if (callback?.ok && !callback?.error) {
78            toast.success("Logged in!");
79            router.push("/users");
80          }
81        })
82        .catch(() => toast.error("Login failed!"))
83        .finally(() => setIsLoading(false));
84    }
85  };
```

Рисунок 4.8 – Код файлу AuthForm.tsx (1)

Для управління аутентифікацією в чат-месенджері використовується NextAuth.js (рис. 4.9), який забезпечує:

– Підключення до бази даних: Використовується адаптер PrismaAdapter для взаємодії з базою даних через Prisma.

– Підтримка різних провайдерів: Налаштовані провайдери для GitHub, Google та власних облікових даних (електронна пошта та пароль).

– Безпечне зберігання сесій: Використовується стратегія зберігання сесій на основі JSON Web Token (JWT), що забезпечує безпеку та зручність управління сесіями.

api\auth\[...nextauth]\route.ts

```
import { PrismaAdapter } from "@next-auth/prisma-adapter";
import prisma from "@/app/libs/prismadb";

const getAuthOptions = (): AuthOptions => ({
  adapter: PrismaAdapter(prisma),
  providers: [
    GithubProvider({
      clientId: process.env.GITHUB_ID as string,
      clientSecret: process.env.GITHUB_SECRET as string,
    }),
    GoogleProvider({
      clientId: process.env.GOOGLE_CLIENT_ID as string,
      clientSecret: process.env.GOOGLE_CLIENT_SECRET as string,
    }),
    CredentialsProvider({
      name: "credentials",
      credentials: {
        email: { label: "email", type: "text" },
        password: { label: "password", type: "password" },
      },
      async authorize(credentials) {
        if (!credentials?.email || !credentials?.password) {
          throw new Error("Please provide both email and password.");
        }
        const user = await prisma.user.findUnique({
          where: {
            email: credentials.email,
          },
        });
        if (!user || !user?.hashedPassword) {
          throw new Error("User not found or invalid email.");
        }
        const isCorrectPassword = await bcrypt.compare(
          credentials.password,
          user.hashedPassword
        );
        if (!isCorrectPassword) {
          throw new Error("Incorrect password.");
        }
        return user;
      },
    }),
  ],
  debug: process.env.NODE_ENV === "development",
  session: {
    strategy: "jwt",
  },
  secret: process.env.NEXTAUTH_SECRET,
});
const handler = NextAuth(getAuthOptions());
export { handler as GET, handler as POST };
```

Рисунок 4.9 – Код файлу Auth route.ts

Усі ці компоненти разом забезпечують надійний, безпечний та зручний для користувача процес реєстрації та авторизації в чат-месенджері, що є важливою складовою успішного функціонування системи.

4.2.2 Реалізація реальної часової взаємодії

Для забезпечення реальної часової взаємодії в чат-месенджері використовується бібліотека Pusher, яка дозволяє впровадити вебсокети для миттєвого обміну повідомленнями між користувачами. Це дозволяє повідомленням відображатися у реальному часі без необхідності оновлення сторінки [3].

Реалізація реальної часової взаємодії включає наступні ключові компоненти:

1. Компонент для відображення повідомлень:

– Використовується React-компонент Body (рис 4.10), який відповідає за відображення списку повідомлень у чаті.

– У компоненті використовується useState для збереження стану повідомлень та useRef для автоматичного прокручування до останнього повідомлення.

2. Підписка на канал Pusher:

– Використовується хук useEffect для підписки на канал Pusher, що відповідає за конкретну розмову.

– При надходженні нового повідомлення або оновлення існуючого, викликаються відповідні обробники, які оновлюють стан компонента.

3. Обробка нових повідомлень:

– Обробник messageHandler додає нові повідомлення до стану та прокручує чат до останнього повідомлення.

– Обробник updateMessageHandler оновлює повідомлення у стані при отриманні оновлень.

4. Компонент для надсилання повідомлень:

– Після надсилання повідомлення, воно автоматично відображається у чаті у всіх користувачів, які підписані на відповідний канал Pusher.

Body.tsx

```tsx
const Body: React.FC<BodyProps> = ({ initialMessages = [] }) => {
  const bottomRef = useRef<HTMLDivElement>(null);
  const [messages, setMessages] = useState(initialMessages);

  const { conversationId } = useConversation();

  useEffect(() => {
    axios.post(`/api/conversations/${conversationId}/seen`);
  }, [conversationId]);

  useEffect(() => {
    pusherClient.subscribe(conversationId);
    bottomRef?.current?.scrollIntoView();

    const messageHandler = (message: FullMessageType) => {
      axios.post(`/api/conversations/${conversationId}/seen`);

      setMessages((current) => {
        if (find(current, { id: message.id })) {
          return current;
        }

        return [...current, message];
      });

      bottomRef?.current?.scrollIntoView();
    };

    const updateMessageHandler = (newMessage: FullMessageType) => {
      setMessages((current) =>
        current.map((currentMessage) => {
          if (currentMessage.id === newMessage.id) {
            return newMessage;
          }

          return currentMessage;
        })
      );
    };

    pusherClient.bind("messages:new", messageHandler);
    pusherClient.bind("message:update", updateMessageHandler);

    return () => {
      pusherClient.unsubscribe(conversationId);
      pusherClient.unbind("messages:new", messageHandler);
      pusherClient.unbind("message:update", updateMessageHandler);
    };
  }, [conversationId]);

```

Рисунок 4.10 – Код файлу Body.tsx

Лістинг коду для серверної сторони взаємодії (рис 4.11):

api\conversations\route.ts

```
export async function POST(request: Request) {
  try {
    const currentUser = await getCurrentUser();
    const body = await request.json();
    const { userId, isGroup, members, name } = body;
    if (!currentUser?.id || !currentUser?.email) {return new NextResponse("Unauthorized", { status: 400 });}
    if (isGroup && (!members || members.length < 2 || !name)) {return new NextResponse("Invalid data", { status: 400 });}
    if (isGroup) {
      const newConversation = await prisma.conversation.create({
        data: {
          name, isGroup, users: {
            connect: [
              ...members.map((member: { value: string }) => ({id: member.value,})),
              {id: currentUser.id,},
            ],},},
        include: { users: true,},
      });
      newConversation.users.forEach((user) => {
        if (user.email) {pusherServer.trigger(user.email, "conversation:new", newConversation);}
      });
      return NextResponse.json(newConversation);
    }
    const existingConversations = await prisma.conversation.findMany({
      where: {
        OR: [
          {userIds: {equals: [currentUser.id, userId],},},
          {userIds: {equals: [userId, currentUser.id],},},
        ],},});
    const singleConversation = existingConversations[0];
    if (singleConversation) {
      return NextResponse.json(singleConversation);
    }
    const newConversation = await prisma.conversation.create({
      data: {
        users: {
          connect: [
            {id: currentUser.id,},
            {id: userId,},
            ],},},
      include: {
        users: true,
      },
    });
    newConversation.users.map((user) => {
      if (user.email) {pusherServer.trigger(user.email, "conversation:new", newConversation);}
    });
    return NextResponse.json(newConversation);
  } catch (error) {
    return new NextResponse("Internal Error", { status: 500 });
  }
}
```

Рисунок 4.11 – Код файлу conversation route.ts

Завдяки використанню бібліотеки Pusher для впровадження вебсокетів, чат-месенджер забезпечує миттєве відображення повідомлень у реальному часі [20]. Це дозволяє користувачам спілкуватися без затримок, що є ключовим для сучасних чат-застосунках. Реалізація включає як клієнтську, так і серверну логіку, що спільно забезпечує безшовну інтеграцію та оновлення повідомлень у реальному часі.

4.3 Тестування вебзастосунка

Для отримання якісного та зручного вебзастосунку перед виданням його необхідно протестувати. При наявності проблем або багів необхідно виправити їх до етапу надання доступу користувачам. Дані результатів тестування містять успішні сценарії виконання та розширення, для опису ситуацій з виведенням попереджень про помилку системою, які наведено в таблицях 4.1- 4.4.

Таблиця 4.1 – Реєстрація у вебзастосунку

<table>
<tr><td>Діючі особи</td><td colspan="2">Гість, система</td></tr>
<tr><td>Мета</td><td colspan="2">Створити обліковий запис.</td></tr>
<tr><td>Передумова</td><td colspan="2">Користувач не авторизований в системі.</td></tr>
<tr><td colspan="3">Успішний сценарій:
1. Користувач переходить до форми реєстрації.
2. Користувач заповнює необхідні дані (ім’я, e-mail, пароль).
3. Користувач натискає на кнопку зареєструватись та надсилає дані системі.
4. Система оброблює обліковий запис повідомляє про підтвердження реєстрації.
5. Система зберігає обліковий запис.
6. Система автоматично переводить користувача до на головну сторінку.</td></tr>
<tr><td colspan="3">Сценарій успішний. Збережено новий обліковий запис.</td></tr>
<tr><td colspan="3">Розширення</td></tr>
<tr><td>1а</td><td colspan="2">Не всі поля заповнені. Система виводить відповідне повідомлення під кожним незаповненим полем. Результат: користувач не зареєстрований.</td></tr>
<tr><td>2а</td><td colspan="2">Користувач вводить вже існуючі дані. Система виводить відповідне повідомлення. Результат: користувач не зареєстрований.</td></tr>
<tr><td colspan="3">Усі сценарії розширення успішно виконані.</td></tr>
</table>

Таблиця 4.2 – Авторизація в системі

Діючі особи	Гість, система
Мета	Виконати авторизацію.
Передумова	Користувач не авторизований в системі.
Успішний сценарій: 1. Користувач переходить на сторінку авторизації. 2. Користувач заповнює необхідні дані (пошту, пароль). 3. Система валідує дані у БД. 4. Система переводить користувача на головну сторінку.	
Сценарій успішний. Авторизацію виконано	
Розширення	
1а	Не всі поля заповнені. Система виводить відповідне повідомлення під кожним незаповненим полем. Результат: користувач не авторизований.
2а	Не існує такого користувача або невірний пароль. Система виводить відповідне повідомлення. Результат: користувач не авторизований.
Усі сценарії розширення успішно виконані.	

Таблиця 4.3 – Реальна часова взаємодія у чат-месенджері

Діючі особи	Користувач, система
Мета	Забезпечити реальну часову взаємодію між користувачами.
Передумова	Користувач авторизований в системі та знаходиться в активній розмові.
Успішний сценарій: 1. Користувач надсилає повідомлення у розмові. 2. Система відправляє повідомлення на сервер. 3. Сервер оброблює повідомлення і надсилає його іншим учасникам розмови. 4. Учасники розмови отримують повідомлення в реальному часі завдяки підключенню до системи реального часу 5. Інтерфейс користувача оновлюється автоматично без необхідності перезавантаження сторінки. 6. Система відмічає повідомлення як «прочитане» для всіх учасників розмови, які переглянули його. 7. Користувач бачить підтвердження доставки та прочитання повідомлення іншими учасниками.	

Кінець таблиці 4.3

Сценарій успішний. Повідомлення надіслано та отримано в реальному часі.	
Розширення	
1а	Сервер недоступний. Система виводить відповідне повідомлення про помилку і намагається повторити відправку. Результат: Повідомлення не надіслано.
2а	Підключення до інтернету нестабільне. Система показує індикатор спроби повторного підключення та повторної відправки повідомлення. Результат: Повідомлення надіслано після відновлення з'єднання.
Усі сценарії розширення успішно виконані.	

Таблиця 4.4 – Створення групового чату

Діючі особи	Користувач, система
Мета	Створити груповий чат для обміну повідомленнями між учасниками.
Передумова	Користувач авторизований в системі
Успішний сценарій: 1. Користувач натискає на опцію створення нового чату. 2. Система відображає форму створення чату, де користувач може вказати назву чату та додати учасників. 3. Користувач вводить назву чату та обирає учасників зі списку контактів. 4. Після вибору учасників користувач натискає кнопку «Створити». 5. Система перевіряє, чи вказані учасники є користувачами системи та чи не вже є у них груповий чат з такою ж назвою. 6. Якщо усі умови виконані і кількість учасників більше або дорівнює 2, система створює новий груповий чат та додає усіх учасників до нього. 7. Усі учасники отримують сповіщення про створення чату та автоматично додаються до нього. 8. Система відображає повідомлення про успішне створення чату.	
Сценарій успішний. Створено груповий з обраною назвою та додано в нього обраних учасників	
Розширення	
1а	Користувач обрав менше ніж 2 учасників або не вказав назву чату. Результат: система виводить відповідне повідомлення про помилку і блокує створення чату.
Усі сценарії розширення успішно виконані.	

Проведено тестування чотирьох основних сценаріїв використання вебзастосунка: реєстрація, авторизація, створення групового чату та реально-часової взаємодії у чат-месенджері. Усі основні та розширені сценарії успішно виконані, що підтвердило правильність роботи функціонала застосунку.

Висновки до розділу 4

У четвертому розділі розглянуто програмну реалізацію та тестування вебзастосунка чат-месенджеру. Основні аспекти, які були висвітлені, включають:

Розглянуто основні екрани, такі як екран авторизації, реєстрації, головний екран, екран користувачів, діалогу та інші. Важливо відзначити, що дизайн кожного екрану був розроблений з урахуванням зручності та інтуїтивного зрозуміння для користувачів.

Описано процеси реєстрації, авторизації та адаптивного інтерфейсу. Використані різні інструменти та бібліотеки, такі як NextAuth.js, react-hook-form, та axios для забезпечення безпеки та функціональності системи.

В деталях продемонстровано реалізацію реально-часової взаємодії. Використання бібліотеки Pusher для забезпечення миттєвого обміну повідомленнями між користувачами. Описано ключові компоненти, такі як відображення повідомлень, підписка на канал Pusher та обробка нових повідомлень.

Проведено всебічне тестування для забезпечення якості та зручності використання. Успішно виконані всі основні та розширені сценарії, що підтверджують правильність роботи функціонала.

Ці аспекти демонструють успішну реалізацію та тестування вебзастосунка, що відповідає вимогам якості та функціональності.

ВИСНОВКИ

В рамках виконання кваліфікаційної роботи бакалавра було створено ефективний та зручний інструменту комунікації за допомогою реалізації Real-time чат-месенджера. Застосовано сучасні технології, зокрема Next.js 14, React, Node JS JavaScript, TypeScript, TailwindCSS, Prisma, MongoDB, NextAuth та Pusher, що дозволило забезпечити високу швидкість роботи, надійність, безпеку та зручність використання для користувачів.

Для досягнення поставленої мети було виконано наступні завдання:

– проведено аналіз предметної області та чинних аналогів браузерних чат-застосунків;

– визначено основні проблеми та знайдено шляхи їх вирішення;

– сформульовано функціональні вимоги до застосунку;

– розроблено блок-схеми алгоритмів роботи застосунку та спроєктовано архітектуру застосунку, створено необхідні UML-діаграми;

– спроєктовано дизайн інтерфейсу користувача та макети основних екранів застосунку;

– зібрано та проаналізовано дані для методичної частини;

– розроблено front-end частину вебзастосунку застосунку;

– реалізовано back-end частину вебзастосунку застосунку з використанням MongoDB та NextAuth для авторизації та бази даних.

Отримані результати демонструють важливість розробки та впровадження real-time чат-месенджерів у різноманітних сферах діяльності, які потребують швидкої та надійної комунікації. Розроблений чат-месенджер ChatApp є ефективним інструментом для полегшення спілкування в онлайн-режимі та забезпечення доступу до інформації в реальному часі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. High Performance Browser Networking: What every web developer should know about networking and web performance. URL: https://z-lib.id/book/high-performance-browser-networking-449496 (Last accessed: 22.02.2024).

2. How to Build a Real-Time React Chat Application: A Step-by-Step Guide. URL: https://www.dhiwise.com/post/how-to-build-a-real-time-react-chat-application (Last accessed: 24.02.2024).

3. Building a Real-time Chat Application with React and WebSocket. URL: https://dev.to/amarondev/building-a-real-time-chat-application-with-react-and-websocket-3138 (Last accessed: 25.02.2024).

4. Tutorial: Build a React.js Chat App. URL: https://www.scaledrone.com/blog/tutorial-build-a-reactjs-chat-app/ (Last accessed: 26.02.2024).

5. Building a real-time chat application using Node.js, MongoDB, and Express. URL: https://dev.to/manthanank/building-a-real-time-chat-application-using-nodejs-mongodb-and-express-3bhp (Last accessed: 27.02.2024).

6. Turan M. Integrating software metrics with UML class diagrams. Lecture notes on software engineering. 2015. Т. 3, № 3. С. 220–224. URL: https://doi.org/10.7763/lnse.2015.v3.194 (Last accessed: 28.02.2024).

7. More P., Phalnikar R. Generating UML diagrams from natural language specifications. International journal of applied information systems. 2012. Т. 1, № 8. С. 19–23. URL: https://doi.org/10.5120/ijais12-450222 (Last accessed: 02.03.2024).

8. Test case generation from UML- diagrams using genetic algorithm / R. Kumar Sahoo та ін. Computers, materials & continua. 2021. Т. 67, № 2. С. 2321– 2336. URL: https://doi.org/10.32604/cmc.2021.013014 (Last accessed: 23.03.2024).

9. Gnesi S., Latella D., Massink M. Model checking UML Statechart diagrams using JACK. HASE 99: fourth IEEE international symposium on high assurance systems engineering, м. Washington, DC, USA. URL: https://doi.org/10.1109/hase.1999.809474 (Last accessed: 02.04.2024).

10. Mokarat C., Vatanawood W. UML component diagram to acme compiler. 2013 international conference on information science and applications (ICISA), м. Suwon, Korea (South), 24–26 черв. 2013 р. 2013. URL: https://doi.org/10.1109/icisa.2013.6579469 (Last accessed: 02.04.2024).

11. Документація React. URL: https://react.dev/ (Last accessed: 03.04.2024).

12. Документація Next.js. URL: https://nextjs.org/docs (Last accessed: 03.04.2024).

13. Блог Next.js. URL: https://nextjs.org/blog (Last accessed: 03.04.2024).

14. Complete Guide To Node Client-Server Communication. URL: https://medium.com/@joekarlsson/complete-guide-to-node-client-server-communication-b156440c029 (Last accessed: 03.04.2024). Last acces

15. Документація Tailwind. URL: https://tailwindcss.com/docs/installation (Last accessed: 05.03.2024).

16. Документація Prisma. URL: https://www.prisma.io/docs (Last accessed: 05.04.2024).

17. Документація MongoDB. URL: https://www.mongodb.com/docs/ (Last accessed: 05.04.2024).

18. Документація Pusher. URL: https://pusher.com/docs/ (Last accessed: 06.04.2024).

19. MongoDB: The Complete Developer's Guide. URL: https://www.udemy.com/course/mongodb-the-complete-developers-guide/ (Last accessed: 06.04.2024).

20. Real-time Communication in Web Applications by Adam Barth. URL: https://www.adambarth.com/papers/2008/barth-jackson-mitchell.pdf (Last accessed: 08.04.2024).

21. Building Scalable Web Applications by Martin Kleppmann. URL: https://www.amazon.com/Designing-Data-Intensive-Applications-Reliable-Maintainable/dp/1449373321 (Last accessed: 10.04.2024).

22. Component diagram. URL: https://www.lucidchart.com/pages/uml-component-diagram (Last accessed: 29.04.2024).

23. Deployment diagram. URL: https://www.lucidchart.com/pages/uml-deployment-diagram (Last accessed: 29.04.2024).

24. Package diagram. Monographs in computer science. New York, NY. С. 133–154. URL: https://doi.org/10.1007/0-387-23803-4_7 (дата звернення: 03.04.2024).