Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук

Допущено до захисту
Завідувач кафедри комп’ютерних наук
доктор технічних наук, професор
Андрій БОНДАРЧУК
«01» червня 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»
спеціальність 122 Комп’ютерні науки
освітня програма 122.00.01 Інформатика

Тема: ВЕБЗАСТОСУНОК ОБМІНУ ПОВІДОМЛЕННЯМИ З ПІДТРИМКОЮ ТЕХНОЛОГІЇ SOCKET.IO

Виконав
студент групи Інб-1-22-4.0д
Биков А.О.

_____
(підпис)
Науковий керівник
кандидат педагогічних наук, доцент
Вембер В.П.

_____
(підпис)

Київ - 2026

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук

«Затверджую»
Завідувач кафедри комп’ютерних наук
доктор технічних наук, професор
Андрій БОНДАРЧУК
31.10.2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА
«Вебзастосунок обміну повідомленнями з підтримкою технології socket.io»

Виконавець - студент групи ІНб-1-22-4.0д,
спеціальності 122 Комп’ютерні наук,
освітньої програми 122.00.01 Інформатика
Биков Артур Олексійович

1. Вихідні дані: Законодавство та нормативно-правові акти України в навчальній сфері, наукові роботи та публікації за напрямом дослідження, зокрема з педагогіки та інформатики.

2. Основні завдання: проаналізувати існуючі альтернативи програмного забезпечення. Визначити та обґрунтувати мету, об'єкт, предмет і наукові основи дослідження. Розробити завдання, структуру та зміст бакалаврської роботи. Обрати і аргументувати методи та засоби дослідження. Підготувати матеріали для розділів роботи відповідно до індивідуального завдання. Проаналізувати потреби користувачів месенджерів. Проаналізувати та обрати технології для розробки свого месенджера. Скласти висновки та оформити бакалаврську роботу відповідно до затверджених на кафедрі вимог.

3. Пояснювальна записка: обсяг - до 42 стор. формату А4 комп’ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри.

4. Графічні матеріали: комп'ютерна презентація доповіді.

5. Додатки: реалізація функціоналу системи

6. Строк подання роботи на кафедру: «1» червня 2026 р.

Науковий керівник	Виконавець:
к.п.н., доцент	
_____ Вембер В.П.	_____ Биков А.О.
_____ дата	_____ дата

КАЛЕНДАРНИЙ ПЛАН

№	Етап виконання роботи	Термін виконання	Примітка
1	Затвердження теми кваліфікаційної роботи бакалавра	31.10.25	виконано
2	Аналіз проблеми, вивчення аналогів, формування цілей і завдань	01.11.25 - 11.01.26	виконано
3	Аналіз сучасних систем обміну повідомленнями, існуючих підходів до реалізації вебчатів, функціональних можливостей месенджерів та обґрунтування вибору технологічного стеку MERN	26.01.26 - 15.03.26	виконано
4	Дослідження та проєктування загальної архітектури програмної системи, структури бази даних MongoDB, REST API, моделей взаємодії користувачів, механізмів авторизації та обміну повідомленнями в реальному часі	16.03.26 - 31.03.26	виконано
5	Розробка клієнтської та серверної частин вебзастосунку Chatico, реалізація функціоналу реєстрації, авторизації, приватних і групових чатів, а також інтеграція Socket.IO для обміну повідомленнями в реальному часі	01.04.26 - 15.04.26	виконано
6	Тестування основних функціональних сценаріїв роботи застосунку, перевірка механізмів безпеки, авторизації користувачів, роботи чатів, сповіщень, індикатора набору тексту та коректності взаємодії клієнтської і серверної частин	16.04.26 - 30.04.26	виконано
7	Впровадження розробленого програмного продукту, фінальне налаштування взаємодії з базою даних, серверною частиною, WebSocket-з’єднанням та підготовка вебзастосунку до дослідної експлуатації	01.05.26 - 15.05.26	виконано
8	Підготовка пояснювальної записки, графічних матеріалів, діаграм, додатків та оформлення результатів кваліфікаційної роботи	25.05.26 - 12.06.26	виконано
9	Захист кваліфікаційної роботи	17.06.26	

Студент Биков А.О

Керівник роботи Вембер В.П.

РЕФЕРАТ

Кваліфікаційна робота: 45 с., 7 рис., 37 посилання, 1 таблиця.

Актуальність: зростання потреби у швидкому та зручному обміні повідомленнями в реальному часі між користувачами та необхідність створення сучасних веб-застосунків для ефективної комунікації.

Одним із найбільш зручних форматів такої взаємодії є вебчати, оскільки вони дають змогу обмінюватися повідомленнями без встановлення окремого програмного забезпечення та можуть працювати безпосередньо у браузері. Для сучасних чат-застосунків важливими є не лише надсилання текстових повідомлень, а й стабільність роботи, швидкість доставки повідомлень, зручність інтерфейсу, захист користувацьких даних, а також можливість розширення функціональності в майбутньому.

Об’єкт дослідження: система обміну повідомленнями в реальному часі.

Предмет дослідження: процес розробки та використання веб-застосунку для обміну повідомленнями з використанням сучасних веб-технологій для забезпечення ефективної взаємодії між користувачами.

Мета даної роботи: розробити веб-застосунок для обміну повідомленнями в реальному часі з підтримкою приватних і групових чатів, що забезпечує комунікацію між користувачами та демонструє можливості використання MERN-стека для створення сучасних веб-систем.

Завдання:

1. Проаналізувати існуючі системи обміну повідомленнями та їх функціональні можливості.
2. Визначити основні функціональні вимоги до веб-застосунку.
3. Обрати технології для реалізації системи обміну повідомленнями.
4. Розробити веб-додаток для обміну повідомленнями в реальному часі.

Основні результати: проаналізовані сучасні підходи до створення систем обміну повідомленнями, визначені функціональні вимоги до системи, обрано технології для розробки, реалізований веб-застосунок для обміну

повідомленнями з підтримкою авторизації користувачів, приватних і групових чатів, повідомлень у реальному часі та керування користувачами.

Практичне значення дослідження: можливість використання розробленого веб-застосунку як основи для створення повноцінної системи обміну повідомленнями або інтеграції функціоналу чатів у інші інформаційні системи.

Ключові слова: веб-застосунок, обмін повідомленнями, чат, реальний час, MERN-стек, Socket.io, веб-технології.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скорочення	Розшифрування
bcrypt	бібліотека для хешування паролів користувачів
BSON	Binary JSON - бінарний формат зберігання документів у MongoDB
CRUD	Create, Read, Update, Delete - базові операції роботи з даними
CSS	Cascading Style Sheets - каскадні таблиці стилів
DOM	Document Object Model - об’єктна модель документа вебсторінки
E2E	End-to-End - наскрізне тестування повного сценарію роботи системи
JWT	JSON Web Token - токен для автентифікації та авторизації
MERN	MongoDB, Express.js, React, Node.js - стек технологій для full-stack веброзробки
MongoDB	документоорієнтована система керування базами даних
Mongoose	ODM-бібліотека для роботи Node.js-застосунків із MongoDB
npm	Node Package Manager - менеджер пакетів для JavaScript
ODM	Object Document Mapper - технологія зіставлення об’єктів із документами бази даних
PWA	Progressive Web App - прогресивний вебзастосунок
React Router	бібліотека для організації маршрутизації у React-застосунках
Socket.IO	бібліотека для реалізації двосторонньої подієвої комунікації в реальному часі
TLS	Transport Layer Security - протокол захисту передавання даних
WebSocket	протокол двостороннього обміну даними між клієнтом і сервером у реальному часі

ЗМІСТ

ВСТУП

Онлайн-комунікація сьогодні є одним із найпоширеніших способів взаємодії між людьми. Обмін повідомленнями використовується не лише для особистого спілкування, а й у навчанні, роботі, технічній підтримці, командній розробці та бізнес-процесах. Користувачі очікують швидкого доступу до інформації, можливості миттєво отримувати відповіді та зручно підтримувати зв’язок незалежно від місця перебування. За даними звіту DataReportal, наприкінці 2025 року в Україні налічувалося 35,3 млн користувачів інтернету, а рівень проникнення інтернету становив 89,6 %, що підтверджує актуальність цифрових сервісів для комунікації [1].

Актуальність: зростання потреби у швидкому та зручному обміні повідомленнями в реальному часі між користувачами та необхідність створення сучасних веб-застосунків для ефективної комунікації.

Одним із найбільш зручних форматів такої взаємодії є вебчати, оскільки вони дають змогу обмінюватися повідомленнями без встановлення окремого програмного забезпечення та можуть працювати безпосередньо у браузері. Для сучасних чат-застосунків важливими є не лише надсилання текстових повідомлень, а й стабільність роботи, швидкість доставки повідомлень, зручність інтерфейсу, захист користувацьких даних, а також можливість розширення функціональності в майбутньому.

Умежах кваліфікаційної роботи розглядається розробка веб-застосунку Chatico, призначеного для обміну повідомленнями між користувачами в режимі реального часу. Застосунок дає змогу створювати приватні та групові чати, надсилати й отримувати повідомлення без перезавантаження сторінки, переглядати профілі інших користувачів, а також керувати груповими бесідами. Такий набір функцій дозволяє використовувати систему як для індивідуального спілкування, так і для комунікації між кількома учасниками.

Для реалізації обміну повідомленнями в реальному часі у проєкті використано технологію WebSocket. Вона забезпечує двосторонній

інтерактивний зв’язок між браузером користувача та сервером, що дозволяє передавати дані без постійного опитування сервера з боку клієнта [2]. На практиці це є важливим для чатів, оскільки нові повідомлення мають з’являтися одразу після їх надсилання, без ручного оновлення сторінки.

Для спрощення роботи з подіями реального часу було використано бібліотеку Socket.io. Вона підтримує двосторонню подієву комунікацію між клієнтом і сервером та може використовувати WebSocket як один із транспортних механізмів [3]. У межах проєкту це дозволило реалізувати миттєве отримання повідомлень, оновлення стану чатів та індикатори активності користувачів.

Клієнтську частину застосунку реалізовано з використанням React.js. Ця бібліотека дозволяє створювати інтерфейс із незалежних компонентів, які можна повторно використовувати в різних частинах застосунку [4]. Такий підхід є зручним для розробки чат-системи, оскільки окремими компонентами можуть бути список чатів, вікно повідомлень, форма введення, профіль користувача та елементи керування групою.

Серверна частина побудована на основі Node.js та Express.js. Node.js є асинхронним подієво-орієнтованим середовищем виконання JavaScript, призначеним для створення масштабованих мережевих застосунків [5]. Express.js, зі свого боку, є мінімалістичним вебфреймворком для Node.js, який надає інструменти для створення вебзастосунків та API [6]. У проєкті ці технології використовуються для обробки HTTP-запитів, авторизації користувачів, роботи з чатами, повідомленнями та підключеннями в реальному часі.

Для зберігання даних застосунку використано MongoDB. Ця система керування базами даних зберігає записи у вигляді документів BSON, об’єднаних у колекції [7]. Така модель добре підходить для чат-застосунку, оскільки дані користувачів, чатів і повідомлень мають гнучку структуру та можуть змінюватися під час подальшого розвитку системи.

Для створення інтерфейсу користувача застосовано Chakra UI - бібліотеку компонентів для React, яка надає готові елементи інтерфейсу та допомагає швидше створювати адаптивні вебзастосунки [8]. Її використання дозволило зосередитися не лише на функціональній частині проєкту, а й на зручності взаємодії користувача із системою.

Об’єкт дослідження: система обміну повідомленнями в реальному часі.

Предмет дослідження: процес розробки та використання веб-застосунку для обміну повідомленнями з використанням сучасних веб-технологій для забезпечення ефективної взаємодії між користувачами.

Мета: розробити веб-застосунок для обміну повідомленнями в реальному часі з підтримкою приватних і групових чатів, що забезпечує комунікацію між користувачами та демонструє можливості використання MERN-стека для створення сучасних веб-систем.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Проаналізувати існуючі системи обміну повідомленнями та їх функціональні можливості.
2. Визначити основні функціональні вимоги до веб-застосунку.
3. Обрати технології для реалізації системи обміну повідомленнями.
4. Розробити веб-додаток для обміну повідомленнями в реальному часі.

Основні результати: проаналізовані сучасні підходи до створення систем обміну повідомленнями, визначені функціональні вимоги до системи, обрано технології для розробки, реалізований веб-застосунок для обміну повідомленнями з підтримкою авторизації користувачів, приватних і групових чатів, повідомлень у реальному часі та керування користувачами.

Практичне значення дослідження: можливість використання розробленого веб-застосунку як основи для створення повноцінної системи обміну повідомленнями або інтеграції функціоналу чатів у інші інформаційні системи.

Результати роботи апробовано шляхом публікації тез доповіді на конференції Інформаційні технології - 2026: зб. тез XII Всеукраїнської науково-

практичної конференції молодих учених, 14 трав. 2025 р., м. Київ / Київський столичний університет імені Бориса Грінченка [34]

РОЗДІЛ 1

ТЕОРЕТИЧНЕ ПІДГРУНТТЯ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд аналогів програмних засобів

Перед початком розробки вебзастосунку для обміну повідомленнями доцільно проаналізувати вже існуючі програмні рішення у цій сфері(рис 1.1). Такий аналіз дає змогу визначити, які функції є базовими для сучасних месенджерів, які підходи використовуються для організації комунікації між користувачами, а також які можливості варто врахувати під час проєктування власної системи.

До найпоширеніших рішень для миттєвого обміну повідомленнями можна віднести Telegram, WhatsApp і Discord. Кожен із цих сервісів має власну цільову аудиторію, набір функцій та особливості реалізації. Порівняння таких систем із розроблюваним застосунком Chatico дозволяє краще визначити місце проєкту серед аналогічних програмних засобів.

Критерій	WhatsApp	Telegram	Slack	Discord	Chatico
Чат один на один	так	так	так	так	так
Групові чати	так	так	так	так	так
Відеодзвінки	так	ні	так	так	ні
Реальний час (WebSocket / Socket.io)	так	так	так(WebSocket)	так	так(Socket.io)
Пошук користувачів	ні	так	так	так	так
Індикатор набору тексту	так	так	ні	так	так
Перегляд профілю	ні	так	так	так	так
Захист даних (шифрування)	так(E2E)	так(E2E, частково)	так(TLS)	так(TLS)	так(bcrypt + JWT)
Сповіщення	так	так	так	так	так
Додавання/видалення з групи	так	так	так	так	так

Рисунок 1.1. Порівняння з інших месенджерів з Chatico

Telegram є популярним хмарним месенджером, який працює на мобільних пристроях, настільних комп’ютерах і у вебверсії. Сервіс підтримує приватні

чати, групи, канали, обмін файлами, створення ботів, а також має відкриті API для розробників. В офіційних матеріалах Telegram зазначено, що платформа орієнтується на швидкість, безпеку та відкритість для інтеграцій [9].

Окремою особливістю Telegram є використання протоколу MTProto, який застосовується для захисту передавання даних. Також у Telegram передбачені “секретні чати”, у яких використовується наскрізне шифрування між учасниками розмови [10]. Це робить Telegram не лише засобом для звичайного спілкування, а й платформою для побудови додаткових сервісів, ботів та інтеграцій.

Водночас Telegram є готовим комерційним продуктом із власною закритою серверною інфраструктурою. Користувач не має повного контролю над серверною частиною, логікою зберігання даних і механізмами обробки повідомлень. На відміну від нього, Chatico розглядається як власна вебсистема, у якій архітектура, база даних і логіка обміну повідомленнями реалізуються в межах дипломного проєкту.

WhatsApp є одним із найбільш масових месенджерів, який орієнтований переважно на мобільне використання. Він підтримує текстові повідомлення, голосові та відеодзвінки, надсилання зображень, документів та інших файлів. Важливою особливістю WhatsApp є наскрізне шифрування особистих повідомлень і дзвінків, завдяки якому вміст розмови доступний лише її учасникам [11].

Перевагою WhatsApp є простота використання та велика кількість користувачів. Саме завдяки цьому він часто використовується для повсякденної комунікації. Однак цей сервіс має закритий вихідний код і обмежені можливості для зміни внутрішньої логіки роботи. Крім того, вебверсія WhatsApp здебільшого виступає як доповнення до основного мобільного застосунку, а не як повністю самостійна система.

У порівнянні з WhatsApp, застосунок Chatico орієнтований саме на вебдоступ через браузер. Це дозволяє використовувати його без встановлення мобільного застосунку та дає змогу розробнику повністю контролювати

функціональність, структуру бази даних, механізми авторизації та подальше розширення системи.

Discord - це платформа для текстового, голосового та відеоспілкування, яка широко використовується спільнотами, ігровими групами, навчальними командами та робочими колективами. Основою Discord є сервери, всередині яких можуть створюватися окремі текстові й голосові канали. Такий підхід дозволяє структурувати спілкування за темами, ролями або напрямами роботи [12].

Сильною стороною Discord є гнучка організація спільнот. У ньому можна створювати ролі, налаштовувати права доступу, використовувати голосові канали, відеозв'язок, демонстрацію екрана та інтегрувати ботів. Завдяки цьому Discord підходить для великих груп користувачів і складних комунікаційних сценаріїв.

Разом із тим така функціональність робить Discord складнішим як для користувача, так і з погляду програмної реалізації. Для невеликого вебзастосунку, метою якого є обмін повідомленнями між користувачами, частина можливостей Discord може бути надлишковою. Саме тому Chatico має простішу структуру: приватні чати, групові бесіди, повідомлення в реальному часі, пошук користувачів і базове керування групами.

На основі аналізу аналогів можна зробити висновок, що більшість сучасних месенджерів мають спільний базовий набір функцій: реєстрацію або вхід користувача, створення чатів, обмін повідомленнями, підтримку групового спілкування, сповіщення та елементи захисту даних. Водночас кожна система має власний акцент: Telegram зосереджується на швидкості, ботах та API, WhatsApp - на простоті й приватності, Discord - на організації спільнот і голосовій комунікації.

Застосунок Chatico реалізує основні можливості, характерні для подібних систем, але має навчально-практичну спрямованість. Його мета полягає не в повному дублюванні функціональності великих комерційних платформ, а в

демонстрації принципів побудови сучасного вебчату з використанням клієнт-серверної архітектури, бази даних, REST API та WebSocket-з’єднання.

Основною перевагою Chatico в межах дипломного проєкту є те, що всі ключові частини системи розробляються та контролюються безпосередньо: клієнтський інтерфейс, серверна логіка, структура бази даних, авторизація користувачів і механізм обміну повідомленнями в реальному часі.

1.2 Загальна характеристика програмного продукту

Програмний продукт Chatico є вебзастосунком для обміну повідомленнями в режимі реального часу. Він розроблений із використанням сучасних вебтехнологій і призначений для організації текстового спілкування між зареєстрованими користувачами.

Основна ідея застосунку полягає в тому, щоб надати користувачеві простий і зрозумілий інструмент для комунікації через браузер. На відміну від багатьох готових месенджерів, Chatico не потребує встановлення окремого мобільного або настільного застосунку. Користувач може перейти на сайт, авторизуватися та одразу почати спілкування.

У системі передбачено можливість створення приватних чатів між двома користувачами, а також групових бесід для комунікації кількох учасників. Повідомлення передаються без перезавантаження сторінки, що забезпечує більш зручну взаємодію та наближає роботу застосунку до звичних месенджерів.

Chatico також містить функції пошуку користувачів, перегляду профілю, відображення сповіщень, індикатора набору тексту та базового керування групами. Такий набір можливостей є достатнім для демонстрації роботи повноцінного вебчату та може бути розширений у майбутньому.

1.3 Основні компоненти системи

Застосунок Chatico побудований на основі клієнт-серверної архітектури. Такий підхід передбачає розділення відповідальності між клієнтською частиною, серверною частиною та базою даних. Клієнтська частина відповідає за

відображення інтерфейсу та взаємодію з користувачем, серверна - за обробку запитів і бізнес-логіку, а база даних - за зберігання інформації.

До основних компонентів системи належать:

1. Клієнтська частина

Frontend-частина застосунку реалізована з використанням бібліотеки React.js. Вона відповідає за побудову інтерфейсу користувача, відображення списку чатів, повідомлень, форм авторизації, профілю користувача та елементів керування групами. Для оформлення інтерфейсу використовується Chakra UI, що дозволяє швидше створювати адаптивні компоненти та підтримувати єдиний стиль застосунку.

2. Серверна частина

Backend-частина побудована на платформі Node.js із використанням Express.js. Сервер обробляє HTTP-запити, відповідає за реєстрацію та автентифікацію користувачів, роботу з чатами, повідомленнями й профілями. Крім REST API, сервер також обробляє WebSocket-з’єднання за допомогою Socket.io, що дозволяє реалізувати миттєвий обмін повідомленнями.

3. База даних

Для зберігання інформації використовується MongoDB. У базі даних зберігаються відомості про користувачів, чати, учасників груп і повідомлення. Документо-орієнтована структура MongoDB є зручною для такого типу застосунку, оскільки дані можуть мати різну вкладеність і гнучко змінюватися під час розвитку проєкту.

4. Модуль авторизації

Окрему роль у системі відіграє механізм авторизації користувачів. Він забезпечує перевірку облікових даних, створення токена доступу та обмеження доступу до захищених маршрутів. Це необхідно для того, щоб користувач міг працювати лише зі своїми чатами й повідомленнями.

5. Модуль обміну повідомленнями

Цей компонент відповідає за створення, надсилання, отримання та збереження повідомлень. Для повідомлень у реальному часі використовується Socket.io, а

для збереження історії листування - MongoDB. Завдяки цьому користувач може бачити нові повідомлення одразу після їх надсилання, а також переглядати попередню історію спілкування.

6. Вибір технологій

Вибір технологічного стеку для розробки Chatico був зумовлений вимогами до швидкості розробки, зручності підтримки, продуктивності та можливості подальшого масштабування.

React.js було обрано для клієнтської частини, оскільки компонентний підхід спрощує розробку інтерфейсу та дозволяє повторно використовувати окремі елементи застосунку. Для чат-системи це особливо зручно, оскільки інтерфейс складається з багатьох повторюваних блоків: списку чатів, повідомлень, аватарів, кнопок, модальних вікон і форм.

Node.js та Express.js використовуються для побудови серверної логіки. Такий вибір дозволяє реалізувати backend на JavaScript, тобто використовувати одну мову програмування як на клієнті, так і на сервері. Це спрощує розробку, підтримку коду та взаємодію між частинами системи.

MongoDB було обрано як базу даних через її гнучку документо-орієнтовану модель. Для чатів це є практичним рішенням, оскільки повідомлення, користувачі та групи можуть мати різну структуру, а вимоги до даних можуть змінюватися в процесі розвитку застосунку.

Socket.io використовується для реалізації обміну повідомленнями в реальному часі. Його застосування дозволяє передавати події між клієнтом і сервером без постійного оновлення сторінки. Саме цей механізм забезпечує основну функціональність Chatico як вебчату.

Для захисту облікових записів у серверній частині використовуються JWT і хешування паролів за допомогою bcrypt. JWT дозволяє передавати інформацію про автентифікованого користувача у вигляді токена, а bcrypt застосовується для безпечного зберігання паролів у вигляді хешів, а не відкритого тексту.

1.4 Призначення та цільова аудиторія

Застосунок Chatico призначений для організації текстового спілкування між користувачами через веббраузер. Основним сценарієм використання є швидкий обмін повідомленнями між двома користувачами або групою учасників без необхідності встановлення додаткового програмного забезпечення.

Цільовою аудиторією застосунку можуть бути звичайні користувачі, які потребують простого засобу для онлайн-спілкування, а також невеликі команди, навчальні групи або робочі колективи. Для таких користувачів важливими є простота доступу, зрозумілий інтерфейс, можливість створення групових чатів і швидке отримання повідомлень.

Крім того, Chatico може розглядатися як основа для подальшого розвитку більш складної системи комунікації. У майбутньому до нього можна додати надсилання файлів, голосові повідомлення, розширені налаштування профілю, систему ролей, пошук по повідомленнях та інші функції.

1.5 Основні можливості

У межах дипломного проєкту було реалізовано набір функцій, необхідних для роботи базового вебчату. До основних можливостей застосунку Chatico належать:

1. Реєстрація та автентифікація користувачів.
Користувач може створити обліковий запис, увійти до системи та отримати доступ до власних чатів.
2. Створення приватних чатів.
Система дозволяє створювати діалоги між двома користувачами для особистого спілкування.
3. Створення та керування груповими чатами.
Користувачі можуть створювати групові бесіди, додавати або видаляти учасників і змінювати параметри групи.
4. Надсиланняповідомлень у реальному часі.
Завдяки використанню Socket.io повідомлення з’являються у співрозмовника без перезавантаження сторінки.

5. Пошук інших користувачів.

У застосунку передбачено можливість пошуку користувачів для створення нових чатів.

6. Індикатор набору тексту.

Система може відображати, що співрозмовник набирає повідомлення, що робить комунікацію більш наближеною до звичних месенджерів.

7. Відображення сповіщень.

Користувач отримує інформацію про нові повідомлення, що дозволяє не пропускати активність у чатах.

8. Перегляд профілю користувача.

У застосунку реалізовано можливість перегляду базової інформації про інших користувачів.

Висновки до розділу 1

У першому розділі було розглянуто теоретичні основи створення вебзастосунків для обміну повідомленнями та визначено основні задачі, які необхідно вирішити під час розробки системи Chatico. Аналіз показав, що сучасні месенджери стали важливою частиною повсякденної комунікації, оскільки вони використовуються не лише для особистого спілкування, а й у навчанні, роботі, командній взаємодії та технічній підтримці.

Під час огляду аналогів було проаналізовано такі популярні сервіси, як Telegram, WhatsApp і Discord. Кожен із них має власні переваги: Telegram орієнтований на швидкість, хмарну синхронізацію та відкриті API, WhatsApp - на простоту використання і приватність, Discord - на організацію спільнот, групове спілкування та гнучке налаштування каналів. Водночас ці рішення є готовими комерційними продуктами, у яких користувач або розробник не має повного контролю над серверною логікою, структурою бази даних і внутрішніми механізмами обробки повідомлень.

На основі порівняння аналогів визначено, що розроблюваний застосунок Chatico має навчально-практичну спрямованість. Його мета полягає не в повному

дублюванні можливостей великих месенджерів, а в демонстрації принципів побудови власної системи обміну повідомленнями з використанням сучасних вебтехнологій. Саме тому в межах проєкту основний акцент зроблено на приватних і групових чатах, повідомленнях у реальному часі, пошуку користувачів, перегляді профілів, індикаторі набору тексту та базових сповіщеннях.

Окремо розглянуто загальну характеристику програмного продукту. Chatico визначено як вебзастосунок, що працює через браузер і не потребує встановлення окремої мобільної або настільної програми. Такий підхід є зручним для користувачів, оскільки дозволяє швидко отримати доступ до системи з різних пристроїв. Водночас вебформат дає змогу легше оновлювати застосунок і розширювати його функціональність у майбутньому.

У першому розділі також визначено основні компоненти системи: клієнтську частину, серверну частину, базу даних, модуль авторизації та модуль обміну повідомленнями. Клієнтська частина відповідає за інтерфейс і взаємодію з користувачем, серверна - за обробку запитів і бізнес-логіку, база даних - за зберігання користувачів, чатів і повідомлень. Модуль авторизації забезпечує захист доступу, а модуль повідомлень реалізує головну функцію застосунку - миттєву комунікацію між користувачами.

У підсумку, у першому розділі сформовано загальне уявлення про призначення Chatico, його цільову аудиторію, функціональність і місце серед аналогічних систем. Проведений аналіз став основою для подальшого визначення функціональних і нефункціональних вимог, а також для вибору архітектури та технологій реалізації застосунку.

РОЗДІЛ 2

ПРОЄКТУВАННЯ ТА ВИМОГИ ДО ЗАСТОСУНКУ

2.1 Опис функціональних вимог

У цьому розділі розглянуто основні функціональні вимоги до вебзастосунку **Chatico**. Вони визначають, які дії може виконувати користувач у системі та які можливості повинні бути реалізовані для повноцінної роботи чату. Основний акцент зроблено на реєстрації користувачів, створенні чатів, обміні повідомленнями в реальному часі, пошуку користувачів, роботі з групами та відображенні сповіщень.

Для наочного подання основних дій користувача в системі було побудовано діаграму варіантів використання. Вона відображає взаємодію користувача із застосунком Chatico, зокрема реєстрацію та вхід, створення приватного й групового чату, надсилання повідомлень, пошук користувачів, перегляд профілю, отримання сповіщень і керування групою.

Реєстрація та автентифікація користувача

Однією з базових функцій системи є реєстрація та вхід користувача. У застосунку Chatico користувач може створити обліковий запис, указавши ім’я, електронну пошту та пароль. Електронна пошта використовується як унікальний ідентифікатор, що дозволяє уникнути дублювання облікових записів.

Для підвищення безпеки паролі не зберігаються у відкритому вигляді. Перед записом у базу даних вони хешуються за допомогою бібліотеки bcrypt. Такий підхід дозволяє зменшити ризики у випадку несанкціонованого доступу до бази даних, оскільки замість реальних паролів зберігаються їх хеші [13].

Після успішної авторизації користувач отримує JWT-токен. JSON Web Token є компактним способом передавання інформації між сторонами у вигляді JSON-об’єкта, який може бути підписаний і перевірений сервером [14]. У Chatico токен використовується для доступу до захищених маршрутів API. Перед

виконанням операцій сервер перевіряє валідність токена через middleware, а також контролює, чи має користувач право доступу до відповідного чату.

Створення та ведення чатів.

У застосунку передбачено створення приватних і групових чатів. Приватний чат створюється після вибору іншого користувача через пошук. Якщо діалог між цими користувачами вже існує, система не створює новий чат, а відкриває наявний. Це дозволяє уникнути дублювання діалогів і підтримувати впорядковану структуру листування.

Груповий чат може створити будь-який зареєстрований користувач. Для цього він обирає кількох учасників і задає назву групи. Користувач, який створив групу, автоматично отримує права адміністратора. Усі дані про чати, їх учасників і пов’язані повідомлення зберігаються в базі даних MongoDB.

Обмін повідомленнями в реальному часі

Ключовою можливістю Chatico є надсилання та отримання повідомлень без перезавантаження сторінки. Для цього використовується Socket.io - бібліотека, яка забезпечує двосторонню подієву комунікацію між клієнтом і сервером [15].

Повідомлення передаються у вигляді подій, наприклад send message та receive message. Коли користувач надсилає повідомлення, клієнт передає його на сервер, після чого сервер зберігає повідомлення в базі даних і надсилає його іншим учасникам відповідного чату. Завдяки цьому співрозмовники бачать нові повідомлення майже одразу після їх надсилання.

Збереження повідомлень у базі даних дозволяє не лише передавати їх у реальному часі, а й формувати історію листування. Це важливо для зручності користувачів, оскільки після повторного входу до системи вони можуть переглядати попередні повідомлення.

Пошук користувачів

Для швидкого початку спілкування в Chatico реалізовано пошук користувачів. Пошук може виконуватися за іменем або електронною поштою. Клієнтська частина надсилає REST-запит на сервер із пошуковим параметром,

після чого сервер повертає список користувачів, які відповідають введеному значенню.

Результати пошуку відображаються у зручному списку, де користувач може вибрати потрібну людину та створити з нею приватний чат. Такий механізм спрощує навігацію в системі та скорочує час, необхідний для початку діалогу.

Індикатор набору тексту

Для підвищення інтерактивності в застосунку реалізовано індикатор набору тексту. Коли користувач починає вводити повідомлення, клієнтська частина надсилає на сервер подію typing. Сервер передає цю подію іншим учасникам чату, після чого в інтерфейсі з’являється повідомлення про те, що співрозмовник набирає текст.

Коли користувач припиняє введення, надсилається подія stop typing, і відповідний індикатор зникає. Така функція робить спілкування більш природним і наближеним до звичних месенджерів.

Управління групами

У групових чатах реалізовано базові можливості керування учасниками. Адміністратор групи може додавати нових користувачів, видаляти учасників і змінювати назву групового чату. Це дозволяє підтримувати порядок у групі та контролювати її склад.

Кожен учасник має можливість вийти з групи самостійно. Після виходу користувач втрачає доступ до нових повідомлень у цьому чаті, однак дані групи зберігаються для інших учасників. Такий підхід дозволяє організувати просте, але достатньо гнучке керування груповими бесідами.

Відображення сповіщень

Система сповіщень призначена для інформування користувача про нові повідомлення. Якщо повідомлення надходить у чат, який наразі не відкритий, користувач бачить відповідне сповіщення або індикатор непрочитаних повідомлень біля назви чату.

Оновлення сповіщень відбувається динамічно завдяки Socket.io. Це дозволяє не перезавантажувати сторінку та одразу показувати зміни в інтерфейсі.

На клієнтському рівні стан сповіщень може зберігатися за допомогою контекстів React або подібних механізмів керування станом.

Перегляд профілю користувача

Chatico також дозволяє переглядати базову інформацію про користувачів. При натисканні на ім’я або аватар відкривається модальне вікно, у якому відображаються ім’я, електронна пошта та зображення профілю, якщо воно було додане користувачем.

Дані профілю отримуються з бази даних через REST API. Ця функція допомагає краще ідентифікувати співрозмовника та робить інтерфейс більш персоналізованим.

2.2 Нефункціональні вимоги

Нефункціональні вимоги визначають не окремі дії користувача, а загальні характеристики системи. Вони впливають на стабільність, безпеку, продуктивність, масштабованість і зручність використання застосунку Chatico.

Безпека

Безпека є важливою вимогою для застосунку, який працює з обліковими записами користувачів і зберігає історію повідомлень. У Chatico реалізовано хешування паролів за допомогою bcrypt, що дозволяє не зберігати паролі у відкритому вигляді [13].

Для авторизації використовується JWT. Після входу в систему користувач отримує токен, який додається до запитів під час звернення до захищених ресурсів. Сервер перевіряє токен і визначає, чи має користувач право виконувати певну дію [14].

Також у системі передбачено обмеження доступу до чатів. Користувач не може переглядати або змінювати чат, учасником якого він не є. Це дозволяє захистити приватність листування та запобігти несанкціонованому доступу до чужих повідомлень.

Продуктивність

Продуктивність застосунку залежить від швидкості обробки запитів, роботи бази даних і передачі подій у реальному часі. Для зменшення навантаження на базу даних запити до MongoDB мають повертати лише необхідні поля, а для частих операцій можуть використовуватися індекси. У документації MongoDB зазначено, що індекси допомагають ефективніше виконувати запити, оскільки обмежують кількість документів, які потрібно переглянути під час пошуку [16].

Передача повідомлень у реальному часі реалізована через Socket.io, що дозволяє зменшити затримку між надсиланням і відображенням повідомлення. На сервері та клієнті використовуються асинхронні операції, щоб не блокувати виконання інших дій під час обробки запитів або очікування відповіді від бази даних.

Масштабованість

Архітектура Chatico побудована з урахуванням можливості подальшого розширення. Окреме розділення клієнтської частини, серверної логіки та бази даних дозволяє додавати нові функції без повної перебудови системи.

У майбутньому застосунок можна масштабувати шляхом розгортання кількох екземплярів серверної частини, використання балансування навантаження або контейнеризації за допомогою Docker. Також можлива інтеграція з додатковими сервісами, наприклад email-розсилками, push-сповіщеннями або системами зберігання файлів.

Кросплатформеність

Клієнтська частина застосунку розроблена для роботи в сучасних веббраузерах. Це дозволяє використовувати Chatico на різних пристроях: персональних комп’ютерах, ноутбуках, планшетах і смартфонах.

Підхід Progressive Web App дає змогу створювати вебзастосунки, які можуть поводитися подібно до встановлених програм, зокрема підтримувати адаптивність, роботу на різних пристроях і покращений користувацький досвід [17]. У поєднанні з Chakra UI це дозволяє забезпечити зручне відображення інтерфейсу на екранах різного розміру.

Зручність інтерфейсу

Інтерфейс Chatico спроєктовано з урахуванням простоти та зрозумілості для користувача. Основні елементи системи - список чатів, область повідомлень, поле введення тексту, профіль користувача та сповіщення - розміщені так, щоб користувач міг швидко виконувати основні дії.

Для побудови інтерфейсу використано компоненти Chakra UI. Вони дозволяють реалізувати адаптивні елементи, модальні вікна, випадаючі списки, кнопки та повідомлення про помилки або успішні дії. Це покращує загальний користувацький досвід і робить роботу із застосунком більш зручною.

Надійність

Надійність системи забезпечується обробкою помилок на клієнтському та серверному рівнях. На клієнті помилки можуть відображатися через повідомлення або toast-сповіщення, а на сервері - оброблятися через спеціальні middleware.

У разі тимчасового розриву з’єднання Socket.io може автоматично намагатися відновити підключення. В офіційній документації зазначено, що бібліотека має механізм heartbeat для перевірки стану з’єднання та автоматичне перепідключення з поступовою затримкою, щоб не перевантажувати сервер [15].

Крім цього, усі критичні дії в системі мають супроводжуватися перевіркою даних. Це стосується реєстрації користувача, входу в систему, створення чатів, надсилання повідомлень і керування групами. Такий підхід зменшує кількість помилок і підвищує стабільність роботи застосунку.

Висновки до розділу 2

У другому розділі було визначено вимоги до вебзастосунку Chatico. Саме цей етап є важливим переходом від загального опису системи до її практичного проєктування, оскільки функціональні та нефункціональні вимоги задають межі майбутньої реалізації та допомагають зрозуміти, які можливості мають бути обов’язково підтримані.

До основних функціональних вимог віднесено реєстрацію та автентифікацію користувачів, створення приватних і групових чатів, надсилання повідомлень у реальному часі, пошук користувачів, індикатор набору тексту, керування групами, відображення сповіщень і перегляд профілю. Такий набір функцій є достатнім для базового, але повноцінного вебчату, який може використовуватися для індивідуального та групового спілкування.

Важливою частиною вимог є безпечна робота з обліковими записами. У системі передбачено хешування паролів за допомогою bcrypt, що дозволяє не зберігати паролі у відкритому вигляді. Для авторизації використовується JWT-токен, який надає доступ до захищених маршрутів API. Такий підхід дозволяє обмежити доступ до приватних даних і забезпечити, щоб користувач працював лише із тими чатами, учасником яких він є.

Окремо визначено вимоги до обміну повідомленнями в реальному часі. Для цього застосовується Socket.IO, який дозволяє передавати події між клієнтом і сервером без перезавантаження сторінки. Такий механізм є ключовим для чат-застосунку, оскільки користувач очікує, що повідомлення буде доставлене та відображене майже одразу після надсилання.

Нефункціональні вимоги охоплюють безпеку, продуктивність, масштабованість, кросплатформеність, зручність інтерфейсу та надійність. Для Chatico важливо не лише реалізувати базові дії користувача, а й забезпечити стабільну роботу системи, швидку обробку запитів, коректне відновлення з’єднання, захист даних і зручну роботу на різних пристроях. Саме ці характеристики впливають на якість використання застосунку.

У розділі також обґрунтовано доцільність використання клієнт-серверної архітектури, MongoDB, React, Node.js, Express.js і Socket.IO. Такий набір технологій відповідає задачам системи, оскільки дозволяє створити вебзастосунок із динамічним інтерфейсом, серверною логікою, документоорієнтованою базою даних і підтримкою подій у реальному часі.

У другому розділі було сформовано основу для реалізації Chatico. Визначені вимоги дозволяють перейти до практичної розробки системи, не

втрачаючи логіки її побудови. Завдяки цьому застосунок може бути реалізований як цілісна система, у якій функціональність, безпека та зручність використання взаємопов’язані між собою.

РОЗДІЛ 3

РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕСПЕЧЕННЯ

3.1 Архітектура та технічна реалізація

У цьому розділі розглянуто архітектуру програмного забезпечення Chatico, принципи взаємодії між основними компонентами системи, а також технічні засоби, використані під час розробки. Перелік основних технологій наведено в таблиці 3.1.

Загальна архітектура

Застосунок Chatico реалізовано за клієнт-серверною архітектурою. Такий підхід передбачає розділення системи на клієнтську частину, серверну частину та базу даних. Клієнт відповідає за відображення інтерфейсу й обробку дій користувача, сервер виконує бізнес-логіку, а база даних забезпечує зберігання інформації про користувачів, чати та повідомлення.

Взаємодія між клієнтом і сервером здійснюється двома способами: через HTTP-запити та через WebSocket-з’єднання. HTTP-запити використовуються для авторизації, реєстрації, пошуку користувачів, створення чатів і отримання даних. Для передавання повідомлень у реальному часі використовується Socket.io, який забезпечує двосторонню подієву комунікацію між клієнтом і сервером [15].

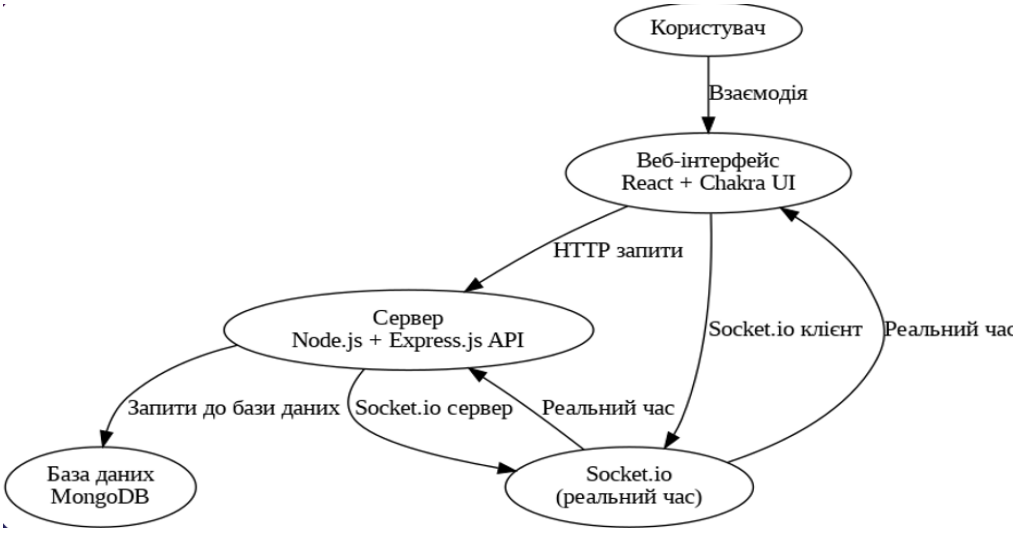

Рисунок 3.1. Схема функціонування застосунку Chatico

Клієнт ініціює запити до сервера для отримання або зміни даних. Сервер обробляє ці запити, перевіряє права доступу користувача, виконує необхідну логіку та звертається до бази даних. Після входу користувача в систему між клієнтом і сервером встановлюється з’єднання Socket.io, яке використовується для миттєвого обміну повідомленнями.

Компоненти системи

Архітектура застосунку включає кілька основних компонентів.

Клієнтська частина (frontend)

Клієнтська частина реалізована з використанням React.js. Вона відповідає за побудову інтерфейсу користувача, маршрутизацію між сторінками, відображення чатів, повідомлень, форм авторизації та модальних вікон. Для стилізації інтерфейсу використано Chakra UI, що дозволяє створювати адаптивні компоненти та забезпечувати єдиний візуальний стиль застосунку.

Для навігації між сторінками використовується React Router. Ця бібліотека застосовується для організації маршрутів у React-застосунках і дозволяє відображати різні сторінки залежно від URL та стану користувача [18].

Серверна частина (backend)

Серверна частина побудована на Node.js із використанням Express.js. Вона відповідає за обробку HTTP-запитів, реалізацію REST API, автентифікацію користувачів, перевірку прав доступу, роботу з чатами та повідомленнями. Сервер також підтримує Socket.io-з’єднання, через які передаються події в реальному часі.

Модуль обміну повідомленнями в реальному часі

Цей модуль реалізовано за допомогою Socket.io. Він забезпечує надсилання й отримання повідомлень без перезавантаження сторінки. Крім самих повідомлень, через Socket.io можуть передаватися службові події, наприклад індикатор набору тексту або оновлення активного чату.

База даних

Для зберігання даних використовується MongoDB. Взаємодія з базою реалізована через Mongoose. Mongoose надає схематичний підхід до

моделювання даних у MongoDB, підтримує створення моделей, валідацію, побудову запитів та роботу з документами [19]. У базі даних зберігається інформація про користувачів, чати, повідомлення та службові зв’язки між ними.

Взаємодія компонентів

Після авторизації користувач отримує JWT-токен, який використовується для подальших запитів до захищених маршрутів API. Коли користувач відкриває сторінку чатів, клієнтська частина надсилає запит до сервера для отримання списку доступних чатів та історії повідомлень.

Нові повідомлення передаються через Socket.io. Після надсилання повідомлення клієнт передає відповідну подію на сервер. Сервер обробляє її, зберігає повідомлення в MongoDB і надсилає його іншим учасникам чату. Завдяки цьому інтерфейс у всіх підключених користувачів оновлюється без перезавантаження сторінки.

Для ініціалізації з’єднання Socket.IO на серверній частині використовується фрагмент коду, який забезпечує підключення клієнта, приєднання користувача до власної кімнати та підключення до кімнати конкретного чату. Приклад реалізації цього фрагмента наведено у фрагменті коду нижче.

Фрагмент коду ініціалізації Socket.io

```
    const io = require("socket.io")(server, {
  pingTimeout: 60000,
  cors: { origin: "http://localhost:5000" },
});

io.on("connection", (socket) => {
  socket.on("setup", (userData) => {
    socket.join(userData._id);
    socket.emit("connected");
  });

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

Стек технологій

Взаємодія між компонентами через Socket.io

У цьому підрозділі розглянуто принцип роботи Socket.io в межах застосунку Chatico. Загальний алгоритм взаємодії показано на рисунку 3.2.

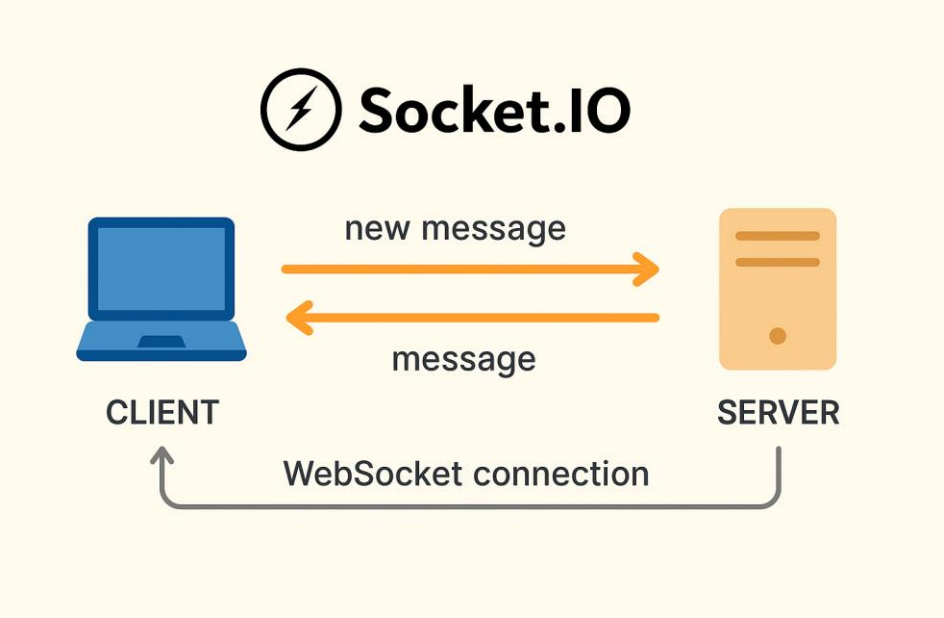

Рисунок 3.2. Принцип роботи Socket.io у застосунку Chatico

Алгоритм взаємодії має такий вигляд:

1. Користувач відкриває вебзастосунок у браузері.
2. Клієнт надсилає HTTP-запити до сервера для реєстрації, входу, пошуку користувачів або створення чатів.
3. Після авторизації між клієнтом і сервером встановлюється Socket.io-з’єднання.
4. Повідомлення в чатах передаються у вигляді подій у реальному часі.
5. Сервер обробляє події, зберігає повідомлення в MongoDB і надсилає їх іншим учасникам чату.
6. Клієнт оновлює інтерфейс відповідно до відповідей сервера або подій Socket.io.

Такий підхід дозволяє поєднати звичайну клієнт-серверну взаємодію через REST API з передаванням подій у реальному часі.

Структура бази даних

Структура бази даних Chatico складається з трьох основних сутностей: користувачів, чатів і повідомлень.

Основні колекції MongoDB:

Модель **User** зберігає інформацію про користувача: ім’я, електронну пошту, захешований пароль, аватар і дату створення облікового запису. Модель **Chat** описує приватний або груповий чат, його учасників, адміністратора групи та останнє повідомлення. Модель **Message** містить інформацію про автора повідомлення, текст повідомлення, пов’язаний чат і час створення.

Безпека та обробка помилок

У серверній частині реалізовано базові механізми безпеки. JWT-токени перевіряються через middleware під час звернення до захищених маршрутів. Якщо токен відсутній або недійсний, користувач не отримує доступ до відповідного ресурсу.

Паролі користувачів не зберігаються у відкритому вигляді. Перед записом до бази даних вони хешуються за допомогою bcrypt [13]. Також на рівні серверних маршрутів виконується базова перевірка вхідних даних, що зменшує ризик помилок під час роботи із запитами.

Помилки обробляються як на клієнтському, так і на серверному рівні. На сервері вони передаються через middleware обробки помилок, а на клієнті можуть відображатися у вигляді повідомлень або toast-сповіщень.

3.2 Інтерфейс користувача (UI/UX дизайн)

Інтерфейс користувача є важливою складовою вебзастосунку, оскільки саме через нього користувач взаємодіє з основними функціями системи. У Chatico реалізовано адаптивний інтерфейс, який забезпечує доступ до чатів, повідомлень, пошуку користувачів, профілю та групових налаштувань.

Структура папок клієнтської частини

Структура frontend-частини застосунку показана на рисунку 3.3.

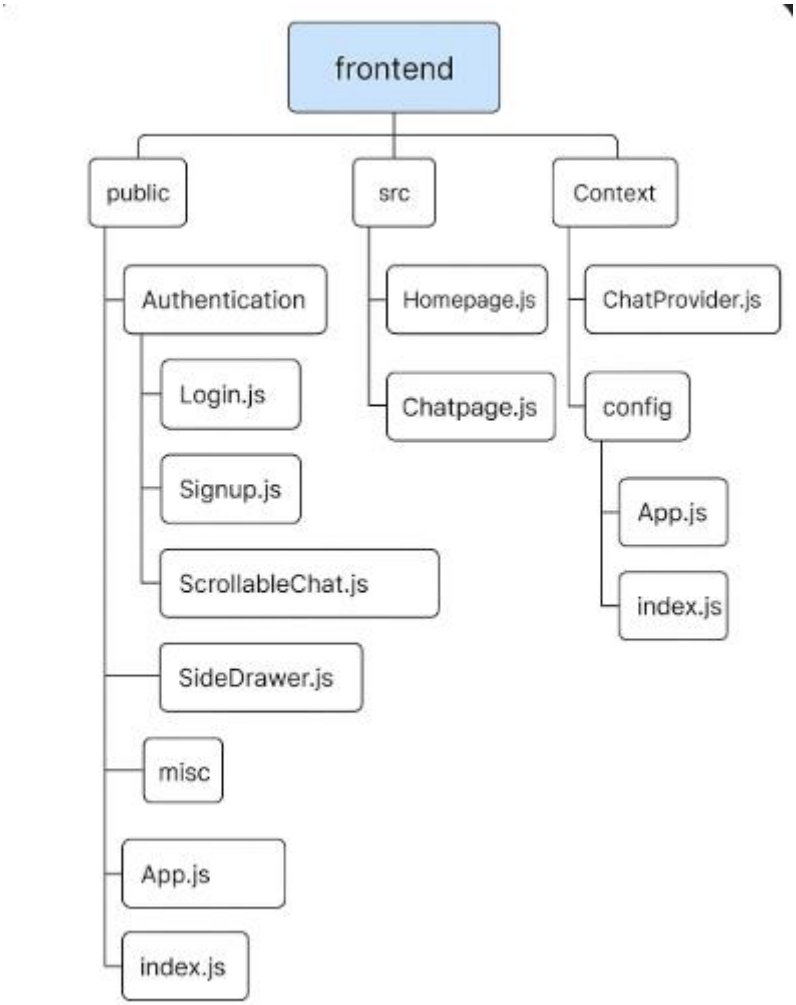

Рисунок 3.3. Структура файлів папки frontend

Клієнтська частина має таку загальну структуру:

Папка public/ містить статичні файли, зокрема HTML-шаблон сторінки. Основний код React-застосунку розташований у директорії src/. У папці components/ зберігаються повторно використовувані компоненти: форми входу та реєстрації, список чатів, вікно активного чату, модальні вікна, елементи профілю та компоненти для відображення користувачів.

Папка Pages/ містить основні сторінки застосунку: Homepage.js для авторизації та Chatpage.js для роботи з чатами. У папці Context/ розміщено ChatProvider.js, який відповідає за глобальний стан застосунку, зокрема дані користувача, обраний чат і список повідомлень.

3.3 Логіка навігації фронтенду Chatico

Навігація (рис у фронтенд-частині реалізована за допомогою React Router. Основна логіка переходів залежить від стану автентифікації користувача. Якщо користувач не авторизований, він бачить сторінку входу або реєстрації. Якщо дані користувача вже збережені на клієнті, застосунок перенаправляє його на сторінку чатів.

Логіку переходів між основними екранами застосунку додатково подано у вигляді графа діалогів. Він демонструє послідовність взаємодії користувача із системою: від входу або реєстрації до переходу на головну сторінку, вибору приватного чи групового чату, пошуку користувачів, перегляду профілю та керування групою. Граф діалогів наведено на рисунку 3.1.

Рисунок 3.4. Граф діалогів навігації

На головній сторінці користувач може вибрати вкладку входу або реєстрації. Після успішної авторизації дані користувача зберігаються в localStorage, а користувач перенаправляється на сторінку /chats.

Приклад перевірки авторизації на клієнті:

```
const user = JSON.parse(localStorage.getItem("userInfo"));
if (user) {
  history.push("/chats");
}
```

Сторінка /chats містить основні елементи інтерфейсу: список доступних чатів, активне вікно листування, пошук користувачів і модальні вікна для роботи з профілем або групами.

3.4 Етапи розробки фронтенду

Розробка frontend-частини Chatico виконувалася поступово. Спочатку було створено React-проєкт і встановлено необхідні залежності: Chakra UI, React Router, Axios та інструменти для роботи зі станом застосунку. Після цього було сформовано структуру папок і розділено код на сторінки, компоненти та контекст.

Реєстрація та вхід

Для реєстрації та входу користувача створено компоненти Signup.js і Login.js. Вони містять форми введення даних, базову валідацію та відправлення запитів до серверного API.

Приклад API-запиту для реєстрації користувача:

POST /api/user/register

Тіло запиту:

```
{
  "name": "Іван",
  "email": "ivan@example.com",
  "password": "secure123"
}
```

Приклад відповіді сервера:

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "_id": "665f4ab2...",
    "name": "Іван",
    "email": "ivan@example.com"
  }
}
```

Після успішного входу JWT-токен і дані користувача зберігаються на клієнті, що дозволяє підтримувати стан авторизації між оновленнями сторінки.

Створення чатів

Компонент MyChats.js відповідає за відображення доступних чатів користувача. Компонент SideDrawer.js використовується для пошуку інших користувачів і створення нового приватного чату. Для роботи з груповими чатами застосовуються модальні вікна створення та редагування групи.

Обмін повідомленнями

Основна логіка листування реалізована в компонентах ChatBox.js, SingleChat.js і ScrollableChat.js. Коли користувач обирає чат, клієнт отримує історію повідомлень із сервера. Нові повідомлення надсилаються через Socket.io та одразу відображаються в інтерфейсі.

Також реалізовано індикатор набору тексту. Під час введення повідомлення клієнт надсилає подію typing, а після завершення введення - stop typing.

Використання Chakra UI

Для побудови інтерфейсу використано Chakra UI. Ця бібліотека надає готові компоненти для React, зокрема кнопки, поля введення, модальні вікна, списки, меню та повідомлення. Chakra UI також підтримує адаптивність і доступність компонентів, що полегшує створення зручного інтерфейсу [8].

Структура інтерфейсу

Основними сторінками застосунку є сторінка авторизації та сторінка чатів.

На сторінці авторизації користувач може виконати вхід або створити новий обліковий запис.

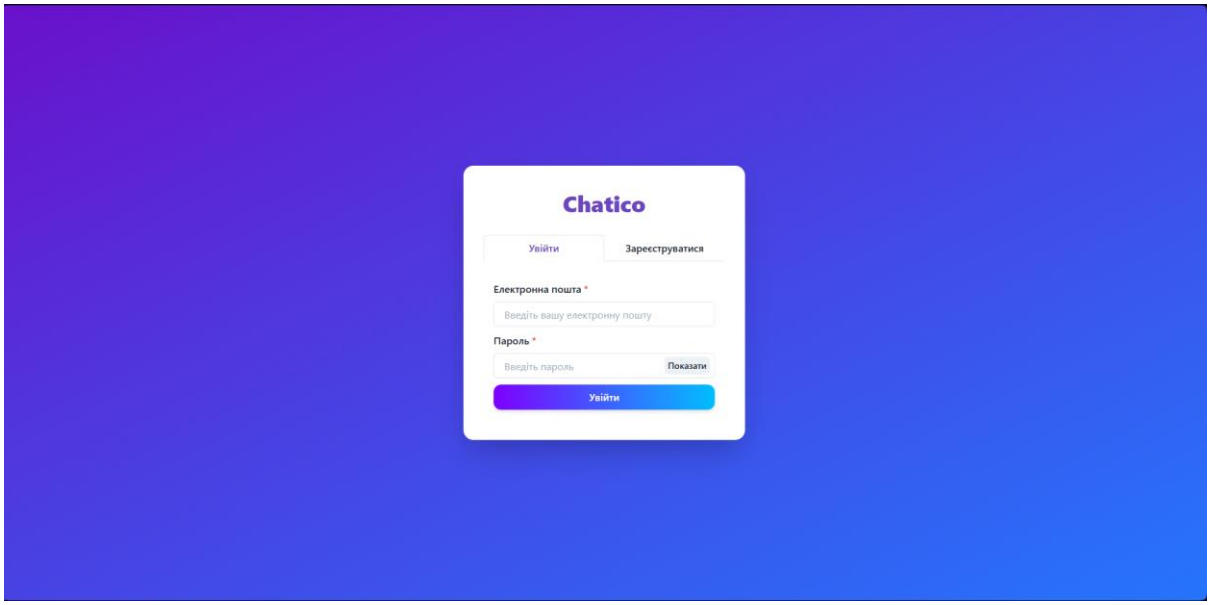

Рисунок 3.5. Скріншот форми входу

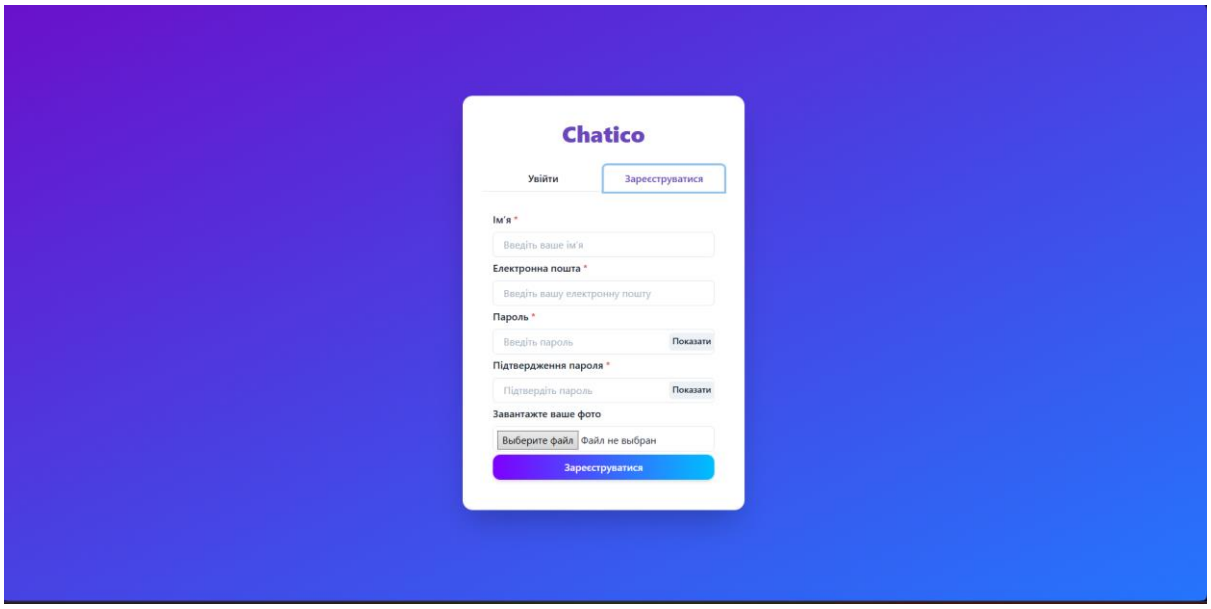

Рисунок 3.6. Скріншот форми реєстрації

На сторінці чатів розміщено такі елементи:

- ліва панель зі списком приватних і групових чатів;
- верхня панель із пошуком користувачів і кнопкою створення групи;
- основна область з історією повідомлень;
- поле введення тексту;
- індикатор набору повідомлення;
- модальні вікна профілю та керування групою.

Усі основні дії користувача виконуються без перезавантаження сторінки. Це стосується створення чатів, надсилання повідомлень, перегляду профілю, пошуку користувачів і роботи з групами.

Адаптивність і стиль

Інтерфейс Chatico адаптований до різних розмірів екрана. На великих екранах користувач бачить список чатів і активне вікно листування одночасно. На менших екранах елементи можуть перебудовуватися для зручнішого перегляду.

Основна палітра інтерфейсу складається із фіолетових і білих відтінків. Візуальний акцент зроблено на активному чаті, повідомленнях і діях користувача. Для тексту використано шрифти без засічок, які добре читаються на екранах різного розміру.

3.5 Тестування та результати

Для перевірки якості та стабільності застосунку Chatico було проведено тестування основних функцій системи. Перевірялися реєстрація, вхід, створення чатів, надсилання повідомлень, робота Socket.io, адаптивність інтерфейсу та обробка помилок.

Модульне тестування

Модульне тестування проводилося для окремих функцій backend-частини. Зокрема, перевірялися такі сценарії:

- реєстрація користувача;
- вхід користувача;
- перевірка JWT-токена;
- створення чату;
- надсилання та збереження повідомлення.

Для перевірки HTTP-запитів використовувався Postman. Офіційна документація Postman описує його як інструмент для роботи з API-проєктами, зокрема для надсилання запитів, перевірки відповідей і тестування API [20]. Для

тестування окремих логічних функцій може використовуватися Jest - JavaScript-фреймворк для перевірки коректності роботи коду [21].

Інтеграційне тестування

Під час інтеграційного тестування перевірялася взаємодія між клієнтом, сервером і базою даних. Основну увагу було приділено таким сценаріям:

- реєстрація користувача та запис даних у MongoDB;
- вхід користувача та отримання JWT-токена;
- створення приватного або групового чату;
- надсилання повідомлення через Socket.io;
- збереження повідомлення в базі даних;
- відображення нового повідомлення на клієнті.

Також перевірялося, чи коректно оновлюється стан інтерфейсу після подій, отриманих із сервера.

Тестування інтерфейсу

Користувацький інтерфейс тестувався вручну. Було перевірено валідацію форм входу й реєстрації, коректність відображення повідомлень, роботу модальних вікон, індикатор набору тексту та сповіщення про нові повідомлення.

Окремо перевірялася адаптивність інтерфейсу на різних типах пристроїв: персональному комп’ютері, планшеті та смартфоні. За результатами перевірки основні елементи інтерфейсу залишаються доступними та зручними для використання.

Навантажувальне тестування

Для перевірки роботи в умовах одночасного використання було імітовано роботу кількох користувачів у чаті. Тестування виконувалося через різні вкладки браузера та окремі облікові записи.

Перевірялися такі дії:

- одночасне надсилання повідомлень у чат;
- створення та перемикання між кількома чатами;
- робота індикатора набору тексту;
- отримання повідомлень у реальному часі;

- стабільність Socket.io-з’єднання.

Результати тестування наведено в таблиці 3.1.

Таблиця 3.1

Результати тестування застосунку Chatico

Тип тесту	**Результат**
Реєстрація та вхід	Успішно, з обробкою помилок
Відправлення повідомлень	Повідомлення надходять у реальному часі
З’єднання Socket.io	Стабільне в типових умовах
Адаптивність UI	Інтерфейс коректно відображається на різних екранах
Групові чати	Створення та керування групами працює коректно
Обробка винятків	Реалізована на клієнті та сервері

За результатами тестування можна зробити висновок, що застосунок Chatico стабільно працює в основних сценаріях використання та виконує ключові функціональні вимоги.

3.6 Охорона праці та безпека в ІТ-проєктах

Під час розробки програмного забезпечення важливо враховувати не лише функціональність застосунку, а й питання безпеки, захисту даних та організації робочого процесу. Для вебзастосунків, які працюють із персональними даними та повідомленнями користувачів, безпека є однією з основних вимог.

У межах дипломного проєкту Chatico було реалізовано базовий рівень інформаційної безпеки, достатній для навчально-практичного застосунку. Він охоплює захист облікових записів, контроль доступу до чатів, хешування паролів, перевірку токенів і обмеження доступу до серверних маршрутів.

Крім захисту даних, під час розробки також враховано інформаційну гігієну розробника: використання системи контролю версій, зберігання конфіденційних ключів у змінних середовища та ізоляцію локального середовища розробки.

3.7 Безпека даних у процесі розробки

У Chatico особливу увагу приділено захисту особистих даних користувачів і запобіганню несанкціонованому доступу до системи.

Основні заходи безпеки:

1. Хешування паролів

Паролі користувачів зберігаються у захешованому вигляді за допомогою bcrypt. Це означає, що навіть у випадку витоку бази даних паролі не будуть доступні у відкритому вигляді.

2. Автентифікація через JWT

Після входу користувач отримує токен, який використовується для доступу до захищених API-маршрутів. Сервер перевіряє токен перед виконанням дій, пов’язаних із чатами або повідомленнями.

3. Контроль доступу

У серверній частині реалізовано middleware, який перевіряє, чи має користувач право доступу до певного ресурсу. Наприклад, користувач не може переглядати чат, учасником якого він не є.

4. Захист API

Вхідні дані перевіряються на коректність перед обробкою. Це знижує ризик помилок і допомагає запобігти типовим атакам на вебзастосунки. У рекомендаціях OWASP звертається увага на необхідність перевірки даних, контролю доступу та захисту автентифікації у вебсистемах [22].

3.8 Інформаційна гігієна під час розробки

Під час роботи над проєктом було дотримано базових принципів безпечної розробки. Розробка виконувалася в локальному середовищі, що дозволяло тестувати функції без відкритого доступу ззовні.

Для контролю змін використовувалася система Git. Вона дозволяє відстежувати історію змін, повертатися до попередніх версій коду та зменшує ризик втрати результатів роботи.

Конфіденційні дані, такі як ключі доступу, токени та параметри підключення до бази даних, не повинні зберігатися безпосередньо в коді. Для цього використовуються змінні середовища у файлі .env. Такий підхід спрощує налаштування застосунку в різних середовищах і зменшує ризик випадкового розкриття секретних даних.

3.9 Технічна безпека під час експлуатації застосунку

Під час експлуатації Chatico користувачі захищені від базових ризиків, пов’язаних із несанкціонованим доступом до чатів і повідомлень. Клієнтська частина не має прямого доступу до бази даних, а всі операції виконуються через серверне API.

У системі передбачено такі обмеження:

- користувач не може переглядати повідомлення без авторизації;
- користувач не може отримати доступ до чатів, учасником яких він не є;
- усі запити до захищених маршрутів перевіряються на сервері;
- паролі не передаються й не зберігаються у відкритому вигляді;
- база даних недоступна напряму з клієнтської частини.

При розгортанні застосунку в публічному середовищі рекомендується використовувати HTTPS-з’єднання. Протокол HTTPS забезпечує захищене передавання даних між клієнтом і сервером за допомогою TLS, що особливо важливо для систем, які обробляють облікові дані та приватні повідомлення користувачів [23].

Також важливо регулярно оновлювати залежності проєкту, оскільки застарілі пакети можуть містити відомі вразливості. У подальшому для підвищення рівня безпеки можна додати обмеження кількості запитів, журналювання підозрілих дій, перевірку складності паролів і розширену валідацію введених даних.

Висновки до розділу 3

У третьому розділі описано практичну реалізацію програмного забезпечення Chatico. На цьому етапі було показано, як теоретичні положення та вимоги, визначені в попередніх розділах, реалізуються у вигляді робочого вебзастосунку для обміну повідомленнями в реальному часі.

Архітектура системи побудована за клієнт-серверним принципом. Клієнтська частина відповідає за відображення інтерфейсу та обробку дій користувача, серверна частина - за бізнес-логіку, авторизацію, обробку запитів і роботу з повідомленнями, а база даних - за зберігання інформації про користувачів, чати та повідомлення. Такий поділ робить систему зрозумілішою для розробки та подальшої підтримки.

Клієнтська частина реалізована з використанням React.js і Chakra UI. React дозволяє будувати інтерфейс із незалежних компонентів, а Chakra UI спрощує створення адаптивного та візуально узгодженого дизайну. У застосунку передбачено сторінки входу, реєстрації, чатів, а також компоненти для списку чатів, активного листування, профілю користувача та групових налаштувань.

Серверна частина побудована на Node.js та Express.js. Вона реалізує REST API, обробляє запити від клієнта, перевіряє JWT-токени, працює з базою даних через Mongoose та забезпечує логіку створення чатів і повідомлень. Використання JavaScript на клієнті й сервері спрощує розробку, оскільки весь проєкт побудований в одному технологічному середовищі.

Ключовим елементом застосунку є Socket.IO. Саме ця бібліотека забезпечує двосторонню комунікацію між клієнтом і сервером, завдяки чому повідомлення передаються без перезавантаження сторінки. Крім самих повідомлень, через Socket.IO можуть передаватися службові події, наприклад індикатор набору тексту або оновлення стану чату.

Для зберігання даних використовується MongoDB. Структура бази даних складається з трьох основних сутностей: користувачів, чатів і повідомлень. Модель користувача зберігає ім’я, електронну пошту, пароль у захешованому

вигляді, аватар і дату створення. Модель чату містить інформацію про учасників, тип чату, адміністратора групи та останнє повідомлення. Модель повідомлення зберігає автора, текст, пов’язаний чат і час створення.

У розділі також описано реалізацію інтерфейсу, навігації, етапів розробки frontend-частини та механізмів безпеки. У системі передбачено перевірку токенів, хешування паролів, обробку помилок, обмеження доступу до захищених маршрутів і базові заходи інформаційної гігієни під час розробки.

У третьому розділі підтверджено практичну можливість реалізації вебзастосунку Chatico на основі MERN-стека та Socket.IO. Розроблена система підтримує основні сценарії роботи месенджера: реєстрацію користувачів, вхід до системи, створення приватних і групових чатів, обмін повідомленнями в реальному часі, пошук користувачів, перегляд профілів і базові сповіщення. Це дозволяє розглядати Chatico як основу для подальшого розвитку повноцінної системи онлайн-комунікації.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено вебзастосунок Chatico, призначений для обміну повідомленнями між користувачами в режимі реального часу. У процесі роботи було реалізовано основні функції чат-платформи: реєстрацію та автентифікацію користувачів, створення приватних і групових чатів, надсилання повідомлень без перезавантаження сторінки, пошук користувачів, перегляд профілю, індикатор набору тексту та базові сповіщення.

Поставлену мету дипломної роботи було досягнуто. Результатом проєкту став функціональний вебзастосунок, який забезпечує зручну комунікацію між користувачами та демонструє практичне використання сучасних вебтехнологій у повностековій розробці.

Під час реалізації клієнтської частини було використано React.js і Chakra UI. Це дозволило створити адаптивний інтерфейс, який коректно відображається на різних пристроях і забезпечує зрозумілу взаємодію користувача із системою. Серверна частина була побудована на основі Node.js та Express.js, а для зберігання даних використано MongoDB. Обмін повідомленнями в реальному часі реалізовано за допомогою Socket.io, що дозволило передавати повідомлення між користувачами без потреби оновлювати сторінку.

Окрему увагу було приділено питанням безпеки. У застосунку реалізовано автентифікацію користувачів за допомогою JWT, хешування паролів із використанням bcrypt, а також перевірку доступу до захищених маршрутів на сервері. Це дозволяє обмежити доступ до чатів лише для авторизованих користувачів і зменшити ризики, пов’язані зі зберіганням облікових даних.

У процесі тестування було перевірено основні сценарії роботи застосунку: реєстрацію, вхід, створення чатів, надсилання повідомлень, роботу групових бесід, сповіщення, індикатор набору тексту та адаптивність інтерфейсу. За результатами перевірки система показала стабільну роботу в типових умовах використання.

Практичне значення розробленого застосунку полягає в тому, що Chatico може бути використаний як основа для подальшого створення системи онлайн-комунікації. Його можна адаптувати для внутрішніх чатів невеликих команд, освітніх платформ, сервісів підтримки користувачів або як базу для подальшої розробки мобільного застосунку.

Перспективними напрямами подальшого розвитку є додавання можливості надсилання файлів і зображень, реалізація реакцій на повідомлення, push-сповіщень, темної теми, розширеного пошуку по повідомленнях, а також створення мобільної версії за допомогою React Native або Flutter. Також у майбутньому можна покращити систему ролей у групових чатах і додати розширені механізми захисту даних.

Отже, дипломний проєкт Chatico демонструє повний цикл розробки вебзастосунку: від аналізу аналогів і формування вимог до проєктування архітектури, реалізації функціоналу, тестування та оцінки результатів. Виконання роботи дозволило закріпити практичні навички у сфері frontend- і backend-розробки, роботи з базами даних, організації обміну даними в реальному часі та забезпечення базової безпеки вебзастосунків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. DataReportal. Digital 2026: Ukraine. URL: https://datareportal.com/reports/digital-2026-ukraine (дата звернення: 07.05.2026).

2. MDN Web Docs. WebSocket API. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (дата звернення: 07.05.2026).

3. Socket.IO Documentation. Introduction. URL: https://socket.io/docs/v4/ (дата звернення: 07.05.2026).

4. React Documentation. React: Create user interfaces from components. URL: https://react.dev/ (дата звернення: 07.05.2026).

5. Node.js Documentation. About Node.js. URL: https://nodejs.org/en/about (дата звернення: 07.05.2026).

6. Express.js. Node.js web application framework. URL: https://expressjs.com/ (дата звернення: 07.05.2026).

7. MongoDB Documentation. Databases and Collections. URL: https://www.mongodb.com/docs/manual/core/databases-and-collections/ (дата звернення: 07.05.2026).

8. Chakra UI Documentation. Chakra UI - Accessible React components. URL: https://chakra-ui.com/ (дата звернення: 07.05.2026).

9. Telegram. Telegram Messenger. URL: https://telegram.org/ (дата звернення: 07.05.2026).

10. Telegram Core. End-to-End Encryption, Secret Chats / MTProto. URL: https://core.telegram.org/api/end-to-end (дата звернення: 07.05.2026).

11. WhatsApp Help Center. About end-to-end encryption. URL: https://faq.whatsapp.com/820124435853543 (дата звернення: 07.05.2026).

12. Discord Support. Beginner’s Guide to Discord / Discord Server Setup Guide. URL: https://support.discord.com/hc/en-us/articles/360045138571-Beginner-s-Guide-to-Discord (дата звернення: 07.05.2026).

13. npm. bcrypt. URL: https://www.npmjs.com/package/bcrypt (дата звернення: 07.05.2026).

14. IETF. RFC 7519: JSON Web Token (JWT). URL: https://datatracker.ietf.org/doc/html/rfc7519 (дата звернення: 07.05.2026).

15. Socket.IO Documentation. Automatic reconnection. URL: https://socket.io/docs/v4/client-options/#reconnection (дата звернення: 07.05.2026).

16. MongoDB Documentation. Indexes / Query Performance. URL: https://www.mongodb.com/docs/manual/indexes/ (дата звернення: 07.05.2026).

17. MDN Web Docs. Progressive web apps. URL: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps (дата звернення: 07.05.2026).

18. React Router. Official Documentation. URL: https://reactrouter.com/ (дата звернення: 07.05.2026).

19. Mongoose. Mongoose ODM Documentation. URL: https://mongoosejs.com/docs/ (дата звернення: 07.05.2026).

20. Postman. Postman Documentation Overview. URL: https://learning.postman.com/docs/introduction/overview/ (дата звернення: 07.05.2026).

21. Jest. Jest Documentation. URL: https://jestjs.io/docs/getting-started (дата звернення: 07.05.2026).

22. OWASP. OWASP Top 10 Web Application Security Risks. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 07.05.2026).

23. MDN Web Docs. HTTPS / Transport Layer Security. URL: https://developer.mozilla.org/en-US/docs/Glossary/HTTPS (дата звернення: 07.05.2026).

24. Hamilpurkar S., Vishvanath A. G. Real-Time Chat Application Using MERN Stack and Socket.IO. International Journal of Advanced Research in Computer and Communication Engineering. 2026. Vol. 15, Issue 1. DOI: 10.17148/IJARCCE.2026.15174. URL: https://ijarcce.com/papers/real-time-chat-application-using-mern-stack-and-socket-io/ (дата звернення: 23.05.2026).

25. Suhail A., Gupta V., Singhal A., Kumar S., Manoj. MERN Stack Chat Application. International Journal of Research Publication and Reviews. 2024. Vol. 5, No. 11. P. 5229-5232. URL: https://ijrpr.com/uploads/V5ISSUE11/IJRPR35351.pdf (дата звернення: 23.05.2026).

26. Realtime Chat Application Using MERN Stack. International Journal of Creative Research Thoughts. 2026. Vol. 14, Issue 5. URL: https://www.ijcrt.org/papers/IJCRT2605115.pdf (дата звернення: 23.05.2026).

27. WebSocket Adoption and the Landscape of the Real-Time Web. Proceedings of the Web Conference 2021. ACM Digital Library. 2021. URL: https://dl.acm.org/doi/10.1145/3442381.3450063 (дата звернення: 23.05.2026).

28. Díaz-Gómez J. M. Protocols, Reactive Architectures, and Computing Paradigms for Real-Time Web Applications. Future Internet. 2026. Vol. 18, No. 5. Article 254. URL: https://www.mdpi.com/1999-5903/18/5/254 (дата звернення: 23.05.2026).

29. Dospinescu O. Mobile Co-Living System for Real-Time Communication and Coordination. Software. 2026. Vol. 6, No. 2. Article 28. URL: https://www.mdpi.com/2673-7116/6/2/28 (дата звернення: 23.05.2026).

30. Real-Time Chat Application with Web Socket for Network Efficiency. ResearchGate. 2025. URL: https://www.researchgate.net/publication/394388196_Real-Time_Chat_Application_with_Web_Socket_for_Network_Efficiency (дата звернення: 23.05.2026).

31. Shahid J., Hameed M. K., Javed I. T., Qureshi K. N., Ali M., Crespi N. A Comparative Study of Web Application Security Parameters: Current Trends and Future Directions. Applied Sciences. 2022. Vol. 12, No. 8. Article 4077. URL: https://www.mdpi.com/2076-3417/12/8/4077 (дата звернення: 23.05.2026).

32. Althunayyan M., Saxena N., Gope P. Evaluation of Black-Box Web Application Security Scanners in Detecting Injection Vulnerabilities. Electronics. 2022. Vol. 11, No. 13. Article 2049. URL: https://www.mdpi.com/2079-9292/11/13/2049 (дата звернення: 23.05.2026).

33. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., ... & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science.

34. Kriskun, I., & Bondarchuk, A. (2026). ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ УПРАВЛІННЯ ПРОЄКТАМИ В ГАЛУЗІ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ. Європейський науковий журнал Економічних та Фінансових інновацій, 1(19), 264-274.

35. Aydos M., Aldan Ç., Coşkun E., Soydan A. Security testing of web applications: A systematic mapping of the literature. Journal of King Saud University - Computer and Information Sciences. 2022. Vol. 34, No. 9. P. 6775-6792. URL: https://www.sciencedirect.com/science/article/pii/S131915782100269X (дата звернення: 23.05.2026).

36. Биков А. О. Вебзастосунок обміну повідомленнями з підтримкою технології Socket.IO. Збірник матеріалів Всеукраїнської конференції молодих учених “Інформаційні технології”. 2026. № 13. С. 3-6. URL: https://zcit.kubg.edu.ua/index.php/journal/article/view/61 (дата звернення: 23.05.2026

37. Машкіна, І. В., Абрамов, В. О., Носенко, Т. І., Іваніченко, Є. В., & Яскевич, В. О. (2024). Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп’ютерні науки для здобувачів вищої освіти денної форми навчання.