

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЧАТУ
SOFTWARE IMPLEMENTATION OF A WEB CHAT

Виконав(ла): студент(ка) 4 курсу, групи КНТ-151

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ЛЕВЧЕНКО Я.Д.

(ПРИЗВИЩЕ та ініціали)

Керівник ФЕДОРОНЧАК Т.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОРОТКИЙ О.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЛЕВЧЕНКА Ярослава Дмитровича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація вебчату.
Software implementation of a web chat
 керівник проєкту (роботи) к.т.н., доцент, ФЕДОРОНЧАК Тетяна Василівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)
 затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209
2. Строк подання студентом проєкту (роботи) 02 червня 2025 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Реалізація основних компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми.
5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	ФЕДОРОНЧАК Т.В., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання « 14 » квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)


(підпис)

Ярослав ЛЕВЧЕНКО

(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Тетяна ФЕДОРОНЧАК

(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
83 с., 37 рис., 7 табл., 3 дод., 19 джерел.

ВЕБЗАСТОСУНОК, ВЕБЧАТ, ІНТЕРАКТИВНІ СИСТЕМИ,
АДАПТИВНИЙ ІНТЕРФЕЙС, NODE.JS, REACT, MYSQL.

Об'єкт дослідження — програмне забезпечення для забезпечення комунікації користувачів у реальному часі.

Предмет дослідження — вебзастосунок для обміну миттєвими повідомленнями (вебчат), його архітектура та функціонал.

Мета — дослідити принципи побудови вебчатів та розробити програмне рішення для онлайн-комунікації з використанням сучасних технологій.

Матеріали, методи та технічні засоби: мова програмування JavaScript, бібліотека React, серверна платформа Node.js, WebSocket-технологія через Socket.IO, система керування базами даних MySQL, середовище розробки Visual Studio Code.

Результати. Розроблено вебзастосунок для онлайн-комунікації, який підтримує обмін текстовими повідомленнями в реальному часі, функції авторизації, персоналізації інтерфейсу та зберігання історії чатів у базі даних MySQL. Застосунок має адаптивний інтерфейс, що забезпечує комфортне використання як на комп'ютерах, так і на мобільних пристроях. Це дозволяє користувачам спілкуватися з будь-якого гаджета, незалежно від розміру екрана.

Висновки. Розроблене програмне рішення ілюструє впровадження сучасних технологій для організації комунікаційного процесу в інтернет-просторі. Вебчат відповідає вимогам щодо зручності, ефективності та безпеки, демонструючи високий рівень доступності завдяки адаптивному дизайну.

Галузь застосування — корпоративні сервіси, соціальні мережі, служби технічної підтримки, онлайн-консультації, освітні платформи.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 83 p., 37 pictur., 7 tables, 3 appendixes, 19 sources.

WEB APPLICATION, WEB CHAT, INTERACTIVE SYSTEMS, ADAPTIVE INTERFACE, NODE.JS, REACT, MYSQL.

The object of research is software for ensuring user communication in real time.

The subject of the study is a web application for instant messaging (webchat), its architecture and functionality.

The purpose is to study the principles of web chatting and develop a software solution for online communication using modern technologies.

Materials, methods and tools: JavaScript programming language, React library, Node.js server platform, WebSocket technology via Socket.IO, MySQL database management system, Visual Studio Code development environment.

Results. A web application for online communication has been developed that supports real-time text messaging, authorization functions, interface personalization, and chat history storage in a MySQL database. The application has a responsive interface that ensures comfortable use on both computers and mobile devices. This allows users to communicate from any gadget, regardless of the screen size.

Conclusions. The developed software solution illustrates implementation of modern technologies for organizing the communication process in the Internet space. Webchat meets the requirements for convenience, efficiency, and security, demonstrating a high level of accessibility due to its adaptive design.

The application area includes corporate services, social networks, technical support services, online consultations, and educational platforms.

ЗМІСТ

	С.
Перелік скорочень на умовних познач 8	8
Вступ 9	9
1 Аналіз предметної області 11	11
1.1 Вебчати як засіб комунікації в реальному часі 11	11
1.2 Об'єкт, предмет, мета та методи дослідження 11	11
1.3 Переваги та недоліки існуючих вебчатів та рішень для онлайн-комунікації 12	12
1.3.1 Аналіз месенджера Slack 13	13
1.3.2 Аналіз чат-платформи Discord.....14	14
1.3.3 Аналіз месенджера Telegram.....15	15
1.4 Висновки за розділом 1 17	17
2 Матеріали і методи 18	18
2.1 Вибір середовища розробки 18	18
2.2 Вибір шаблону архітектури проєктування 20	20
2.3 Вибір бази даних.....22	22
2.4 Вибір мови програмування.....24	24
2.5 Висновки за розділом 2 25	25
3 Опис програми 27	27
3.1 Функціональні можливості вебчату 27	27
3.2 Архітектура програмного забезпечення 28	28
3.3 Принцип та логіка роботи.....31	31
3.4 Структура бази даних.....36	36
3.5 Проєктування інтерфейсу вебчату.....39	39
3.6 Висновки за розділом 3 46	46
4 Експлуатація, тестування та експериментальне дослідження програми.....48	48
4.1 Призначення та умови використання вебчату.....48	48
4.2 Технічні характеристики вебчату 49	49
4.3 Ініціалізація, запуск і взаємодія з компонентами 50	50

4.3.1 Звернення до програми	51
4.3.2 Вхідні та вихідні дані	51
4.3.3 Обробка помилок і повідомлень у системі	52
4.4 Чек-лист тестування функціоналу	52
4.5 Висновки за розділом 4	54
Висновки	55
Перелік джерел посилання	56
Додаток А Технічне завдання	58
Додаток Б Текст програми	65
Додаток В Слайди презентації	74

ПЕРЕЛІК СКОРОЧЕНЬ НА УМОВНИХ ПОЗНАК

MySQL – My Structured Query Language;

NODE.JS – Node JavaScript;

VS – Visual Studio;

WC – Web Chat;

WEB APP – Web Application;

ПЗ – програмне забезпечення.

ВСТУП

На сьогоднішній день комунікація в режимі реального часу є незамінною частиною цифрового життя. З розвитком мережевих технологій користувачі очікують миттєвого обміну повідомленнями, незалежно від того, який пристрій чи платформу вони використовують. Традиційні методи комунікації, такі як електронна пошта та форуми, поступово втрачають свою актуальність через затримки в доставці повідомлень і обмежені можливості. У зв'язку з цим вебчати, які забезпечують миттєвий обмін інформацією, стають усе більш популярними серед розробників і кінцевих користувачів.

ІТ-технології надають можливість створювати ефективні, гнучкі та масштабовані онлайн-комунікаційні системи. Завдяки таким інструментам, як WebSocket, React і Node.js, розробники мають можливість реалізувати інтерактивні інтерфейси та забезпечити обмін повідомленнями в режимі реального часу з мінімальними затримками. Ці технології активно використовуються не лише в соціальних мережах, а й у технічній підтримці, електронній комерції, освітніх платформах тощо.

Мета розробки – дослідити принципи побудови вебчатів та розробити програмне рішення для онлайн-комунікації з використанням сучасних технологій.

Предмет дослідження – вебзастосунок для миттєвого обміну повідомленнями (вебчат), його архітектура та функціональні компоненти.

Об'єкт дослідження – програмне забезпечення для забезпечення комунікації користувачів у реальному часі.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати сучасний стан і тенденції розвитку технологій вебчатів;
- дослідити існуючі рішення, виявити їх сильні та слабкі сторони;
- розробити концепцію власного вебчату, враховуючи актуальні вимоги та потреби користувачів;

- реалізувати архітектуру веб-застосунку з використанням JavaScript, React, Node.js, WebSocket через Socket.IO і MySQL;
- протестувати й налаштувати розроблену систему, забезпечивши її стабільну роботу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Вебчати як засіб комунікації в реальному часі

У сучасному цифровому світі миттєва комунікація стала невід'ємною частиною багатьох онлайн-сервісів. Вебчати є інструментами, які забезпечують обмін повідомленнями між користувачами в реальному часі через мережу інтернет. Вони активно використовуються в таких сферах, як електронна комерція, онлайн-освіта, технічна підтримка, соціальні мережі, а також у корпоративному спілкуванні.

Головною перевагою вебчатів є їхня швидкість і зручність в спілкуванні. Користувачі можуть швидко обмінюватися текстовими повідомленнями, файлами, зображеннями та навіть голосовими повідомленнями, не потребуючи додаткового програмного забезпечення – достатньо лише мати доступ до браузера.

З точки зору користувацького досвіду вебчати сприяють покращенню взаємодії, підвищенню лояльності та зручності використання сервісів. Вони можуть бути інтегровані в будь-який вебзастосунок як окремий модуль або як ключова функція, що становить основу продукту.

У контексті стрімкого розвитку інформаційних технологій та зростання потреби в гнучких і надійних комунікаційних інструментах вебчати залишаються актуальними і динамічно розвиваються. Їхнє застосування охоплює як прості рішення для особистої комунікації, так і масштабовані платформи для корпоративного обміну повідомленнями або підтримки користувачів.

1.2 Об'єкт, предмет, мета та методи дослідження

Об'єктом дослідження є процес комунікації між користувачами в реальному часі в контексті вебсередовища. У сучасних умовах цифрової взаємодії цей процес має важливе значення, охоплюючи широкий спектр сфер –

від особистого спілкування до обслуговування клієнтів у бізнесі.

Предметом дослідження виступає вебзастосунок для миттєвого обміну повідомленнями (вебчат). У фокусі знаходяться його архітектура, функціональні можливості, принципи роботи, а також технології, використовувані для його реалізації.

Метою дослідження є вивчення принципів побудови вебчатів та розробка ефективного програмного рішення для онлайн-комунікації з використанням сучасних технологій. Особлива увага буде приділена створенню зручного, надійного та масштабованого інтерфейсу, що забезпечить можливість обміну повідомленнями в режимі реального часу.

Для досягнення поставленої мети були використані наступні методи дослідження:

- аналіз існуючих рішень у сфері вебчатів, їх функціональних можливостей, переваг і недоліків;
- проєктування архітектури клієнт-серверної взаємодії на основі WebSocket-з'єднання;
- моделювання поведінки системи при навантаженні та тестування комунікаційної стабільності;
- програмна реалізація вебзастосунку за допомогою сучасного стеку технологій.

1.3 Переваги та недоліки існуючих вебчатів та рішень для онлайн-комунікації

На ринку існує широкий вибір рішень, які реалізують функціональність вебчатів. Вони можуть бути умовно поділені на два основні типи: готові інтегровані сервіси та кастомні (індивідуально розроблені) рішення. Далі буде розглянуто найбільш популярні приклади кожного з цих типів.

1.3.1 Аналіз месенджера Slack

Slack – це корпоративний месенджер, орієнтований на командну роботу. Він дозволяє створювати канали, вести приватні та групові розмови, обмінюватись файлами, додавати інтеграції з іншими сервісами (Trello, GitHub, Google Drive тощо) [1].

Переваги:

- зручна структура каналів для командної роботи;
- потужна інтеграція з сервісами (Google Drive, Trello, GitHub);
- підтримка ботів та автоматизації: можна реалізувати оповіщення, відповіді та інші автоматизовані дії.

Недоліки:

- платна модель для повного функціоналу: безкоштовна версія обмежує історію повідомлень та кількість інтеграцій;
- не оптимізований для неформального спілкування: більше підходить для робочих цілей;
- немає можливості повноцінної модерації повідомлень: Slack не дозволяє вбудовано модерувати або автоматично фільтрувати контент (наприклад, заборонені слова).

На рисунку 1.1 продемонстровано месенджер Slack [1].

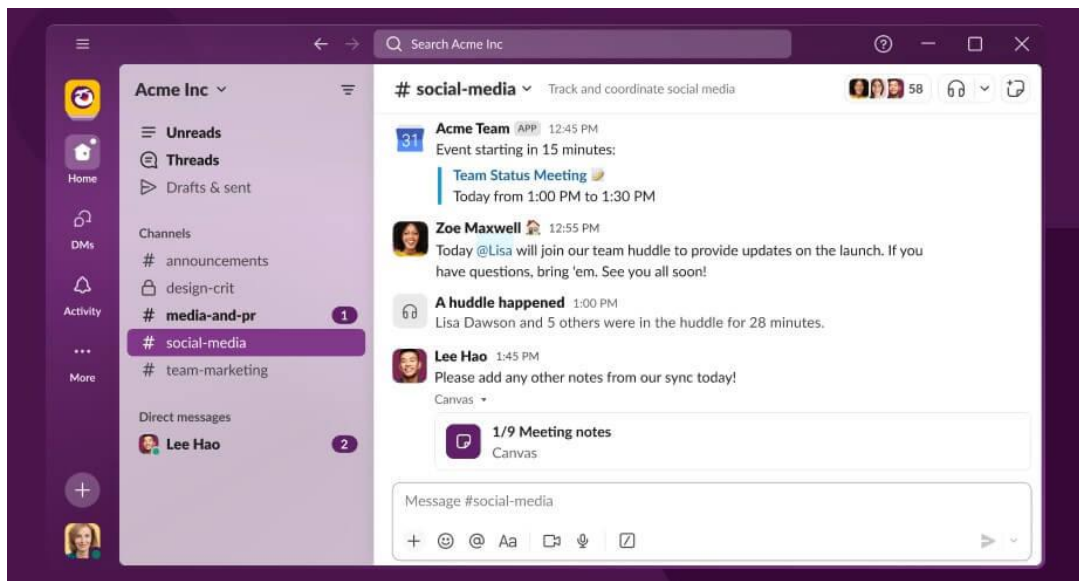


Рисунок 1.1 – Месенджер Slack [1]

1.3.2 Аналіз чат-платформи Discord

Discord – це чат-платформа, спочатку розроблена для геймерів, однак наразі широко використовується для різних типів онлайн-спільнот. Підтримує текстові, голосові та відеочати [2].

Переваги:

- підтримка голосових, відео та текстових каналів;
- велика кількість ботів для автоматизації та модерації;
- активна спільнота: Discord підтримує велику кількість людей в одній групі, з можливістю налаштування прав доступу і ролей для кожного учасника.

Недоліки:

- залежність від серверів Discord: всі повідомлення проходять через сервери Discord, що може створювати проблеми з конфіденційністю та безпекою;
- обмежена кастомізація: неможливо повністю змінити інтерфейс або логіку роботи чату. Усе обмежено стандартними функціями;
- затримки в повідомленнях: в години пік може бути затримка при доставці повідомлень, що може бути незручним для користувачів.

На рисунку 1.2 продемонстровано чат-платформу Discord [2].

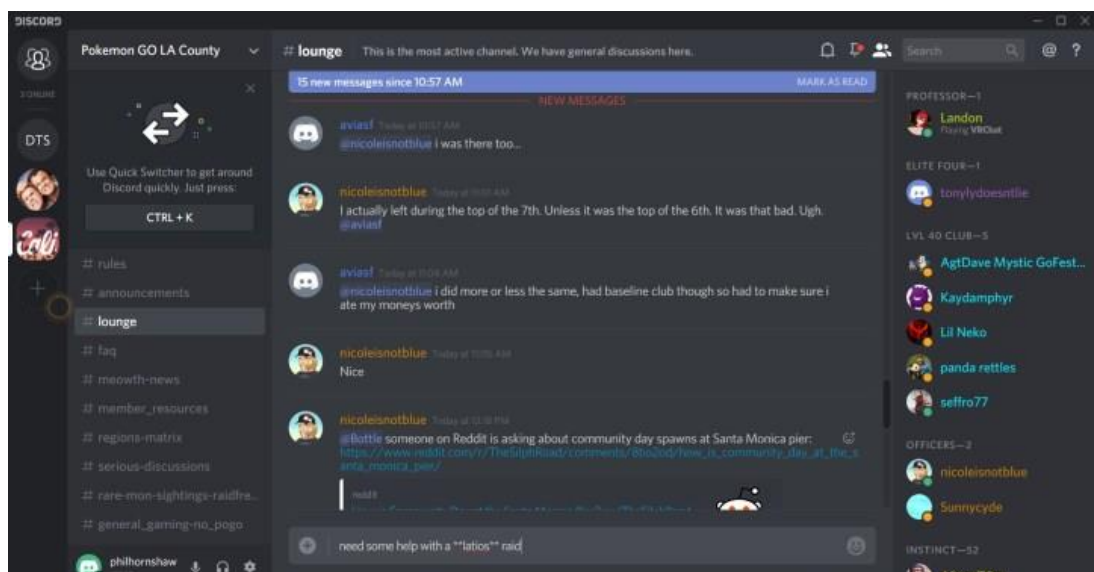


Рисунок 1.2 – Чат-платформа Discord [2]

1.3.3 Аналіз месенджера Telegram

Telegram – це месенджер, який забезпечує доступ до текстових чатів, медіа, каналів та ботів [3].

Переваги:

- шифрування та безпека: Telegram славиться високим рівнем захисту даних;
- боти та автоматизація: можливість створення ботів для обслуговування, FAQ, опитувань тощо;
- швидкість роботи: миттєва доставка повідомлень, легкий інтерфейс.

Недоліки:

- прив'язка до мобільного телефону для авторизації;
- відсутність структуризації (немає каналів за темами, як у Slack);
- неможливість зміни логіку під вебпроєкт.

На рисунку 1.3 продемонстровано месенджер Telegram [3].

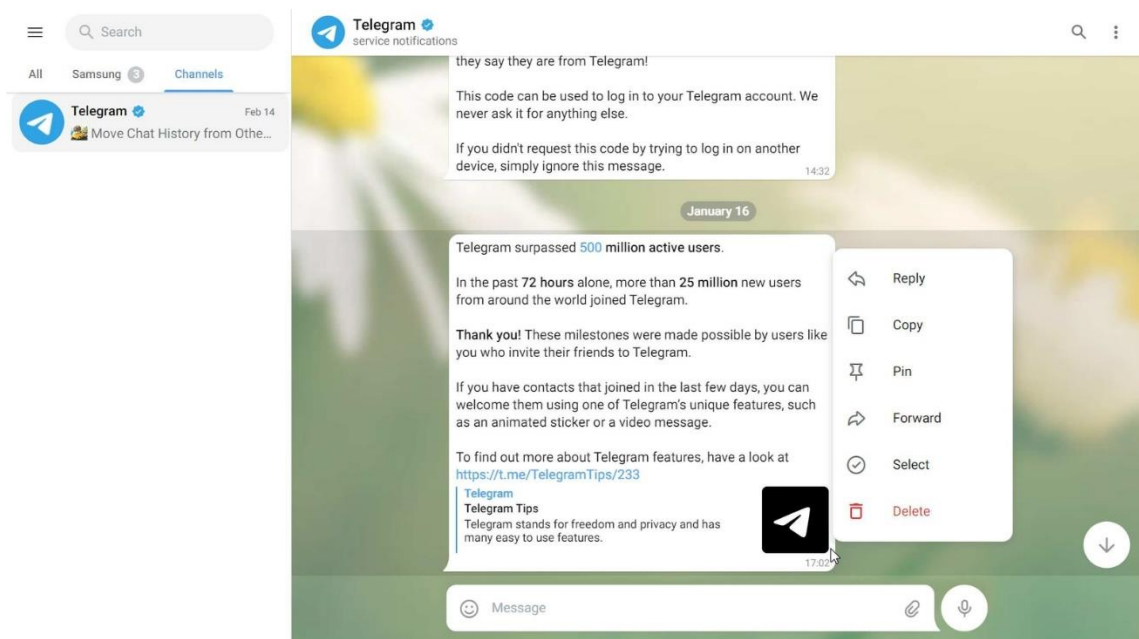


Рисунок 1.3 – Месенджер Telegram [3]

У ході виконання дипломної роботи буде розроблено вебчат для онлайн-комунікації, який дозволить користувачам обмінюватися миттєвими повідомленнями в реальному часі. Вебчат буде включати функціонал для створення чатів, надсилення повідомлень та інтеграцію з базою даних для зберігання історії повідомлень.

У таблиці 1.1 продемонстровано порівняння функціоналу чатів Slack, Discord, Telegram Web та розроблюваного вебчату.

Таблиця 1.1 – Порівняння функціоналу існуючих аналогів з розроблюваним вебчатом

Функціональність	Slack	Discord	Telegram	Розроблюючий вебчат
Кастомізація інтерфейсу	—	—	—	+
Безпека / шифрування	—	—	+	+
Підходить для бізнесу	+	—	—	+

У порівняльній таблиці 1.1 представлені основні функції чотирьох чат-систем: Slack, Discord, Telegram Web та розроблюваного вебчату. Зіставлення популярних чат-платформ із розроблюваним вебчатом демонструє його переваги в аспектах кастомізації, безпеки та орієнтації на бізнес-користувачів. На відміну від Slack, Discord і Telegram, вебчат повністю адаптується під потреби конкретного проекту, дозволяє реалізувати кастомну модерацію, налаштування ролей та шифрування повідомлень. У той час як існуючі платформи мають або закриту архітектуру, або обмеження щодо бізнес-функціональності, створений вебчат дозволяє поєднати необхідні можливості без надмірної складності й зайвого функціонального перевантаження.

1.4 Висновки за розділом 1

Перший розділ дипломної роботи присвячений вивченню вебчатів як важливого інструменту для комунікації в реальному часі. Вебчати стали основою для ефективної взаємодії в онлайн-середовищі, пропонуючи можливість миттєвого обміну повідомленнями та безперешкодного спілкування незалежно від географічних бар'єрів. У розділі детально розглядаються ключові особливості функціонування вебчатів, їх переваги та недоліки, а також формується об'єкт і предмет дослідження, мета та методи, які будуть використані в подальшій роботі.

У результаті аналізу поточних чат-платформ, таких як Slack, Discord і Telegram, були виділені їх сильні та слабкі сторони. Врахування цих аспектів дозволить під час розробки нового вебчату ефективно використовувати найкращі практики та уникати недоліків, які зустрічаються в існуючих рішеннях.

Загалом, перший розділ закладає теоретичну основу для подальших етапів створення вебчату, враховуючи широкий спектр функціональних і технічних вимог для створення зручного та ефективного інструменту для онлайн-комунікації.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір середовища розробки

Для розробки вебчату було обрано Visual Studio Code (VS Code) як основне середовище розробки. VS Code є легким, але потужним редактором коду, що підтримує безліч мов програмування, розширень і функцій для налагодження, тестування та інтеграції з різними інструментами [4].

Visual Studio Code забезпечує широкі можливості для веб-розробки завдяки підтримці таких мов, як JavaScript, TypeScript, HTML, CSS, а також інтеграції з популярними фреймворками та бібліотеками, наприклад, React, Angular, Node.js. Це середовище надає багатий набір плагінів, таких як розширення для роботи з Git, інструменти для дебагінгу, а також можливість працювати з Docker, що робить розробку зручнішою та ефективнішою.

Переваги Visual Studio Code:

- легкість і швидкість: це середовище має мінімальні вимоги до системних ресурсів, що забезпечує високу швидкість роботи, навіть при великих проєктах;
- багатофункціональність: вбудована підтримка налагодження коду, інструменти для роботи з Git, інтеграція з різноманітними базами даних та іншими сервісами;
- розширюваність: користувачі можуть додавати необхідні плагіни та розширення для роботи з різними мовами програмування, фреймворками та бібліотеками;
- активна підтримка спільноти: Visual Studio Code постійно оновлюється та має велику кількість доступних плагінів, що забезпечують інтеграцію з різноманітними інструментами для розробки.

Альтернативи Visual Studio Code:

Однак існують й інші середовища розробки, які можуть бути використані для подібних проєктів:

а) WebStorm – це інтегроване середовище розробки, спеціалізоване на JavaScript, Node.js, Angular, React, Vue.js та інших вебтехнологіях. WebStorm має потужні інструменти для автодоповнення коду, рефакторингу та налагодження [5];

б) Atom – легкий текстовий редактор, що підтримує великий набір плагінів та інтеграцій для веброзробки. Він добре підходить для проектів середнього розміру, але менш потужний в порівнянні з VS Code [6];

в) Sublime Text – також є легким редактором з підтримкою плагінів, що дозволяє ефективно працювати з кодом, але його можливості налагодження та інтеграції з іншими інструментами є обмеженими порівняно з VS Code [7].

У таблиці 2.1 продемонстровано порівняння альтернатив середовищ розробки з обраним середовищем.

Таблиця 2.1 – Порівняння середовищ розробки

Критерії	WebStorm	Atom	Sublime Text	VS Code
Легкість і швидкість	–	+	+	+
Інтеграція з фреймворками	+	+	–	+
Підтримка плагінів	+	+	–	+
Підтримка Git	+	–	+	+
Інтеграція з Docker	–	–	–	+

З огляду на порівняння середовищ розробки, Visual Studio Code є найкращим вибором для створення вебчату завдяки своїй легкості, швидкості, широкій підтримці мов програмування та фреймворків, а також безкоштовній доступності. Крім того, це середовище має вбудовані інструменти для дебагінгу, інтеграцію з Git та можливість роботи з Docker, що робить розробку більш ефективною і зручною.

WebStorm, хоча і є потужним середовищем для розробки, має платну

ліцензію, що може бути перешкодою для деяких розробників, особливо якщо потрібно використовувати його на великих проектах.

Atom і Sublime Text, хоча й мають хорошу підтримку плагінів і зручні для роботи з кодом, не пропонують такого рівня інтеграції з сучасними інструментами та фреймворками, як Visual Studio Code, і мають обмежені можливості для дебагінгу та роботи з великими проектами.

Таким чином, для проєкту вебчату Visual Studio Code є найкращим варіантом завдяки своїй потужності, доступності та гнучкості у роботі з різними інструментами.

2.2 Вибір шаблону архітектури проєктування

При розробці вебчату для реалізації структури проєкту було обрано архітектуру MVC (Model-View-Controller). Ця архітектура є однією з найпопулярніших для розробки вебзастосунків, оскільки вона чітко розділяє бізнес-логіку, інтерфейс користувача та управління запитами, що дозволяє зручніше організувати процес розробки та підтримки проєкту [8].

Опис архітектури MVC:

- Model: модель відповідає за логіку бізнес-процесів та роботу з даними;
- View: перегляд (View) відповідає за представлення даних користувачу;
- Controller: контролер відповідає за приймання запитів від користувача, їх обробку та виклик відповідних методів з моделі для отримання даних або зміни їх стану.

Альтернативи архітектури MVC:

а) MVVM (Model-View-ViewModel)

У даній архітектурі ViewModel є посередником між View та Model. Вона відповідає за управління станом інтерфейсу та взаємодією з даними.

MVVM більше підходить для додатків з інтерактивними інтерфейсами, де

необхідно часто оновлювати дані без перезавантаження сторінки [9];

б) Microservices Architecture

Розділення додатка на окремі мікросервіси, кожен з яких виконує певну функцію (наприклад, управління користувачами, чати, повідомлення тощо).

Мікросервіси зазвичай використовуються для великих і складних проєктів, де кожен компонент має свою автономну функціональність.

Може бути зайвим для невеликих або середніх вебдодатків, як вебчат [10];

в) Monolithic Architecture

У цій архітектурі весь додаток складається з одного великого коду, який включає всі функції (UI, логіка, база даних).

Це спрощує розробку на початкових етапах, але з часом може призвести до проблем з масштабованістю та підтримкою [11].

У таблиці 2.2 продемонстровано порівняння архітектур проєктування з обраною архітектурою.

Таблиця 2.2 – Порівняння архітектур проєктування

Критерії	MVVM	Microservices	Monolithic	MVC
Масштабованість	+	+	–	+
Простота реалізації	+	–	+	+
Гнучкість	+	+	–	+
Інтерактивність інтерфейсу	+	+	–	+
Використання ресурсів	+	–	–	+

З огляду на порівняння архітектур, MVC є оптимальним вибором для розробки вебчату, оскільки вона забезпечує чітке розділення між моделлю, виглядом і контролером, що спрощує розробку та підтримку коду. Вона добре підходить для проєктів середнього розміру, де потрібно підтримувати логіку бізнесу та користувацький інтерфейс окремо, з мінімальними вимогами до

масштабованості.

MVVM, хоча і забезпечує високу інтерактивність та зручність тестування, може бути більш складною для реалізації, що потребує додаткових ресурсів. Це може бути корисним для більш складних вебдодатків, але для вебчату, який не потребує надмірної складності UI, MVC буде зручнішим.

Microservices – потужна архітектура для великих проєктів з високими вимогами до масштабованості та розподіленості, але для простого вебчату ця архітектура може бути занадто складною і вимагати додаткових ресурсів на підтримку та інтеграцію.

Monolithic архітектура є простим рішенням, яке підходить для невеликих проєктів, але з часом може мати обмеження щодо масштабованості та гнучкості.

Таким чином, для вебчату, враховуючи вимоги до простоти реалізації та підтримки, MVC є найкращим варіантом, оскільки вона забезпечує хорошу структуру коду, зручність в роботі з базою даних і адаптується до вимог проєкту.

2.3 Вибір бази даних

Для збереження даних у вебчаті було обрано MySQL – реляційну систему управління базами даних (СУБД), що відзначається надійністю, високою продуктивністю та широкою підтримкою з боку спільноти. MySQL є відкритим ПЗ, активно використовується у веброзробці та добре інтегрується з серверною частиною.

MySQL дозволяє ефективно працювати з табличними структурами, зручно реалізовує зв'язки між сутностями, має вбудовані механізми транзакцій, реплікації та резервного копіювання, що особливо важливо для надійної роботи вебдодатку [12].

Альтернативи MySQL:

а) PostgreSQL – потужна об'єктно-реляційна СУБД з відкритим кодом. Має ширший функціонал у порівнянні з MySQL, зокрема підтримку збережених процедур, розширену роботу з JSON та складні типи даних. Часто

використовується там, де потрібна точна відповідність ACID-принципам або складна логіка на рівні бази [13];

б) MongoDB – документно-орієнтована NoSQL-база даних, що краще підходить для гнучких, нереляційних структур. Добре масштабується, особливо під великі обсяги даних, але не гарантує таких строгих зв'язків між сутностями, як у реляційних БД [14];

в) SQLite – легка, вбудована БД, яка не потребує окремого сервера. Підходить для невеликих додатків або прототипів. Має обмежену масштабованість і меншу продуктивність при високих навантаженнях [15].

У таблиці 2.3 продемонстровано порівняння баз даних з обраною базою.

Таблиця 2.3 – Порівняння баз даних

Критерії	PostgreSQL	MongoDB	SQLite	MySQL
Реляційна модель	+	–	+	+
Підтримка ACID	+	–	+	+
Швидкість запису	–	+	–	+
Простота використання	–	+	+	+

З огляду на порівняння систем керування базами даних, MySQL є найкращим вибором для реалізації вебчату завдяки своїй простоті у використанні, високій продуктивності та широкій підтримці у спільноті. Це рішення добре масштабується, має зрозумілий синтаксис SQL та чудово інтегрується.

PostgreSQL, хоча і забезпечує більш гнучку роботу з типами даних і складними запитамі, може виявитися надлишковим для вебчату, де потрібна насамперед швидкість і надійність при роботі з простими таблицями.

MongoDB пропонує гнучкість у структурі даних і зручність у роботі з JSON-форматом, але не забезпечує реляційні зв'язки, що можуть бути критично важливими при моделюванні логіки користувачів і чатів.

SQLite підходить переважно для невеликих локальних застосунків, однак

у масштабованих мережових рішеннях, як вебчат, її можливостей буде недостатньо.

Таким чином, MySQL – це найоптимальніший варіант для даного проєкту завдяки балансу між функціональністю, стабільністю та зручністю у використанні.

2.4 Вибір мови програмування

Для створення вебчату було обрано JavaScript як основну мову програмування, разом з фреймворками React для фронтенду та Node.js для серверної частини. JavaScript є однією з найпопулярніших мов для розробки веб і дозволяє реалізувати як клієнтську, так і серверну логіку в межах однієї мови.

Поєднання React і Node.js забезпечує високу швидкість розробки, зручну модульну структуру коду, а також ефективну взаємодію між інтерфейсом користувача і серверною частиною. Це рішення дозволяє створювати масштабовані, динамічні інтерфейси й ефективно працювати з асинхронними запитами [16].

Альтернативи JavaScript:

а) Python – сучасна мова з простим синтаксисом і великим набором бібліотек. Часто використовується для серверної частини (наприклад, Flask або Django), але менш придатна для клієнтської веброботи [17];

б) PHP – класична мова для розробки динамічних вебсайтів. Добре працює з серверною логікою, однак має менш сучасний підхід у порівнянні з JavaScript-орієнтованими фреймворками [18];

в) C# – використовується у зв'язці з ASP.NET, добре підходить для корпоративних рішень, але вимагає більш складної інфраструктури та має вищий поріг входу [19].

У таблиці 2.4 продемонстровано порівняння мов програмування з обраною мовою.

Таблиця 2.4 – Порівняння мов програмування

Критерії	Python	PHP	C#	JavaScript
Робота на клієнті	–	–	–	+
Простота вивчення	+	+	–	+
Швидкість розробки	+	–	–	+
Підтримка сучасних UI	–	–	+	+

З огляду на порівняння мов програмування, JavaScript у поєднанні з фреймворками React і Node.js є найкращим вибором для розробки вебчату. Він дозволяє реалізовувати як клієнтську, так і серверну логіку в межах однієї мови, що значно спрощує розробку, супровід і масштабування застосунку.

Альтернативи, такі як Python, PHP чи C#, також придатні для створення вебдодатків, проте вони або мають слабшу підтримку клієнтської частини (Python, PHP), або потребують складнішого стеку та мають вищий поріг входу (C#). Таким чином, обраний стек JavaScript (React + Node.js) є найбільш гнучким, сучасним та ефективним рішенням для реалізації функціонального, масштабованого та інтерактивного вебчату.

2.5 Висновки за розділом 2

У другому розділі було здійснено обґрунтований вибір ключових технологій, необхідних для реалізації вебчату. Середовище розробки Visual Studio Code обране завдяки своїй гнучкості, широкій підтримці розширень, інтеграції з системами контролю версій і зручності роботи з JavaScript/Node.js-проектами.

Як архітектурний шаблон було обрано MVC (Model–View–Controller), оскільки він чітко розділяє логіку даних, відображення та управління, що полегшує розробку, тестування та підтримку проєкту.

Для збереження даних використано MySQL – потужну реляційну СУБД, яка відзначається продуктивністю, стабільністю та зручністю при роботі зі

структурованими даними. Вона чудово підходить для реалізації чату зі зв'язками між користувачами, повідомленнями та чат-кімнатами.

Основною мовою програмування обрано JavaScript у поєднанні з React (для клієнта) та Node.js (для сервера). Такий стек дозволяє ефективно реалізовувати вебчат з високою швидкістю роботи, гнучкою логікою та сучасним користувацьким інтерфейсом.

Таким чином, усі обрані інструменти, технології та архітектурні рішення є оптимальними для створення функціонального, масштабованого та надійного вебчату.

3 ОПИС ПРОГРАМИ

3.1 Функціональні можливості вебчату

Створений вебчат буде забезпечувати ефективну та зручну комунікацію між користувачами у реальному часі. Основні функціональні можливості охоплюють як базові, так і додаткові сервіси, що будуть сприяти покращенню користувацького досвіду:

а) реєстрація та авторизація користувачів

Користувачі матимуть змогу створювати облікові записи та входити до системи через форму логіну;

б) обмін текстовими повідомленнями в реальному часі

Завдяки WebSocket-з'єднанню повідомлення передаватимуться миттєво без потреби оновлення сторінки;

в) вивід історії повідомлень

При вході в чат користувач матиме змогу бачити хронологічну історію переписки, збережену в базі даних MySQL;

г) створення нових чатів

Користувачі зможуть створювати нові чати. При створенні чату буде можливість задати його назву, тип доступу та учасників;

д) пошук користувачів і повідомлень

Буде реалізований пошук за іменем користувача або ключовими словами в історії чату;

е) можливість видалення та редагування власних повідомлень

Користувачі зможуть змінювати або видаляти власні повідомлення протягом певного часу після надсилання;

ж) адаптивний дизайн

Інтерфейс вебчату автоматично підлаштується під різні типи пристроїв – десктопи, планшети, смартфони;

з) видалення чатів

Користувачі зможуть видаляти створені ними чати, що дозволить

очищати список чатів та уникати зайвої інформації;

и) редагування чатів

Дозволяє змінювати параметри створених чатів, такі як назва, опис, або склад учасників;

к) сторінка налаштувань користувача

Окрема сторінка для редагування особистої інформації користувача: ім'я, фото профілю, пароль, та інші дані;

л) відправка файлів

Можливість надсилати медіа- та інші файли в чатах, що розширює функціонал обміну даними.

3.2 Архітектура програмного забезпечення

У процесі розробки вебчату важливо чітко визначити логіку взаємодії між клієнтською та серверною частинами. Це дозволяє оптимізувати обмін даними, забезпечити масштабованість системи та мінімізувати можливі помилки в обробці запитів. Одним із ключових етапів проєктування є побудова структурної схеми, яка зображена на рисунку 3.1. Структурна схема демонструє загальну архітектуру вебчату – від інтерфейсу користувача до внутрішніх обробників на сервері.

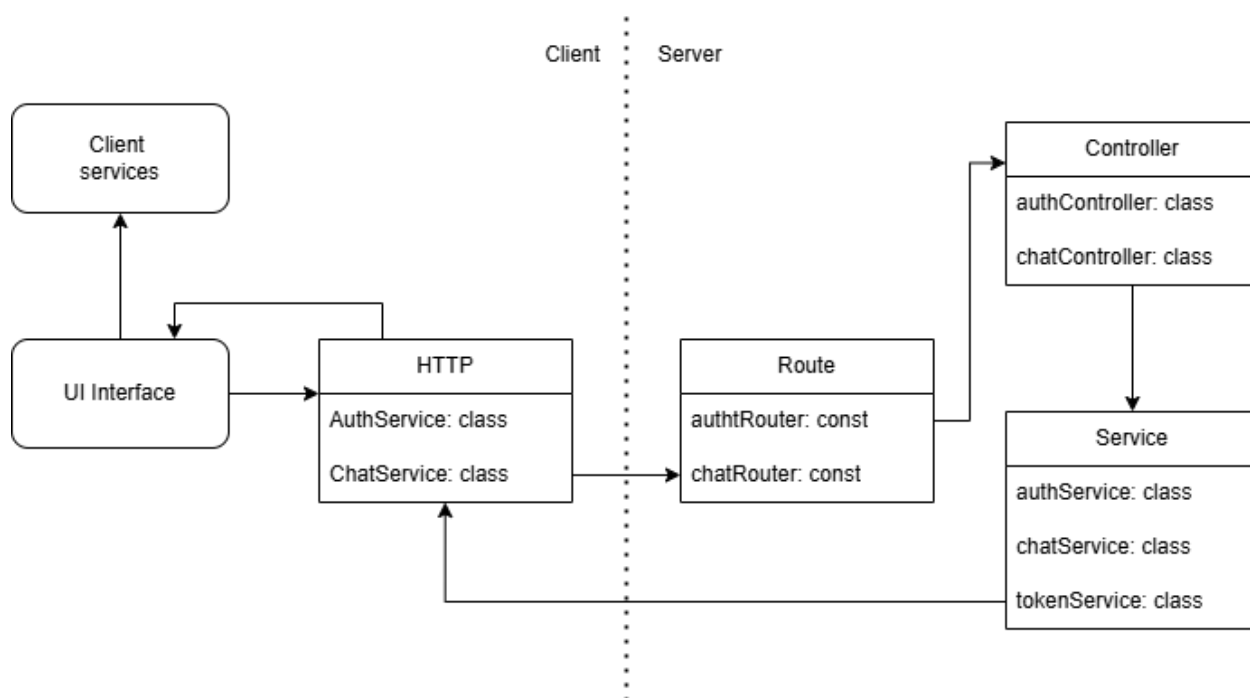


Рисунок 3.1 – Структурна схема вебчату, який розроблюється

На рисунку 3.1 представлено загальну схему обміну даними між клієнтською частиною та серверною частиною вебчату, а саме:

а) клієнтська частина (Client side):

1) UI Interface: Інтерфейс користувача, через який відбувається ініціація дій (вхід у чат, відправлення повідомлень, створення кімнати тощо);

2) Client Services: Містить функціональні компоненти, які готують HTTP-запити до сервера (наприклад, `fetch()`, `axios.post()`);

б) передача даних (Transport Layer):

1) HTTP: Класичний протокол обміну повідомленнями між клієнтом і сервером, що забезпечує надійний транспорт запиту та відповіді;

2) використовується для RESTful-запитів (GET, POST, PUT, DELETE);

в) серверна частина (Server side):

1) Route (Маршрутизатор): Визначає, який контролер обробить запит відповідно до URL та типу методу (наприклад, `POST /sendMessage`);

2) Controller (Контролер): Приймає вхідний запит, перевіряє його на валідність, викликає відповідний сервіс;

3) Service (Сервіс): Реалізує основну бізнес-логіку (збереження повідомлень у базі, перевірка прав доступу, обробка помилок);

г) зворотна відповідь до клієнта:

1) після обробки запиту, відповідь через той самий HTTP-канал повертається на клієнт;

2) дані оновлюються на інтерфейсі користувача (UI) автоматично або через зворотні виклики (callback, WebSocket, push-події).

Розроблюючи вебчат доцільно структурувати систему на окремі функціональні компоненти, що полегшує масштабування, супровід та тестування програмного забезпечення. Для цього було побудовано діаграму компонентів, яка демонструє ключові елементи системи та взаємозв'язки між ними (рис. 3.2).

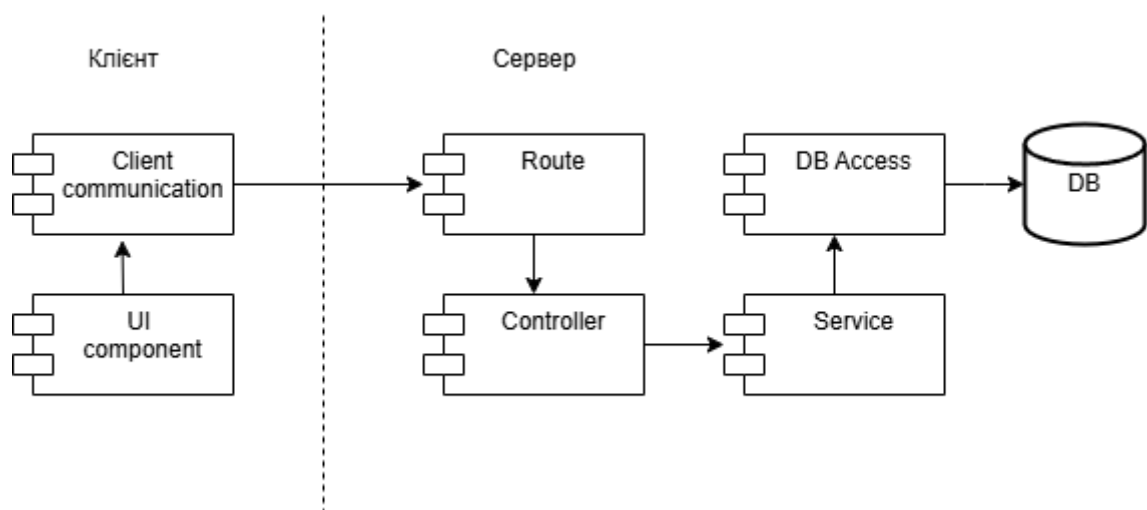


Рисунок 3.2 – Діаграма компонентів вебчату, який розроблюється

У таблиці 3.1 детально представлено основні компоненти системи.

Таблиця 3.1 – Основні компоненти системи

Компонент	Призначення
Client communication	Модуль клієнтської частини, що відповідає за надсилання запитів на сервер.
UI component	Інтерфейс користувача, через який здійснюється взаємодія з вебчатом.
Route	Відповідає за перенаправлення запитів до відповідних контролерів.
Controller	Приймає запити, викликає відповідні сервіси, обробляє помилки.
Service	Містить бізнес-логіку: обробка повідомлень, створення чатів тощо.
DB Access	Шар доступу до бази даних.
DB	Сховище даних – база, що містить інформацію про повідомлення.

3.3 Принцип та логіка роботи

На основі логічної структури системи (діаграми компонентів) розроблено схему функціонування вебчату, що відображає основні процеси обміну даними між користувачем (клієнтом), серверною логікою та базою даних. Ця схема представлена на рисунку 3.3 і демонструє типові сценарії роботи вебчату.

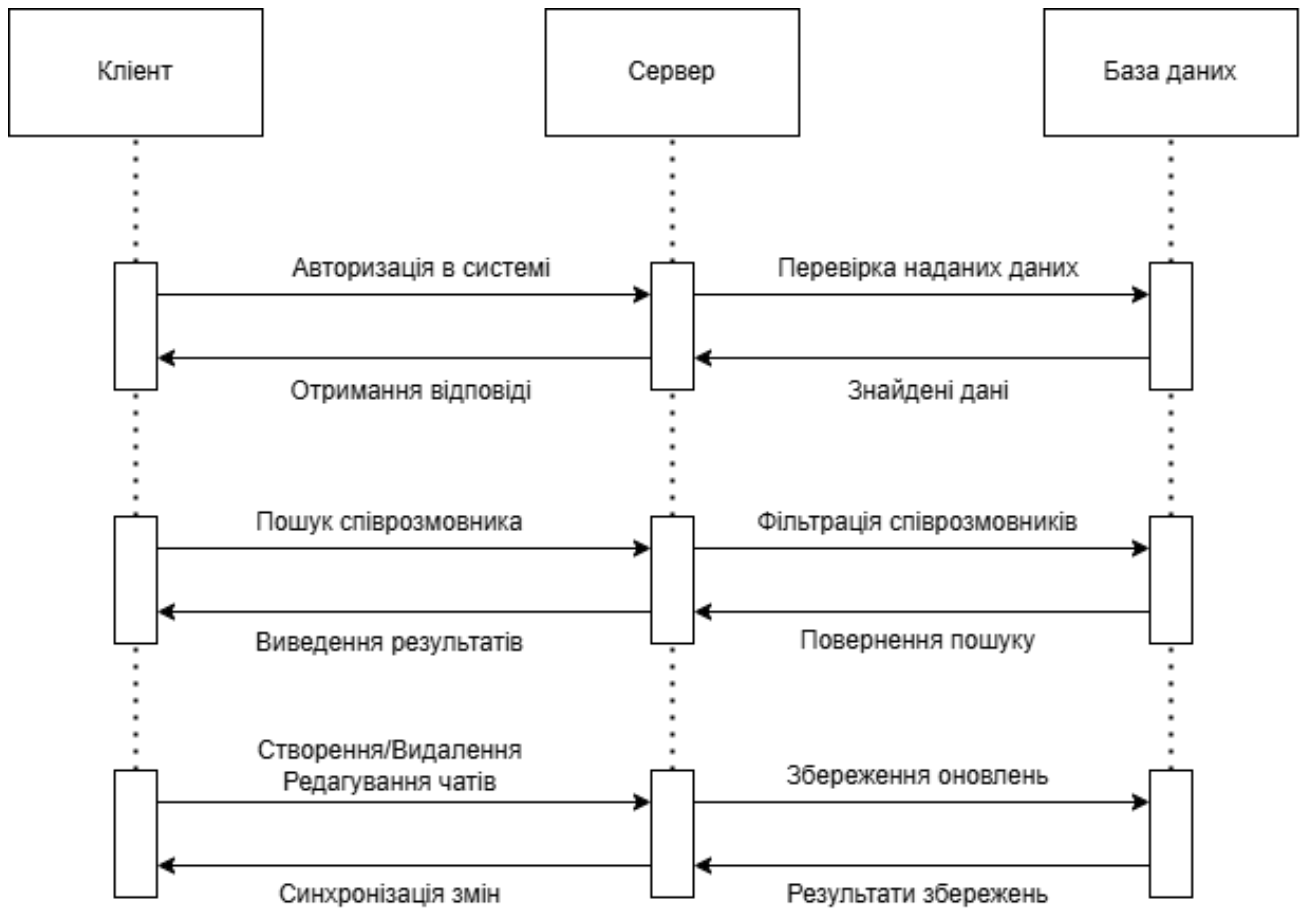


Рисунок 3.3 – Схема функціонування вебчату, який розроблюється

Основні етапи функціонування системи:

а) авторизація в системі:

- 1) користувач вводить облікові дані;
- 2) сервер перевіряє надану інформацію;
- 3) здійснюється звернення до бази даних для підтвердження існування облікового запису;
- 4) у випадку успіху – повертається відповідь з даними сесії;

б) пошук співрозмовника:

- 1) клієнт надсилає запит на пошук;
- 2) сервер виконує фільтрацію користувачів відповідно до параметрів пошуку;
- 3) база даних надає відповідні записи;
- 4) результати пошуку повертаються клієнту;

- в) створення чатів:
 - 1) користувач ініціює створення нового чату;
 - 2) сервер фіксує новий чат у системі;
 - 3) інформація зберігається у базі даних;
- г) синхронізація змін:
 - 1) після внесення змін (нові повідомлення, учасники тощо), відбувається синхронізація;
 - 2) сервер отримує оновлені дані та передає їх у базу;
 - 3) клієнт отримує оновлені результати для відображення;
- д) редагування чатів:
 - 1) користувач надсилає запит на зміну параметрів чату (назва, опис);
 - 2) сервер перевіряє права доступу та оновлює дані;
 - 3) зміни зберігаються у базі даних та повертаються клієнту для відображення;
- е) видалення чатів:
 - 1) користувач ініціює видалення чату;
 - 2) сервер обробляє запит та видаляє чат разом із повідомленнями;
 - 3) база даних оновлюється, а клієнту надсилається підтвердження видалення;
- ж) відправка файлів:
 - 1) користувач додає файл до повідомлення;
 - 2) файл передається на сервер, де проходить перевірку та зберігається;
 - 3) у базу даних записується інформація про файл;
 - 4) клієнт отримує оновлене повідомлення з прикріпленим файлом;
- з) редагування повідомлень:
 - 1) користувач надсилає запит на редагування свого

повідомлення;

2) сервер перевіряє права доступу та оновлює текст повідомлення;

3) зміни зберігаються у базі даних та відображаються у клієнтському інтерфейсі.

Розробка вебчату передбачає реалізацію ряду функцій, що забезпечують повноцінну взаємодію користувача із системою: обмін повідомленнями, створення та управління чатами, перегляд історії тощо. Для системного представлення цих можливостей використовується діаграма прецедентів (рисунок 3.4), що є складовою стандарту моделювання UML.

Ця діаграма дозволяє візуально показати, як саме користувач (актор) взаємодіє з програмною системою через набір доступних йому функцій (прецедентів). Вона демонструє основні сценарії поведінки, залежності між прецедентами (зокрема включення *include* і розширення *extend*), а також дозволяє визначити межі системи, яку проєктують.

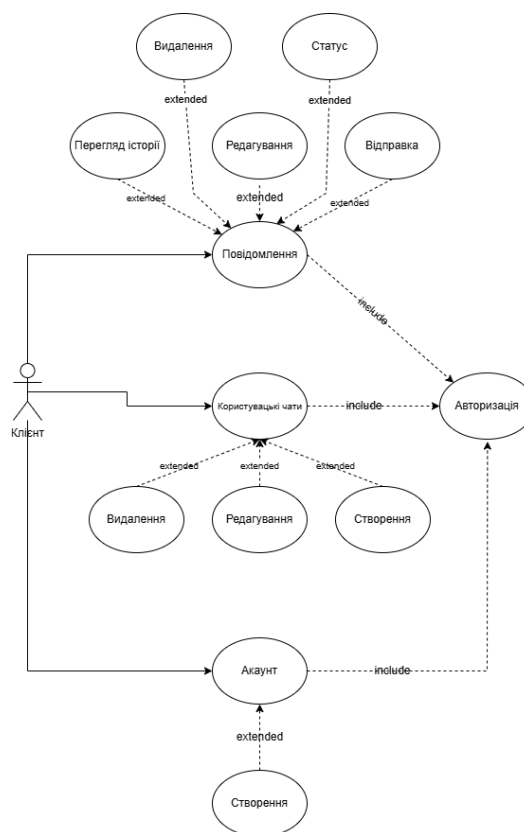


Рисунок 3.4 – Діаграма прецедентів вебчату, який розроблюється

Основні прецеденти:

а) авторизація:

- 1) є обов'язковою умовою для доступу до інших функцій системи;
- 2) виконується як базовий сценарій, що включається (include) у всі інші операції;
- 3) забезпечує перевірку облікових даних користувача перед виконанням дій;

б) користування чатами:

- 1) охоплює основну взаємодію користувача з чатами (створення, редагування, видалення);
- 2) включає (include) авторизацію як необхідний крок;
- 3) передбачає доступ до функцій управління чатами;

в) повідомлення:

- 1) дозволяє користувачеві обмінюватися повідомленнями в межах чату;
- 2) включає (include) авторизацію для підтвердження доступу;
- 3) підтримує додаткові сценарії (розширення extended);

г) створення облікового запису:

- 1) окремий сценарій для реєстрації нового користувача в системі;
- 2) включає (include) авторизацію як крок перевірки та підтвердження облікового запису;

д) створення чатів:

- 1) сценарій, що дозволяє користувачу створювати нові чати;
- 2) включає (include) авторизацію;
- 3) підтримує розширення (extended) для додаткових можливостей у чаті;

е) редагування чатів:

- 1) сценарій для внесення змін до параметрів чату (назва, опис, учасники);

- 2) включає (include) авторизацію;
- 3) можливість змінювати дані доступна лише після проходження перевірки доступу;

ж) видалення чатів:

- 1) дозволяє користувачу видаляти створені чати;
- 2) вимагає авторизації для підтвердження прав на видалення.

3.4 Структура бази даних

Для забезпечення роботи вебчату було спроектовано реляційну базу даних, яка охоплює основні аспекти функціональності системи: реєстрацію та логін користувачів, обмін повідомленнями, створення чатів та збереження історії взаємодій.

База даних складається з декількох логічно пов'язаних таблиць, кожна з яких виконує певну роль у функціонуванні вебчату. Основними об'єктами, які моделюються у базі, є користувачі, чати та повідомлення.

На рисунку 3.5 зображено структурну схему бази даних.

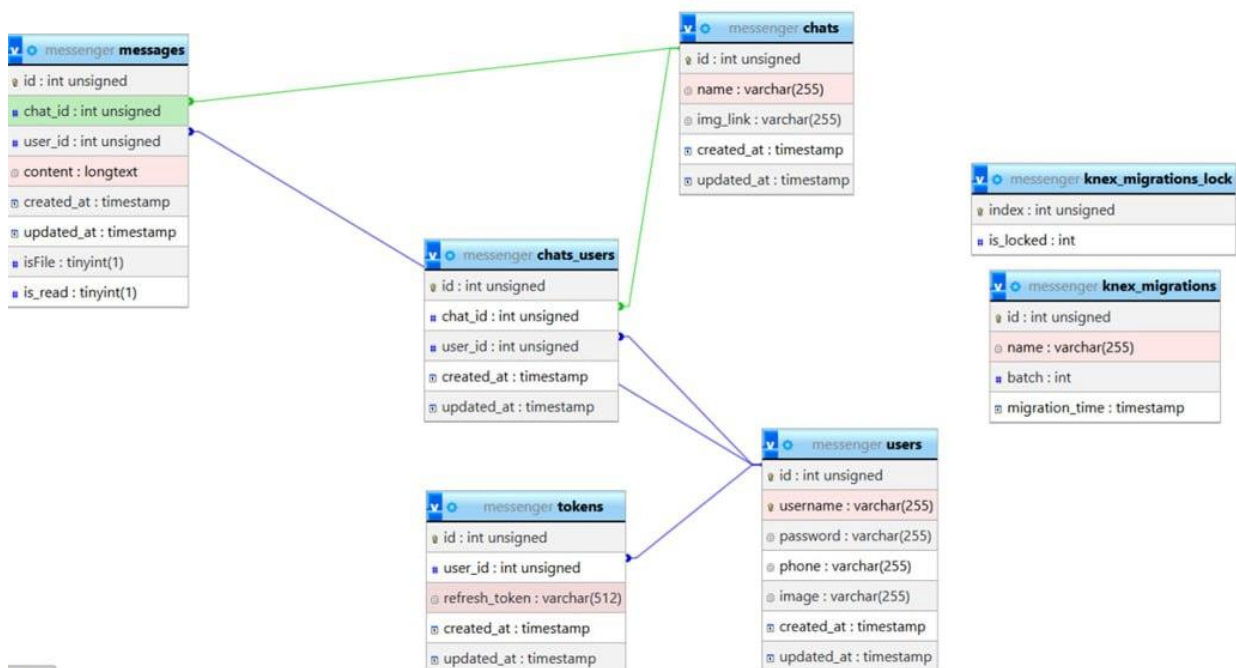


Рисунок 3.5 – Структурна схема бази даних вебчату, який розроблюється

У таблиці 3.2 зображено опис кожної таблиці, типи полів та їх призначення.

Таблиця 3.2 – Опис структурної схеми бази даних

Таблиця	Поле	Тип даних	Призначення
1	2	3	4
users	id	int	Унікальний ідентифікатор користувача
	username	varchar	Ім'я користувача (нікнейм)
	firstName	varchar	Ім'я користувача
	lastName	varchar	Прізвище користувача
	email	varchar	Електронна пошта
	password	varchar	Хеш пароля користувача
	phone	varchar	Номер телефону
	image	varchar	Посилання на зображення профілю
	verificationLink	varchar	Посилання для підтвердження електронної пошти
	isVerified	tinyint	Статус верифікації (0 – ні, 1 – так)
	created_at	timestamp	Дата створення запису
	updated_at	timestamp	Дата останнього оновлення запису
tokens	id	int	Унікальний ідентифікатор токена
	user_id	int	Зовнішній ключ на користувача
	refresh_token	varchar	Токен оновлення, що використовується для логіну
	created_at	timestamp	Дата створення
	updated_at	timestamp	Дата оновлення
chats	id	int	Унікальний ідентифікатор чату
	name	varchar	Назва чату (наприклад, група)

Продовження таблиці 3.2

1	2	3	4
	img_link	varchar	Посилання на аватар/іконку чату
	created_at	timestamp	Дата створення
	updated_at	timestamp	Дата оновлення
chat_users	id	int	Унікальний запис
	chat_id	int	Зовнішній ключ на таблицю chats
	user_id	int	Зовнішній ключ на таблицю users
	created_at	timestamp	Дата додавання користувача до чату
	updated_at	timestamp	Дата оновлення
messages	id	int	Унікальний ідентифікатор повідомлення
	chat_id	int	Чат, до якого належить повідомлення
	user_id	int	Користувач, який відправив повідомлення
	content	longtext	Текст повідомлення
	created_at	timestamp	Час створення повідомлення
	updated_at	timestamp	Час оновлення повідомлення (за потреби)
knex_migrations	id	int	Ідентифікатор міграції
	name	varchar	Назва файлу міграції
	batch	int	Номер партії міграцій
	migration_time	timestamp	Час застосування міграції

3.5 Проєктування інтерфейсу вебчату

Інтерфейс вебчату розроблений таким чином, щоб забезпечити зручний, інтуїтивно зрозумілий та швидкий доступ до основних функцій системи. Представлено покроковий алгоритм взаємодії користувача з системою на основі зображень, які демонструють ключові етапи користування чатом – від входу до листування та виходу з облікового запису.

Нижче подано опис кожного екрана, що відповідає послідовним діям користувача у процесі користування вебчатом.

1. Сторінка авторизації користувача у десктопній і мобільній версіях (рисунки 5.1 – 5.2).

Після відкриття вебчату користувач потрапляє на сторінку авторизації. У центральній частині вікна розміщено поля введення логіна та пароля. Нижче знаходиться кнопка "login", яка дозволяє перейти до основного інтерфейсу чату після успішної перевірки облікових даних.

Під формою входу знаходиться кнопка "Do you want to register?", яка призначена для нових користувачів – натиснувши на неї, можна перейти до сторінки реєстрації. Дизайн сторінки витриманий у світлих тонах, що робить інтерфейс простим та зручним для сприйняття.

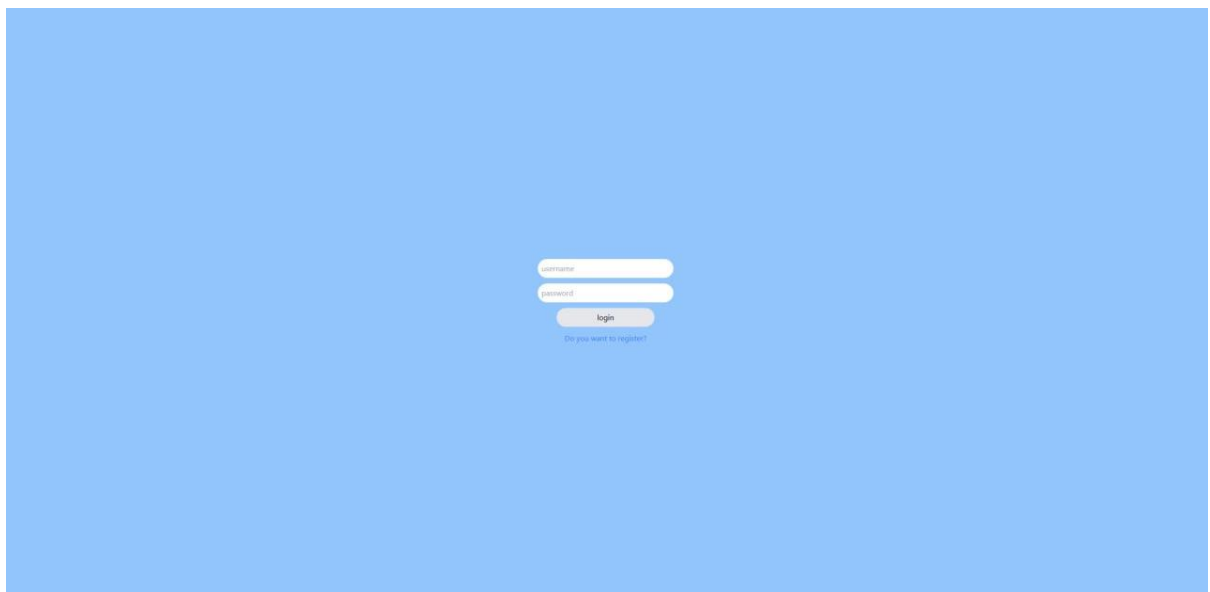


Рисунок 5.1 – Сторінка авторизації вебчату (десктопна версія)

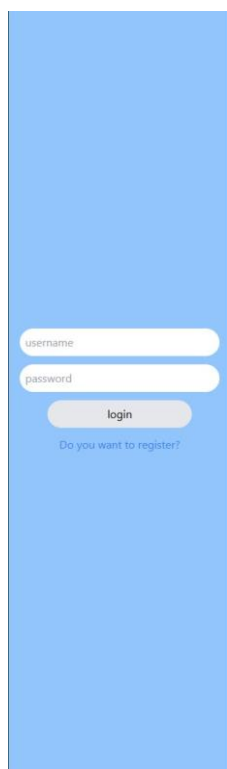
The image shows a mobile version of a login page. It features a light blue background. In the center, there are two white input fields stacked vertically. The top field is labeled 'username' and the bottom field is labeled 'password'. Below these fields is a light blue button with the text 'login'. Underneath the button, there is a link that says 'Do you want to register?' in a smaller, lighter blue font.

Рисунок 5.2 – Сторінка авторизації вебчату (мобільна версія)

2. Сторінка реєстрації нового користувача у десктопній і мобільній версіях (рисунки 5.3 – 5.4).

При натисканні на кнопку "Do you want to register?" користувач переходить до форми реєстрації. У центральній частині вікна розміщено поля для введення логіна, пароля, а також підтвердження пароля. Після заповнення всіх полів потрібно натиснути кнопку "register" для створення нового облікового запису.

Під формою реєстрації знаходиться кнопка "Do you want to login?", яка призначена для авторизованих користувачів – натиснувши на неї, можна перейти до сторінки логіну.

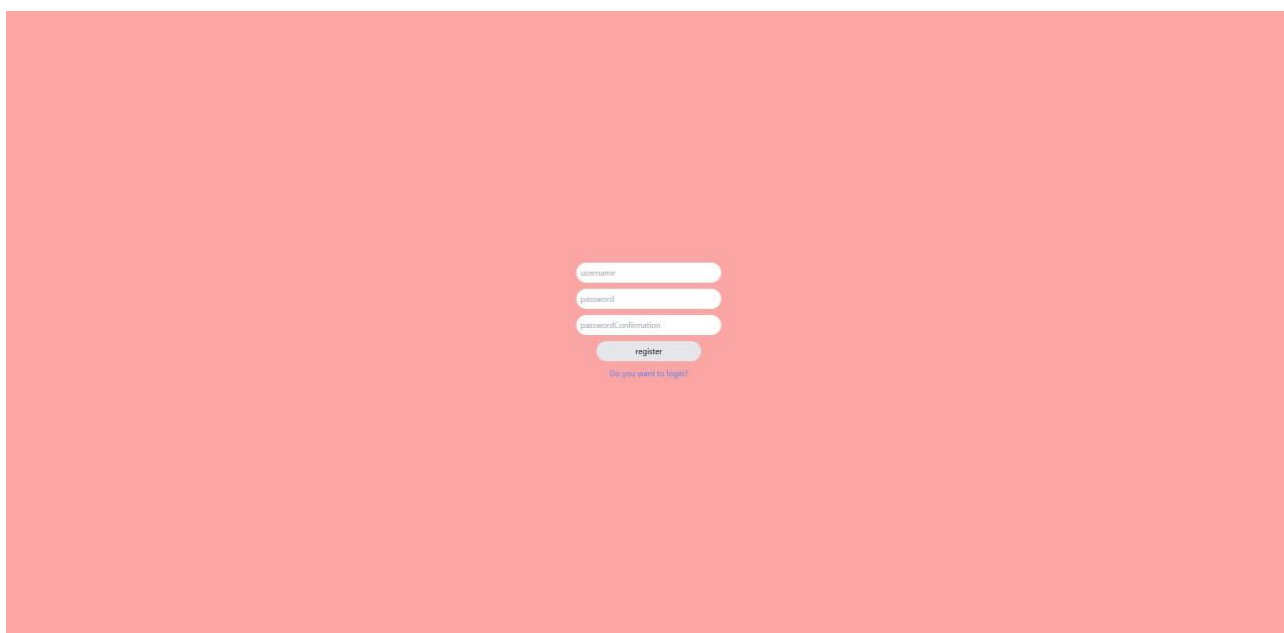
The image shows a desktop version of a registration page. It features a light red background. In the center, there is a white registration form. The form contains three input fields: 'username', 'password', and 'passwordConfirmation'. Below these fields is a blue 'register' button. Under the button, there is a link that says 'Do you want to login?' in a smaller, lighter blue font.

Рисунок 5.3 – Сторінка реєстрації вебчату (десктопна версія)

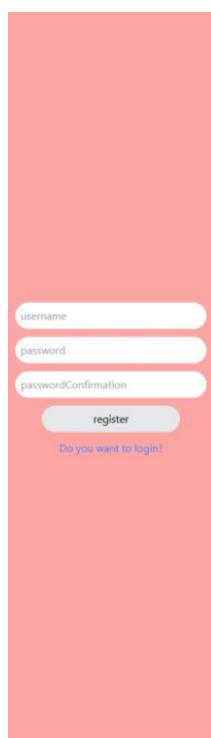
The image shows a mobile version of the registration page. It features a light red background. In the center, there is a white registration form. The form contains three input fields: 'username', 'password', and 'passwordConfirmation'. Below these fields is a blue 'register' button. Under the button, there is a link that says 'Do you want to login?' in a smaller, lighter blue font.

Рисунок 5.4 – Сторінка реєстрації вебчату (мобільна версія)

3. Сторінка основного інтерфейсу вебчату після входу у десктопній і мобільній версіях (рисунки 5.5 – 5.6).

Після успішної авторизації або реєстрації користувач потрапляє до головного вікна чату. У верхній частині лівої панелі розташоване поле пошуку.

Нижче розміщується список користувачів, з якими вже було розпочато спілкування, та дві функціональні кнопки: "Add New Chat" – для створення нового діалогу, та "Logout" – для виходу з облікового запису.

У правій частині вікна знаходиться область повідомлень. Для відображення історії повідомлень необхідно обрати співрозмовника з існуючих або додати нового. Історія повідомлень включає як надіслані, так і отримані повідомлення.



Рисунок 5.5 – Сторінка основного інтерфейсу вебчату (десктопна версія)



Рисунок 5.6 – Сторінка основного інтерфейсу вебчату (мобільна версія)

4. Сторінка обміну повідомленнями між користувачами у десктопній і мобільній версіях (рисунки 5.7 – 5.8).

Після вибору співрозмовника з наявного списку або створення нового чату, у правій частині інтерфейсу відображається історія переписки. Повідомлення поділяються візуально: власні – вирівнюються праворуч, повідомлення співрозмовника – ліворуч. Нижче знаходиться текстове поле для введення нового повідомлення. Введене повідомлення одразу з'являється в історії переписки. Інтерфейс підтримує автоматичне оновлення, що дозволяє бачити нові повідомлення в режимі реального часу.

У мобільній версії інтерфейс адаптовано: елементи розміщені вертикально, що забезпечує зручність користування на екранах з обмеженим простором.



Рисунок 5.7 – Сторінка обміну повідомленнями між користувачами вебчату (десктопна версія)



Рисунок 5.8 – Сторінка обміну повідомленнями між користувачами вебчату (мобільна версія)

5. Сторінка пошуку співрозмовника через спливаюче вікно у десктопній версії (рисунок 5.9).

Після натискання на поле "Search" відкривається спливаюче вікно для пошуку користувача. У цьому вікні передбачене поле введення, куди необхідно ввести ім'я або логін співрозмовника. Система автоматично підбирає відповідних користувачів зі списку зареєстрованих.

Після вибору потрібного користувача створюється новий діалог, і користувач повертається до основного інтерфейсу чату, де відкривається відповідна вкладка спілкування.

Цей функціонал дозволяє швидко знаходити співрозмовників, не перегортаючи вручну великий список.

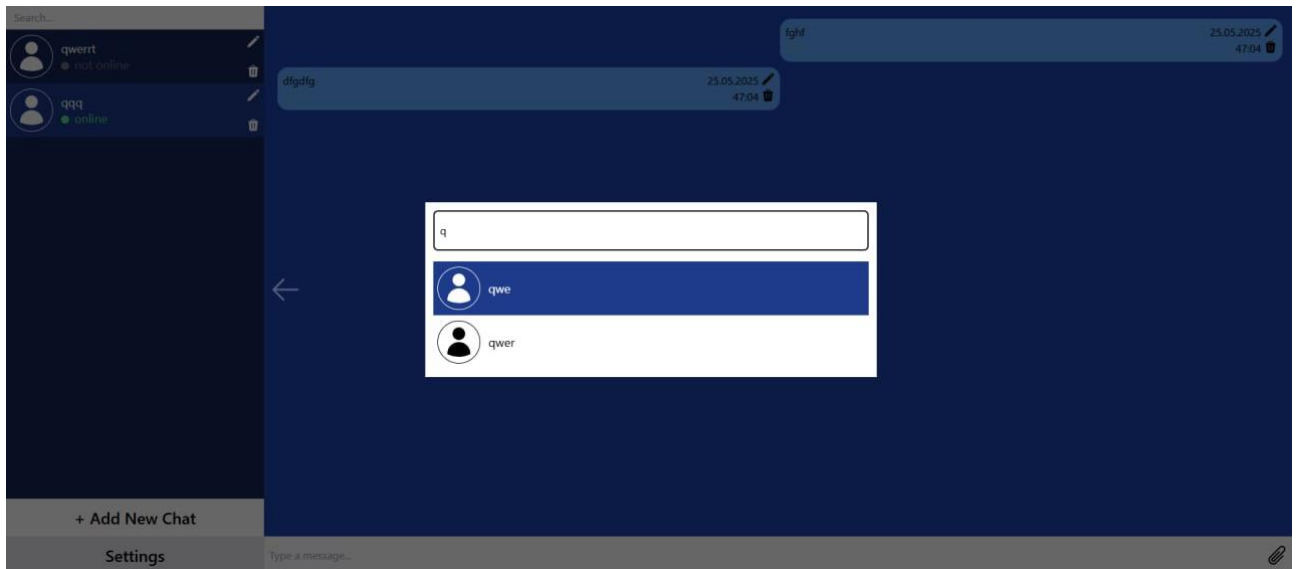


Рисунок 5.9 – Сторінка пошуку співрозмовника через спливаюче вікно вебчату (десктопна версія)

6. Сторінка додавання файлу у десктопній версії (рисунок 5.10).

Ця сторінка відображає інтерфейс обміну повідомленнями між користувачами з активованою функцією додавання файлу. У нижній частині інтерфейсу присутня кнопка "Send Photo", яка слугує для прикріплення зображень або інших файлів до повідомлення. Натискання на неї дозволяє користувачеві обрати файл зі свого пристрою для надсилання у чат.

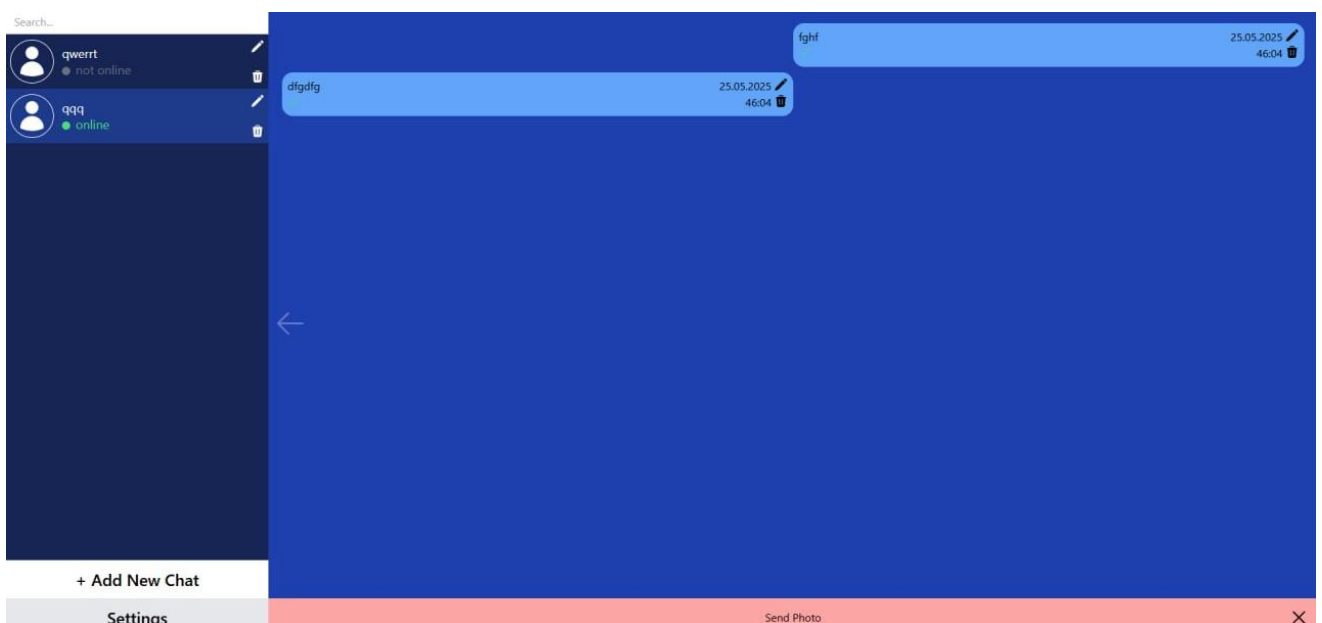


Рисунок 5.10 – Сторінка додавання файлу вебчату (десктопна версія)

7. Сторінка налаштувань юзера у десктопній версії (рисунок 5.11).

Ця сторінка призначена для редагування особистих даних користувача. Вона містить форму з кількома полями вводу, серед яких: "Username", "New password", "New password confirmation" та "Phone". Користувач може змінити своє ім'я користувача, оновити пароль, підтвердити новий пароль та вказати номер телефону.

У верхній частині сторінки розміщене місце для аватарки користувача, яке наразі не завантажене. Після внесення змін користувач має змогу натиснути кнопку "Save", щоб зберегти оновлену інформацію.

Також на сторінці присутні дві додаткові кнопки:

- "Logout" – для виходу з облікового запису;
- "Back" – для повернення до попереднього екрану.

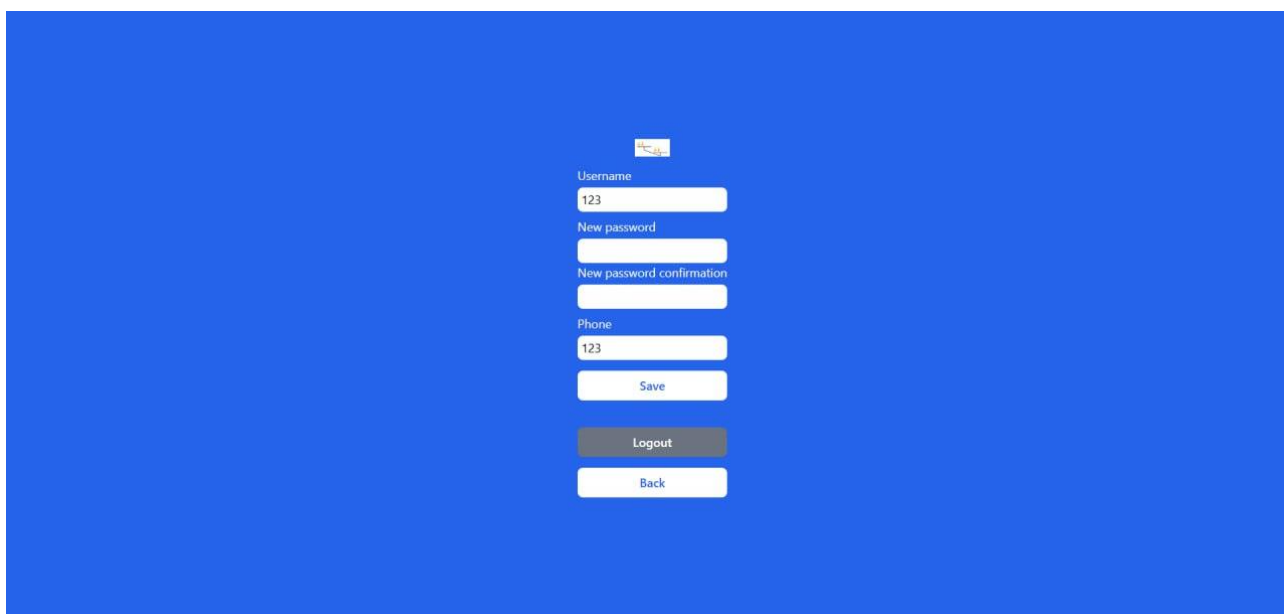
The image shows a desktop version of a user settings page with a solid blue background. In the center, there is a white form. At the top of the form is a small square placeholder for an avatar. Below it are four input fields: 'Username' (containing '123'), 'New password', 'New password confirmation', and 'Phone' (containing '123'). Below these fields are three buttons: a white 'Save' button, a grey 'Logout' button, and a white 'Back' button.

Рисунок 5.11 – Сторінка налаштувань юзера вебчату (десктопна версія)

3.6 Висновки за розділом 3

У даному розділі було здійснено детальну розробку та опис основних компонентів системи вебчату. На основі визначених функціональних можливостей сформовано вимоги до системи, які охоплюють створення

акаунтів, логін користувачів, створення чатів, обмін повідомленнями в реальному часі та підтримку кількох учасників у чаті.

Розглянуто архітектуру програмного забезпечення, яка реалізує принцип розділення відповідальностей та дозволяє легко масштабувати й модифікувати систему. Представлена логіка роботи демонструє, як користувачі взаємодіють із системою – від реєстрації до обміну повідомленнями – з урахуванням обробки сесій, перевірки прав доступу та асинхронної передачі даних.

Особливу увагу приділено структурі бази даних, яка забезпечує зберігання всіх необхідних даних – користувачів, чатів, повідомлень, токенів та історії змін. Реляційна модель дозволяє підтримувати цілісність даних і забезпечує ефективну взаємодію між компонентами системи.

Таким чином, розроблені компоненти формують цілісну та функціональну основу для стабільної роботи вебчату та забезпечують подальші можливості для розширення системи.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення та умови використання вебчату

Для забезпечення повноцінного та стабільного функціонування вебчату необхідно дотримуватися ряду технічних вимог, як для серверної, так і для клієнтської частини. Ці вимоги гарантують безперебійну роботу системи, високу швидкість обробки запитів, безпечний обмін даними та комфортне використання вебчату.

Для запуску серверної частини вебчату потрібні:

- надійне серверне середовище – це може бути хмарна платформа (наприклад, Heroku, Vercel, Render, AWS або інші) або фізичний сервер з можливістю постійної роботи без перерв. Необхідна мінімальна конфігурація – це система із сучасним процесором, щонайменше 4 ГБ оперативної пам'яті та підтримкою Node.js (або іншого середовища, у якому розгорнуто сервер);
- постійне з'єднання з мережею – безперервний доступ до Інтернету є обов'язковим, оскільки система обміну повідомленнями працює в режимі реального часу;
- можливість обробки запитів одночасно від кількох користувачів – для цього необхідна підтримка паралельних підключень через WebSocket або іншу технологію зв'язку в реальному часі.

Що потрібно клієнтській частині для використання вебчату:

- пристрій із сучасним браузером – підійде будь-який комп'ютер, ноутбук, планшет або смартфон з актуальною версією браузера (Chrome, Microsoft Edge, Firefox, Safari);
- підключення до Інтернету – під час спілкування необхідне стабільне інтернет-з'єднання, адже чат не зберігає повідомлення локально, а миттєво передає їх між клієнтами через сервер;
- наявність облікового запису – для користування деякими функціями чату, такими як створення нових розмов чи збереження історії, може

знадобитися вхід у систему.

Інші технічні особливості:

- інтерфейс чату розрахований на просте використання – немає складних елементів або перевантаження графікою, тому навіть малопотужні пристрої справляються з ним без проблем;

- адаптивність – вебчат автоматично підлаштовується під розмір екрана, тому зручно користуватись як із телефону, так і з ноутбука.

4.2 Технічні характеристики вебчату

Технічні характеристики вебчату визначають основні функціональні та нефункціональні вимоги, що забезпечують ефективну роботу системи. Вони охоплюють не лише можливості вебчату, але й його поведінку під час використання, безпеку, продуктивність та обмеження, які впливають на якість та ефективність роботи.

Функціональні вимоги:

- а) реєстрація та авторизація користувача

Користувач має змогу створити обліковий запис і здійснювати вхід у систему за допомогою логіна та пароля;

- б) списки контактів та пошук користувачів

Система дозволяє знаходити інших користувачів за іменем або іншими ідентифікаторами та додавати їх до списку контактів;

- в) обмін повідомленнями в реальному часі

Передача повідомлень здійснюється миттєво без оновлення сторінки, за допомогою WebSocket-з'єднання або аналогічної технології;

- г) створення нового чату

Користувач має можливість створювати індивідуальні або групові чати, обираючи учасників із власного списку контактів;

- д) редагування чатів

Користувач може редагувати параметри чату, зокрема назву, опис та

склад учасників;

е) вихід з системи

Користувач має можливість безпечно вийти зі свого облікового запису та завершити поточну сесію.

Нефункціональні вимоги:

а) продуктивність

Система повинна обробляти щонайменше 100 одночасних підключень без істотного зниження швидкості відповіді (<500 мс на повідомлення);

б) масштабованість

Архітектура вебчату повинна дозволяти розширення – додавання нових серверів або розміщення компонентів окремо (frontend, backend, база даних);

в) надійність

При збої в мережі або оновленні сторінки повідомлення мають зберігатися у черзі або локально (залежно від реалізації), щоб уникнути втрати даних;

г) сумісність

Підтримка всіх основних браузерів і пристроїв. Чат повинен відображатись однаково коректно на різних розмірах екранів;

д) адаптивність інтерфейсу

Інтерфейс має автоматично пристосовуватися до ширини вікна браузера без втрати функціональності.

4.3 Ініціалізація, запуск і взаємодія з компонентами

Ініціалізація та запуск вебчату є критично важливими етапами, що забезпечують правильну роботу всієї системи. Після ініціалізації програма повинна бути готова до взаємодії з користувачем, а також із серверами та іншими компонентами, необхідними для підтримки функціональності вебчату. Основні етапи цього процесу включають звернення до програми, обробку вхідних і вихідних даних, а також ефективну обробку помилок і повідомлень.

4.3.1 Звернення до програми

Запуск програми: Після відкриття вебчату або запуску мобільного додатку користувачі потрапляють на екран авторизації. Програма ініціалізує всі необхідні компоненти та підключається до серверної частини через інтерфейс користувача (UI).

Підключення до сервера: Програма на клієнтському боці встановлює з'єднання з сервером для обміну даними та отримання актуальної інформації про чати, повідомлення та користувачів.

Ініціалізація інтерфейсу: Після встановлення з'єднання на клієнтському боці відображається головний інтерфейс вебчату, де користувач може вводити повідомлення, переглядати чати та іншу інформацію. Відбувається завантаження даних з бази даних і налаштування зв'язку з сервером для наступних операцій.

4.3.2 Вхідні та вихідні дані

Вхідні дані:

- користувачі вводять свої дані для авторизації в системі (логін та пароль або інші методи авторизації);
- після успішного входу клієнтська частина програми отримує відповідні дані про доступні чати, повідомлення, профіль користувача тощо;
- кожен вхідний запит до серверної частини містить параметри, такі як ідентифікатор користувача, тип операції (реєстрація, авторизація, обробка повідомлень тощо) та інші параметри, необхідні для виконання запиту.

Вихідні дані:

- вихідні дані включають відповідь від сервера, яка містить результат запиту користувача: підтвердження авторизації, повідомлення про успішну реєстрацію, нові повідомлення в чаті або підтвердження створення нового чату;
- сервер також надсилає дані користувачу через WebSocket, якщо це необхідно для реалізації обміну повідомленнями в реальному часі.

4.3.3 Обробка помилок і повідомлень у системі

Обробка помилок на клієнтському боці:

а) програма повинна правильно обробляти помилки, що виникають при взаємодії з сервером, наприклад, неправильний логін або пароль, неуспішна спроба підключення до серверу, відсутність інтернет-з'єднання тощо;

б) всі помилки, що виникають на клієнтському боці, мають відображатися у вигляді відповідних повідомлень, що сповіщають користувача про проблему. Ці повідомлення повинні бути зрозумілими та пропонувати варіанти вирішення проблеми (наприклад, перевірка введених даних або повторна спроба з'єднання).

Обробка помилок на сервері:

а) сервер повинен обробляти всі запити від користувачів і надавати чіткі повідомлення у випадку помилки. Це можуть бути помилки авторизації, неіснуючі чати, помилки при роботі з базою даних тощо;

б) якщо виникає помилка при обробці запиту, сервер має надіслати помилку у вигляді коду (наприклад, 404 або 500) з поясненням причини проблеми.

Повідомлення про статус операцій:

а) кожна операція, така як створення чату, надсилання повідомлення, додавання користувача до списку контактів, повинна супроводжуватися повідомленням про успішне виконання операції або, у випадку помилки, відповідним повідомленням;

б) ці повідомлення мають бути відображені на інтерфейсі користувача для забезпечення зручного та зрозумілого досвіду взаємодії.

4.4 Чек-лист тестування функціоналу

Перед запуском вебчату до експлуатації було проведено тестування основних функціональних можливостей системи (створення чек-листу, який

представлено у таблиці 4.1). Перевірки виконувались як у мобільній версії, так і на десктопній версії застосунку через веббраузери.

Перелік основних функцій, які підлягали тестуванню:

- відображення інтерфейсу;
- реєстрація нового користувача;
- авторизація користувача;
- відправлення та отримання повідомлень у чаті;
- підключення до WebSocket-серверу;
- адаптивність інтерфейсу під розміри екрану;
- вихід із системи (logout).

Тестування проводилося на:

- мобільна версія: Google Chrome на Android 14.0 (Samsung Galaxy);
- десктопна версія: Google Chrome на Windows 11.

Таблиця 4.1 – Чек-лист вебчату

Перевірка	Android 14.0, Samsung	Windows 11, Chrome
Коректне відображення інтерфейсу	Passed	Passed
Реєстрація нового користувача	Passed	Passed
Авторизація з правильними даними	Passed	Passed
Відображення помилки при хибному логіні/паролі	Passed	Passed
Відправка повідомлень у чаті	Passed	Passed
Отримання повідомлень у чаті	Passed	Passed
Працююче WebSocket-підключення	Passed	Passed
Кнопка "Logout" повертає на сторінку логіну	Passed	Passed

Таким чином, усі основні функції вебчату успішно пройшли тестування як на мобільному пристрої, так і на десктопній версії. Система стабільно реагує на дії користувача, правильно обробляє повідомлення, та забезпечує коректне

відображення інтерфейсу на різних пристроях. Помилки або збоїв у роботі виявлено не було, що дозволяє надавати систему до подальшої експлуатації.

4.5 Висновки за розділом 4

У четвертому розділі було ретельно проаналізовано керівництво для програмістів, яке охоплює процеси розробки вебчату, його ініціалізації та взаємодію між компонентами. Особлива увага присвячена функціональним вимогам, що визначають основні можливості системи, зокрема реєстрацію користувачів, обмін повідомленнями в реальному часі та створення чатів. Це сприяє зручності та ефективності комунікації в межах вебчату.

Процес ініціалізації та запуску системи включає всі основні етапи взаємодії між клієнтом і сервером, такі як налаштування з'єднання, обробка вхідних і вихідних даних, а також коректне управління помилками. Це забезпечує стабільну роботу програми навіть у випадку непередбачених ситуацій.

Технічні характеристики, включаючи підтримку обміну повідомленнями в реальному часі та наявність масштабованої архітектури, гарантують високу продуктивність і надійність вебчату. Завдяки продуманому підходу до обробки помилок та інформування користувачів, система має можливість ефективно реагувати на будь-які проблеми чи некоректні дії, при цьому зберігаючи високий рівень задоволення користувачів.

Отже, усі представлені компоненти та процеси сприяють формуванню потужного, стабільного та зручного вебчату, що відповідає запитам користувачів і забезпечує високу ефективність та надійність у реальних умовах експлуатації.

ВИСНОВКИ

У процесі виконання цієї роботи було розроблено повнофункціональний вебчат, який забезпечує зручну, швидку та безпечну комунікацію між користувачами в реальному часі. Систему створено з урахуванням сучасних вимог – як з технічної, так і з користувацької точки зору.

Під час дослідження проаналізовано існуючі популярні платформи для онлайн-комунікації, вивчено їх переваги та недоліки, що дозволило сформувані обґрунтовані вимоги до майбутньої системи. Завдяки цьому вебчат отримав інтуїтивно зрозумілий інтерфейс, адаптивний дизайн, базовий функціонал для обміну повідомленнями та можливість масштабування.

У рамках реалізації було обрано доцільні інструменти й середовище розробки, побудовано логічну архітектуру системи, створено структуру бази даних, що відповідає потребам зберігання та обробки інформації. Особливу увагу приділено стабільності роботи вебчату, обробці помилок, зручності ініціалізації та взаємодії з користувачем.

Описано як технічні аспекти запуску й налаштування, так і докладну інструкцію для кінцевого користувача. Створено чіткий алгоритм користування, включно з авторизацією та реєстрацією, пошуком співрозмовника, веденням діалогів і завершенням сеансу. Усі дії користувача супроводжуються інформаційними повідомленнями, що підвищує рівень зручності й взаєморозуміння із системою.

Проведене тестування підтвердило коректність роботи основного функціоналу. Результати експериментального дослідження показали, що система відповідає поставленим вимогам, демонструє стабільність і задовільну швидкодію в типовому середовищі експлуатації.

Таким чином, розроблений вебчат є прикладом ефективного застосування вебтехнологій для реалізації платформи миттєвого обміну повідомленнями, яка поєднує простоту використання з технічною надійністю.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Slack [Electronic resource]. – Access mode:
<https://slack.com/downloads/windows>.
2. Discord [Electronic resource]. – Access mode:
<https://discord.com/download>.
3. Telegram [Electronic resource]. – Access mode:
<https://desktop.telegram.org/?setln=en>.
4. Visual Studio [Electronic resource]. – Access mode:
<https://code.visualstudio.com/>.
5. WebStorm [Electronic resource]. – Access mode:
<https://www.jetbrains.com/webstorm/>.
6. Atom [Electronic resource] – Access mode: <https://atom-editor.cc/>.
7. Sublime Text [Electronic resource]. – Access mode:
<https://www.sublimetext.com/>.
8. MVC Design Pattern [Electronic resource]. – Access mode:
<https://www.geeksforgeeks.org/mvc-design-pattern/>.
9. MVVM Pattern [Electronic resource]. – Access mode:
<https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm>.
10. Microservices Architecture [Electronic resource]. – Access mode:
<https://www.geeksforgeeks.org/microservice-architecture-introduction-challenges-best-practices/>.
11. Monolithic Architecture [Electronic resource]. – Access mode:
<https://www.geeksforgeeks.org/monolithic-architecture-system-design/>.
12. MySQL [Electronic resource] – Access mode: <https://www.mysql.com/>.
13. PostgreSQL [Electronic resource]. – Access mode:
<https://www.postgresql.org/>.
14. MongoDB [Electronic resource]. – Access mode:
<https://www.mongodb.com/>.
15. SQLite [Electronic resource]. – Access mode:

<https://sqlite.org/index.html>.

16. JavaScript [Electronic resource]. – Access mode:
<https://developer.mozilla.org/en-US/docs/Web/JavaScript>.

17. Python [Electronic resource]. – Access mode: <https://www.python.org/>.

18. PHP [Electronic resource]. – Access mode: <https://www.php.net/>.

19. C# [Electronic resource]. – Access mode:
<https://dotnet.microsoft.com/en-us/languages/csharp>.

ДОДАТОК А

Технічне завдання

A.1 Вступ

Програмне забезпечення буде використовуватися для підтримки процесу обміну повідомленнями між користувачами в режимі реального часу. Вебчат дозволить створювати текстові діалоги, зберігатиме історію повідомлень, надаватиме можливість здійснювати пошук співрозмовників та працювати з декількох пристроїв без потреби встановлення окремого клієнта.

A.2 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему: Програмна реалізація вебчату, затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 р..

A.3 Призначення розробки

Призначенням розробленого вебчату є забезпечення зручного, швидкого та доступного каналу комунікації між користувачами через інтернет у режимі реального часу. Система дозволяє обмінюватися текстовими повідомленнями, створювати нові діалоги, переглядати історію спілкування та здійснювати пошук співрозмовників у базі зареєстрованих користувачів.

Вебчат призначений для використання як у десктопних браузерах, так і на мобільних пристроях, що розширює можливості користувачів у різних умовах. Його реалізація базується на сучасних вебтехнологіях, що забезпечує стабільну роботу, адаптивний інтерфейс та високу швидкодію.

A.4 Основні вимоги до програми

Програма повинна відповідати низці вимог, які забезпечують її коректну та ефективну роботу в умовах реального використання. Основні вимоги

поділяються на функціональні та нефункціональні. Функціональні визначають, що саме повинна робити система, а нефункціональні – якою вона повинна бути з точки зору зручності, продуктивності та сумісності.

A.4.1 Функціональні вимоги

а) реєстрація та авторизація користувача

Користувач має змогу створити обліковий запис і здійснювати вхід у систему за допомогою логіна та пароля;

б) списки контактів та пошук користувачів

Система дозволяє знаходити інших користувачів за іменем або іншими ідентифікаторами та додавати їх до списку контактів;

в) обмін повідомленнями в реальному часі

Передача повідомлень здійснюється миттєво без оновлення сторінки, за допомогою WebSocket-з'єднання або аналогічної технології;

г) створення нового чату

Користувач має можливість створювати індивідуальні або групові чати, обираючи учасників із власного списку контактів;

д) редагування чатів

Користувач може редагувати параметри чату, зокрема назву, опис та склад учасників;

е) вихід з системи

Користувач має можливість безпечно вийти зі свого облікового запису та завершити поточну сесію.

A.4.2 Нефункціональні вимоги

а) продуктивність

Система повинна обробляти щонайменше 100 одночасних підключень без істотного зниження швидкості відповіді (<500 мс на повідомлення);

б) масштабованість

Архітектура вебчату повинна дозволяти розширення – додавання нових серверів або розміщення компонентів окремо (frontend, backend, база даних);

в) надійність

При збої в мережі або оновленні сторінки повідомлення мають зберігатися у черзі або локально (залежно від реалізації), щоб уникнути втрати даних;

г) сумісність

Підтримка всіх основних браузерів і пристроїв. Чат повинен відображатись однаково коректно на різних розмірах екранів;

д) адаптивність інтерфейсу

Інтерфейс має автоматично пристосовуватися до ширини вікна браузера без втрати функціональності.

A.5 Вимоги до програмної документації

Програмна документація повинна бути складена відповідно до встановлених стандартів і містити повну інформацію, необхідну для розуміння, встановлення, використання, супроводу та модернізації вебчату.

Документація повинна включати:

- загальний опис системи;
- інструкції для користувача;
- технічний опис;
- інструкції з розгортання та налаштування;
- опис структури бази даних;
- перелік функціональних можливостей;
- результати тестування.

Документація має бути зрозумілою, структурованою, без технічних помилок, складеною українською мовою, з чітким поділом на розділи та підрозділи.

A.6 Вимоги до складу та параметрів технічних засобів

Для коректної роботи вебчату необхідна наявність наступного технічного забезпечення:

Клієнтська частина (користувач):

- пристрій з доступом до Інтернету (ПК, ноутбук, смартфон або планшет);
- встановлений сучасний веббраузер (Google Chrome, Mozilla Firefox, Microsoft Edge, Opera);
- роздільна здатність екрана не менше 360×640 пікселів;

Серверна частина (розробник/адміністратор):

- хостинг або локальний сервер, що підтримує обрані технології (наприклад, Node.js, MongoDB);
- мінімальні характеристики: процесор від 2 ядер, вільне місце на диску від 5 ГБ;
- підключення до Інтернету з високою стабільністю та швидкістю.

A.7 Порядок контролю та приймання

Контроль якості та приймання програмного забезпечення здійснюється з метою перевірки його відповідності заданим функціональним і нефункціональним вимогам.

Перед передачею вебчату до експлуатації має бути проведене тестування, результати якого документуються. Усі знайдені помилки мають бути усунуті до моменту приймання.

Прийняття вважається успішним, якщо вебчат:

- забезпечує стабільну роботу всіх основних функцій;
- не містить критичних помилок;
- відповідає заявленим технічним характеристикам;
- має супровідну документацію.

A.7.1 Процедура приймання

а) підготовчий етап

На цьому етапі проводиться інсталяція вебчату в тестовому середовищі або на демонстраційному сервері. Перевіряється відповідність технічних умов: конфігурація сервера, доступ до бази даних, стабільність мережевого з'єднання. Забезпечується наявність усіх необхідних компонентів для запуску системи;

б) ознайомлення з документацією

Комісія з приймання або представники замовника перевіряють комплект документації: пояснювальна записка, інструкції користувача, технічний опис, результати тестування, чек-листи, тощо. Документація має бути зрозумілою, повною, без суперечностей;

в) проведення приймальних випробувань

На цьому етапі здійснюється практична перевірка вебчату за допомогою спеціально підготовлених сценаріїв тестування. Випробування охоплюють:

- функціональне тестування: перевірка роботи авторизації, реєстрації, створення чатів, надсилання та отримання повідомлень;
- перевірка надійності та стабільності: оцінка реакції системи на неправильні або неочікувані дії користувача, стійкість до збоїв;
- перевірка швидкодії: вимірювання часу відповіді на запити, швидкість оновлення повідомлень;
- перевірка адаптивності: тестування відображення вебчату на різних пристроях та у різних браузерах;
- перевірка безпеки: перевірка збереження паролів у зашифрованому вигляді, відсутність прямого доступу до конфіденційних даних через URL-запити;

г) усунення виявлених зауважень

У разі виявлення помилок або невідповідностей вони фіксуються в протоколі. Розробник зобов'язаний усунути їх у погоджений термін. Повторне тестування проводиться після внесення змін;

д) формування та підписання акту приймання-передачі

Після успішного проходження всіх етапів тестування складається акт приймання програмного продукту, який підписується відповідальними особами. Акт підтверджує готовність вебчату до дослідної або промислової експлуатації.

ДОДАТОК Б
Текст програми

B.1 App.tsx

```
import React, { useEffect } from "react"
import Router from "../components/Router"
import { useCookies } from "react-cookie"
import { CookiesTypes } from "../types/CookieTypes"
import { useAuth, useChats } from "../context"
import ChatService from "../services/chat-service"

function App() {
  const [cookies] = useCookies([CookiesTypes.USER])
  const { setUser } = useAuth()
  const { setChats, setIsChatLoading } = useChats()

  useEffect(() => {
    if (cookies?.user) {
      setUser(cookies?.user)
      setIsChatLoading(true)
      ChatService.fetchChats({
        user_id: cookies?.user?.id as unknown as number,
      })
        .then((res) => {
          setChats(res.data.chats)
          setIsChatLoading(false)
        })
        .finally(() => setIsChatLoading(false))
    }
  }, [])

  return (
    <div className=" style={{ height: "100vh" }}>
      <Router />
    </div>
  )
}

export default App
```

B.2 Index.tsx

```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";
import { BrowserRouter } from "react-router-dom";
import { CookiesProvider } from "react-cookie";
import { ErrorBoundary } from "react-error-boundary";
import { MainContext } from "./context";

const root = ReactDOM.createRoot(
  document.getElementById("root") as HTMLElement
);
root.render(
  <React.StrictMode>
    <ErrorBoundary fallback={<div>Something went wrong</div>}>
      <BrowserRouter>
        <CookiesProvider>
          <MainContext>
            <App />
          </MainContext>
        </CookiesProvider>
      </BrowserRouter>
    </ErrorBoundary>
  </React.StrictMode>
);
```

B.3 Context/index.tsx

```
import React, {
  createContext,
  useState,
  ReactNode,
  useContext,
  useEffect,
} from "react"
import { IUser } from "../types/UserTypes"
import { IChat } from "../types/ChatTypes"
import ChatService from "../services/chat-service"
import { IMessageType } from "../types/MessageType"
import { io, Socket } from "socket.io-client"
```

```

// export const MainContext = React.createContext(defaultValue);

const AuthContext = createContext<
  | {
    user: IUser | null
    setUser: React.Dispatch<React.SetStateAction<IUser | null>>
  }
  | undefined
>(undefined)

const AuthProvider = ({ children }: { children: ReactNode }) => {
  const [user, setUser] = useState<IUser | null>(null)

  return (
    <AuthContext.Provider value={{ user, setUser }}>
      {children}
    </AuthContext.Provider>
  )
}

const ChatContext = createContext<
  | {
    selectedChat: number | null
    setSelectedChat: (id: number) => void
    chats: IChat[]
    setChats: React.Dispatch<React.SetStateAction<IChat[]>>
    messages: IMessageType[]
    setMessages: React.Dispatch<React.SetStateAction<IMessageType[]>>
    sendMessage: (user_id: number, content: string) => Promise<void>
    isChatLoading: boolean
    setIsChatLoading: React.Dispatch<React.SetStateAction<boolean>>
    createChat: (user_ids: number[], admin_id: number) => Promise<void>
    socket: Socket
  }
  | undefined
>(undefined)

const ChatProvider = ({ children }: { children: ReactNode }) => {
  const [chats, setChats] = useState<IChat[]>([])
  const [selectedChat, setSelectedChat] = useState<number | null>(null)
  const [messages, setMessages] = useState<IMessageType[]>([])

  const [isChatLoading, setIsChatLoading] = useState<boolean>(false)

```

```

const socket = io("http://localhost:5001")

const sendMessage = async (user_id: number, content: string) => {
  socket.emit("sendMessage", selectedChat, content, user_id)
}

const createChat = async (user_ids: number[], admin_id: number) => {
  await ChatService.createChat({ user_ids, admin_id }).then((res) =>
    setChats((prev) => [...prev, res.data.chat])
  )
}

useEffect(() => {
  if (selectedChat) {
    socket.on("connect", () => {
      console.log("Connected to server:")
    })

    socket.emit("joinChat", selectedChat)

    socket.on("newMessage", (newMessage) => {
      setMessages((prevMessages) => [...prevMessages, newMessage])
    })

    ChatService.fetchChatMessages({ chat_id: selectedChat }).then((res) =>
      setMessages(res.data.messages)
    )

    return () => {
      socket.off("newMessage")
    }
  }
}, [selectedChat])

return (
  <ChatContext.Provider
    value={{
      chats,
      setChats,
      selectedChat,
      setSelectedChat: (id) =>
        setSelectedChat(() => {
          socket.on("disconnect", () => {})

```

```

        return id
      }),
      messages,
      setMessages,
      sendMessage,
      isChatLoading,
      setIsChatLoading,
      createChat,
      socket,
    })
  >
  {children}
</ChatContext.Provider>
)
}

export const MainContext = ({ children }: { children: ReactNode }) => (
  <AuthProvider>
    <ChatProvider>{children}</ChatProvider>
  </AuthProvider>
)

export const useAuth = () => useContext(AuthContext)!
export const useChats = () => useContext(ChatContext)!

```

B.4 services/auth-service.ts

```

import { AxiosResponse } from "axios"
import { AuthResponse } from "../types/UserTypes"
import $api from "../http"

class AuthService {
  static login = async ({
    username,
    password,
  }: {
    username: string
    password: string
  }): Promise<AxiosResponse<AuthResponse>> =>
    $api.post<AuthResponse>("/auth/login", { username, password })

  static register = async ({
    username,

```

```

    password,
  }: {
    username: string
    password: string
  }): Promise<AxiosResponse<AuthResponse>> => {
    return $api.post<AuthResponse>("/auth/register", {
      username,
      password,
    })
  }
}

static logout = async (): Promise<void> => $api.post("/auth/logout")
}

export default AuthService

```

B.5 services/chat-service.ts

```

import { AxiosResponse } from "axios"
import $api from "../http"
import { IMessageType } from "../types/MessageType"
import { IChat } from "../types/ChatTypes"
import { IUser } from "../types/UserTypes"

class ChatService {
  static fetchChatMessages = async ({
    chat_id,
  }): {
    chat_id: number
  }): Promise<AxiosResponse<{ messages: IMessageType[] }>> => {
    return $api.get<{ messages: IMessageType[] }>(
      `/chat/messages?chat_id=${chat_id}`
    )
  }

  static sendChatMessages = async ({
    user_id,
    content,
    chat_id,
  }): {
    user_id: number
    content: string
    chat_id: number
  }

```

```

}): Promise<AxiosResponse<{ message: IMessageType }>> => {
  return $api.post<{ message: IMessageType }>(`/chat/messages`, {
    user_id,
    content,
    chat_id,
  })
}

```

```

static fetchChats = async ({
  user_id,
  search_string,
}): {
  user_id: number
  search_string?: string
}): Promise<AxiosResponse<{ chats: IChat[] }>> => {
  return $api.get<{ chats: IChat[] }>(
    `/chat?user_id=${user_id}&search_string=${search_string || ""}`
  )
}

```

```

static fetchNewChats = async ({
  search_string,
  user_ids,
}): {
  search_string?: string
  user_ids: number[]
}): Promise<AxiosResponse<{ chats: IChat[]; users: IUser[] }>> => {
  return $api.get<{ chats: IChat[]; users: IUser[] }>(
    `/chat/new?search_string=${search_string || ""}${user_ids
      .map((id) => `&user_ids=${id}`)
      .join("")}`
  )
}

```

```

static createChat = async ({
  user_ids,
  admin_id,
}): {
  user_ids: number[]
  admin_id: number
}): Promise<AxiosResponse<{ chat: IChat }>> => {
  return $api.post<{ chat: IChat }>(`/chat`, {
    user_ids,
    admin_id,
  })
}

```

```
    })  
  }  
}
```

```
export default ChatService
```

ДОДАТОК В
Слайди презентації

Національний університет “Запорізька Політехніка”
кафедра програмних засобів

Випускна робота бакалавра
на тему: **Програмна реалізація вебчату**

Виконав:
студент групи КНТ-151
Левченко Ярослав Дмитрович

Керівник дипломного проєкту:
к.т.н.
Федорончак Тетяна Василівна

1

Рисунок В.1 – Слайд №1

Об’єкт, предмет та мета роботи

- Об’єкт дослідження – процес комунікації користувачів у реальному часі в межах вебсередовища.
- Предмет дослідження – вебзастосунок для миттєвого обміну повідомленнями (вебчат), його архітектура та функціональні компоненти.
- Мета розробки – дослідити принципи побудови вебчатів та розробити програмне рішення для онлайн-комунікації з використанням сучасних технологій.

2

Рисунок В.2 – Слайд №2

Постановка завдання

- Необхідно розробити вебзастосунок для онлайн-комунікації (вебчат), який забезпечить миттєвий обмін повідомленнями між користувачами в реальному часі. У процесі розробки потрібно дослідити архітектурні підходи до побудови вебчатів, обґрунтувати вибір технологій, реалізувати серверну та клієнтську частини застосунку, забезпечити збереження повідомлень і користувацьких даних у базі даних, а також протестувати систему на предмет стабільності та працездатності.

3

Рисунок В.3 – Слайд №3

Аналіз предметної області

- Призначення вебчатів: вебчати призначені для забезпечення миттєвої комунікації між користувачами через інтернет. Вони дозволяють обмінюватися текстовими повідомленнями, файлами, а в деяких випадках – мультимедійним контентом у режимі реального часу.
- Сфери застосування: соціальні мережі та месенджери, онлайн-магазини та служби підтримки, освітні платформи, корпоративні інструменти для внутрішнього спілкування.

4

Рисунок В.4 – Слайд №4

Порівняння аналогів вебчату



Функціональність	Slack	Discord	Telegram	Розробляючий вебчат
Кастомізація інтерфейсу	–	–	–	+
Безпека / шифрування	–	–	+	+
Підходить для бізнесу	+	–	–	+
Ролі, доступи, модерація	–	+	–	+

5

Рисунок В.5 – Слайд №5

Порівняння середовищ розробки



Критерії	WebStorm	Atom	Sublime Text	VS Code
Легкість і швидкість	–	+	+	+
Інтеграція з фреймворками	+	+	–	+
Підтримка плагінів	+	+	–	+
Підтримка Git	+	–	+	+
Інтеграція з Docker	–	–	–	+

6

Рисунок В.6 – Слайд №6

Порівняння архітектур проєктування



Критерії	MVVM	Microservices	Monolithic	MVC
Масштабованість	+	+	–	+
Простота реалізації	+	–	+	+
Гнучкість	+	+	–	+
Інтерактивність інтерфейсу	+	+	–	+
Використання ресурсів	+	–	–	+

7

Рисунок В.7 – Слайд №7

Порівняння баз даних



Критерії	PostgreSQL	MongoDB	SQLite	MySQL
Реляційна модель	+	–	+	+
Підтримка ACID	+	–	+	+
Швидкість запису	–	+	–	+
Простота використання	–	+	+	+
Підтримка великих даних	+	+	–	+

8

Рисунок В.8 – Слайд №8

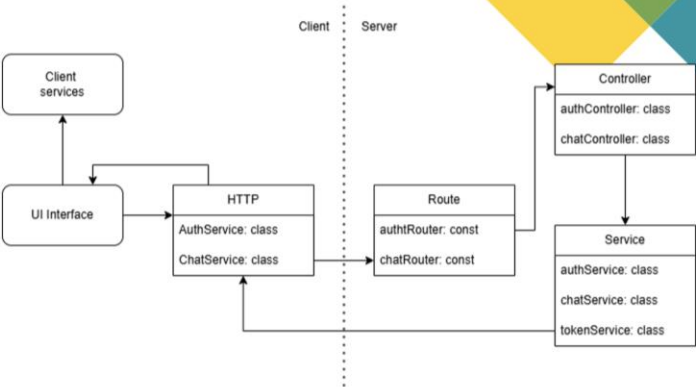
Порівняння мов програмування

Критерії	Python	PHP	C#	JavaScript
Робота на клієнті	–	–	–	+
Простота вивчення	+	+	–	+
Швидкість розробки	+	–	–	+
Підтримка сучасних UI	–	–	+	+

9

Рисунок В.9 – Слайд №9

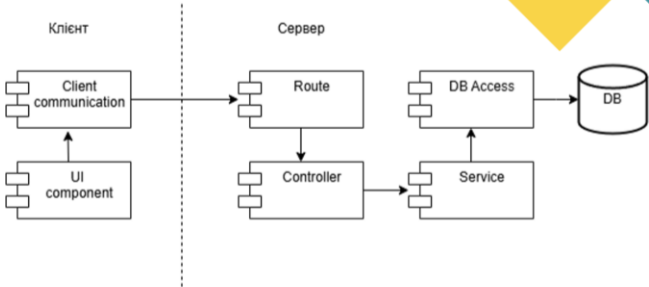
Структурна схема вебчату



10

Рисунок В.10 – Слайд №10

Діаграма компонентів вебчату



11

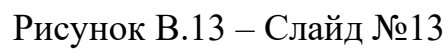
Рисунок В.11 – Слайд №11

Схема функціонування вебчату



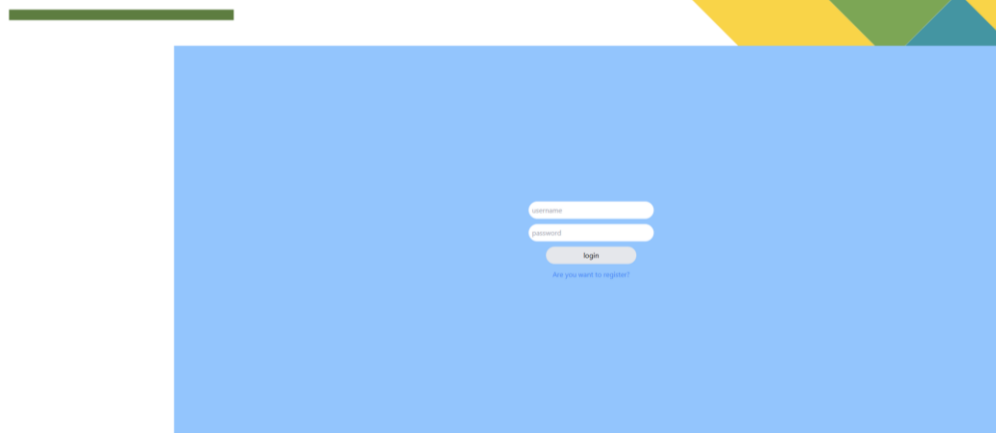
12

Рисунок В.12 – Слайд №12





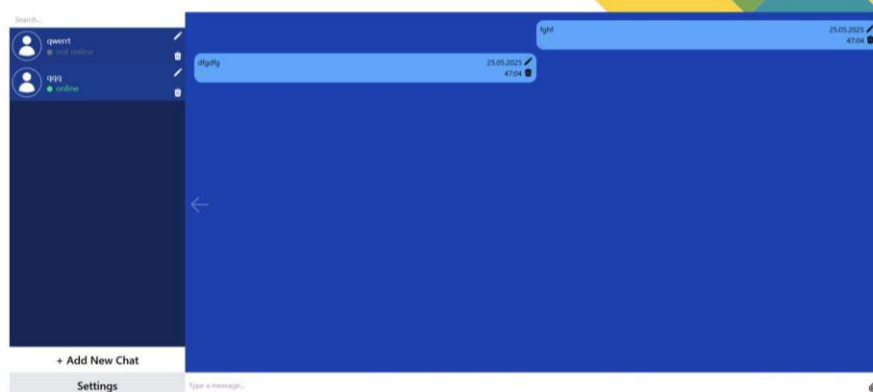
Сторінка авторизації вебчату



15

Рисунок В.15 – Слайд №15

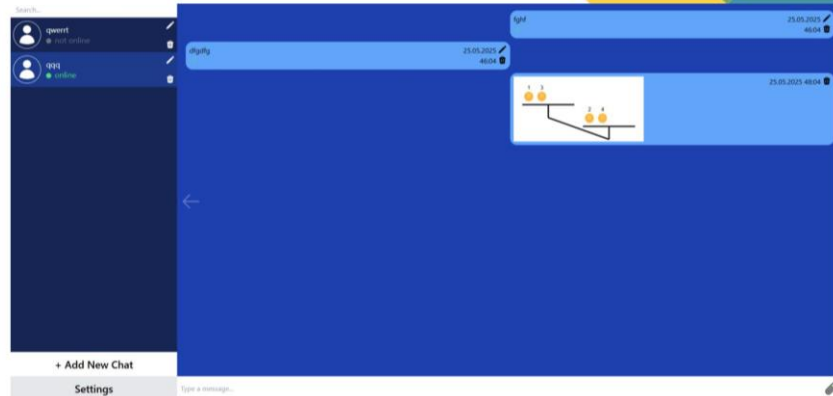
Сторінка основного інтерфейсу вебчату



16

Рисунок В.16 – Слайд №16

Сторінка обміну повідомленнями вебчату



17

Рисунок В.17 – Слайд №17

Висновки



Під час виконання дипломної роботи було досягнуто поставленої мети та вирішено такі задачі:

- проведено аналіз предметної області та сучасних вебчатів; вивчено архітектурні підходи (MVC, MVVM, мікросервіси, моноліт);
- проаналізовано популярні технології (JavaScript, Node.js, WebSocket, MySQL);
- розроблено структуру бази даних і логіку взаємодії клієнта з сервером;
- здійснено розробку користувацького інтерфейсу з урахуванням сучасних вимог UX/UI;
- протестовано функціональність чату та перевірено стабільність його роботи.

18

Рисунок В.18 – Слайд №18