**Міністерство освіти і науки України**
**Тернопільський національний технічний університет імені Івана Пулюя**

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету )
Кафедра комп'ютерних наук
(повна назва кафедри)

# КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка соціальної мережі засобами MongoDB та Node.js

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Галюк М. В.
(підпис) (прізвище та ініціали)

Керівник _____ Струтинська І. В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Осухівська Г.М.
(підпис) (прізвище та ініціали)

Тернопіль
2022

Міністерство освіти і науки України
**Тернопільський національний технічний університет імені Івана Пулюя**

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2022 р.

# ЗАВДАННЯ
## НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Галюку Миколі Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка соціальної мережі засобами MongoDB та Node.js

Керівник роботи доктор економічних наук, професор кафедри КН Струтинська Ірина Володимирівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 13 » березня 2022 року № 4/7-161
2. Термін подання студентом завершеної роботи 20 червня 2022р.
3. Вихідні дані до роботи Літературні та інтернет джерела щодо створення соціально мережі засобами MongoDB та Node.js

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Постановка задачі та формування вимог до соціальної мережі. 1.1 Історія розвитку соціальних мереж. 1.2 Аналіз предметної області. 1.3 Формування вимог до соціальної мережі 1.4 Пошук актантів та варіантів використання. 1.5 Пошук варіантів використання. 1.6 Вибір середовища розробки. 1.7 Висновки до першого розділку. 2 Проектування соціальної мережі 2.1 Обґрунтування розробленої структури соціальної мережі. 2.2 Проектування таблиць БД 2.3 Розробка інтерфейсу користувача та його верстка. 2.4 Розробка серверної частини. 2.5 Розробка месенджера з використання Socket.io. 2.6 Висновок до другого розділу. 3. Безпека життєдіяльності, основи охорони праці. 3.1 Долікарська допомога при ураженні електричним струмом. 3.2 Вимоги ергономіки до організації робочого місця оператора ПК. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Безпека життєдіяльності, основи хорони праці | Гурик О.Я., доцент кафедри МТ | | |

7. Дата видачі завдання 24 січня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Ознайомлення з завданням до кваліфікаційної роботи | 24.01.2022 | *Виконано* |
| 2. | Підбір джерел про створення соціальних мереж | 04.01.2022-30.01.2022 | *Виконано* |
| 3. | Переклад та опрацювання джерел про створення | 31.01.2022-06.02.2022 | *Виконано* |
| | Соціальних мереж | | |
| 4. | Виконання дослідження щодо створення соціальної | 07.02.2022-13.02.2022 | *Виконано* |
| | мережі. Розроблення соціально мережі засобами | | |
| | MongoDB та Node.js | | |
| 5. | Оформлення розділу «Постановка задачі та | 14.02.2022-06.03.2022 | *Виконано* |
| | формування вимог до соціальної мережі» | | |
| 6. | Оформлення розділу «Проектування соціальної | 07.03.2022-03.04.2022 | *Виконано* |
| | мережі» | | |
| 7. | Виконання завдання до підрозділу «Безпека життєдіяльності» | 04.04.2022-17.04.2022 | *Виконано* |
| 8. | Виконання завдання до підрозділу «Основи охорони | 18.04.2022-01.05.2022 | *Виконано* |
| | праці» | | |
| 9. | Оформлення кваліфікаційної роботи | 02.05.2022-15.05.2022 | *Виконано* |
| 10. | Нормоконтроль | 16.05.2022-22.05.2022 | *Виконано* |
| 11. | Перевірка на плагіат | 06.06.2022 | *Виконано* |
| 12. | Попередній захист кваліфікаційної роботи | 07.06.2022 | *Виконано* |
| 13. | Захист кваліфікаційної роботи | 20.06.2022 | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Студент _____ Галюк М. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Струтинська І. В.
(підпис) (прізвище та ініціали)

# АНОТАЦІЯ

Розробка соціальної мережі засобами MongoDB та Node.js // Кваліфікаційна робота освітнього рівня «Бакалавр» // Галюк Микола Васильович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2022 // С. 48 , рис. 24, табл. 5, додат. 4, бібліогр. 35.

**Ключові слова:** бази даних, соціальна мережа, програмування MongoDB, Node.js, React, Express.

Кваліфікаційна робота присвячена розробці соціальної мережі із використанням серверної платформи Node.js.

Мета даної роботи полягає в створенні мережі для спілкування, розміщення інформації та шифруванням повідомлень.

В першому розділі кваліфікаційної роботи проведено аналіз обраної предметної області, сформовано перелік вимог до соціальної мережі, розроблено діаграми використання, аргументовано вибір середовища та основних технологій розробки.

В другому розділі кваліфікаційної роботи описано проектування БД для соціальної мережі, проведено моделювання робочої архітектури, описано розробку як серверної, так і клієнтської частин застосунку, вказано основні особливості даного процесу, проведено тестування розробленого функціоналу.

Об'єктом дослідження є сучасна соціальна мережа та нереляційна БД MongoDB.

Предметом дослідження є засоби і методи розробки веб-застосунків для соціальної мережі із використанням серверної платформи Node.js.

# ANNOTATION

Social network development using MongoDB and Node.js // Qualification work of educational level "Bachelor" // Haliuk Mykola // Ternopil Ivan Pul'uj National Technical University, Department of Computer Information Systems and Software Engineering, Department of Computer Science, group CH-41 // Ternopil, 2022 // P. – 48, Fig. – 24, Table. – 5, Annexes. – 4, References. - 35.

Keywords: databases, social network, MongoDB programming, Node.js, React, Express.

Qualification work is devoted to the development of a social network using the server platform Node.js.

The purpose of this work is to create a network for communication, posting information and encrypting messages.

In the first section of the qualification work the analysis of the chosen subject area is carried out, the list of requirements to a social network is formed, diagrams of use are developed, the basic methods of encryption of messages are investigated, the choice of environment and basic technologies of development is argued.

The second section of the qualification work describes the design of the database for the web application, modeling the working architecture, describes the development of both server and client parts of the application, the main features of this process, testing the developed functionality.

The object of research is a modern social network and non-relational database MongoDB.

The subject of research is the tools and methods of developing web applications for social networks using the server platform Node.js.

# ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interfaice.

AJAX – Asynchronous JavaScript And XML.

JSON – JavaScript Object Notation.

SAAS – Software as a service.

XML – Extensible Markup Language.

UI – User Interfaice – інтерфейс користувача.

БД – база даних.

IDE – Integrated Drive Electronics.

UML – Unified Modeling Language.

CRUD – Create Read Update Delete.

MVC – Model-View-Controller.

REST – Representational State Transfer.

CORS – Cross-Origin Resource Sharing.

**ЗМІСТ**

# ВСТУП

**Актуальність теми.** Інтернет – це мережа, яка швидко розвивається. Одне з найпопулярніших місць в Інтернеті - соціальні мережі. Вони слугують місцем для спілкування людей зі своїми близькими, допомагають розширити комунікацію та встановити зв’язки з людьми по всьому світу. Можна розрізняти соціальні мережі як платформи для ведення блогів, відеоблогів, подкастингів тощо [1]. Соціальні мережі використовуються для різних цілей. Вони зараз всюди. Бізнес отримує велику користь від ведення сторінок у соціальних мережах. Підприємці використовують їх для пошуку і спілкування зі своїми потенційними клієнтами та бізнес-партнерами. Також люди шукають роботу, використовуючи сторінки для кращого зв’язку з роботодавцями. Це дає їм великі можливості для пошуку кращої роботи [2].

**Мета і задачі дослідження.** Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є створення безпечної соціальної мережі, яка дозволить привернути увагу великої кількості користувачів та подальшої популяризації за допомогою розроблюваного веб-застосунку на базі серверної платформи Node.js. Для успішного досягнення поставленої задачі необхідно:

- проаналізувати обрану предметну область;
- розробити моделі сутностей для зберігання даних та спроектувати базу даних;
- спроектувати робочу архітектуру соціальної мережі;
- розробити соціальну мережу відповідно до поставлених вимог;
- провести повне тестування усіх функціональних можливостей соціальної мережі.

**Практичне значення одержаних результатів.** Розроблена в ході виконання поставлених задач соціальна мережа буде зручною у використанні і багатою на функціонал, який дозволить утримувати велику кількість активних користувачів.

# РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ ТА ФОРМУВАННЯ ВИМОГ ДО СОЦІАЛЬНОЇ МЕРЕЖІ

## 1.1 Історія розвитку соціальних мереж

Можна подумати, що соціальна мережа це щось нове, але якщо подивитися на історію розвитку, ми можемо побачити що вона сягає минулого століття. Немає точної дати чи року коли почався початок впровадження соціальних мереж. Однак можна зауважити, що схожий функціонал почав з'являтися в 90-х роках [3].

Перший крок для створення соціальної мережі відбувся у 1971 році. Рей Томлінсон здійснив перше успішне відправлення електронного листа через два комп'ютери, які були розміщені поруч .

У 1991 році Тім Бернерс Лі створив мову розмітки гіпертексту HTML, один із складових елементів веб-сайту, щоб створити перегляд веб-сторінок простим і доступним. Після цього, у 1994 році, перший раз запущено прототип соціальної мережі GeoCities.

У 1995 році побачив світ TheGlobe.com. Ця соціальна мережа вперше надала можливість користувачам публікувати свій контент та взаємодіяти з іншими користувачами, які мають схожі інтереси.

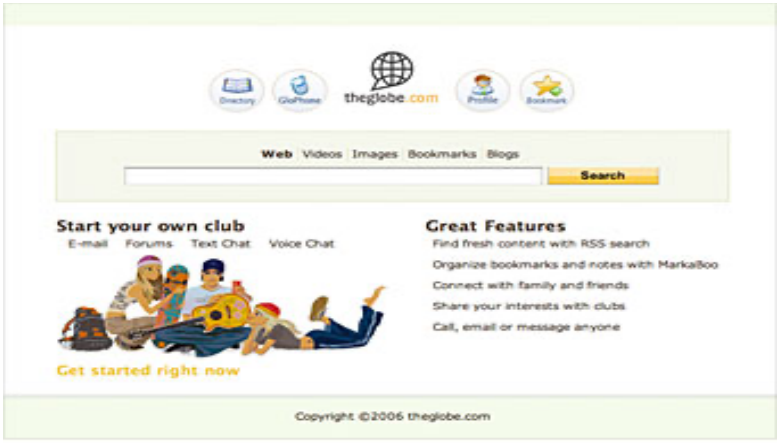


Рисунок 1.1 – Інтерфейс веб-сайту TheGlobe.com

У 2002 році вперше був запущений Friendster з метою познайомити користувачів з людьми із різних кіл друзів. Це допомогло створити довіру між користувачами, тому що вони були знайомі їхнім друзям. Friendster виявився вдалим проектом і за кілька місяців досяг трьох мільйонів користувачів.

Рисунок 1.2 – Інтерфейс соціальної мережі Friendster

Після створення Friendster через рік, була започаткована нова соціальна мережа – MySpace. Вона включала більше інтерактивних елементів, ніж її попередники. MySpace дає можливість керувати від особистих профілів до хостингу фотографій.

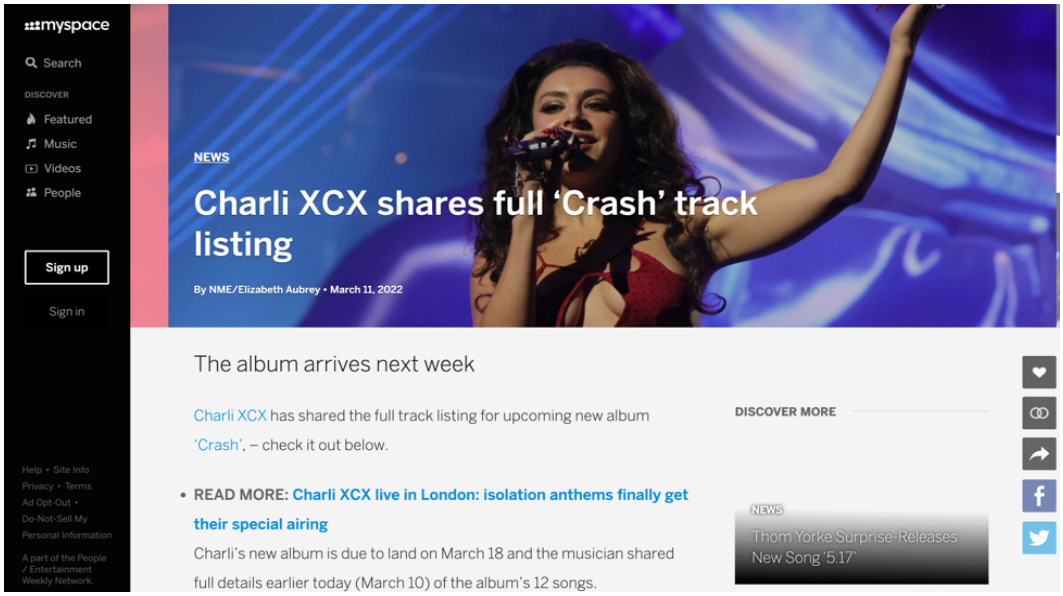

Рисунок 1.3 – Сучасний інтерфейс соціальної мережі Myspace

У 2004 році була започаткована одна з найпопулярніших мереж Facebook. Вона досить швидко стала популярнішою ніж Myspace за кількістю відвідувань у місяць, тому Myspace змінила стратегію і цільову аудиторію, ставши розважальною платформою.

Facebook, який створений Марком Цукербергом, мав на меті згуртувати студентів Гарвардського університету. Більше половини студентів цього освітнього закладу передплатили підписку в перший місяць його роботи, а через рік він працював у 500-ах американських університетах із 2-ма мільйонами користувачами.

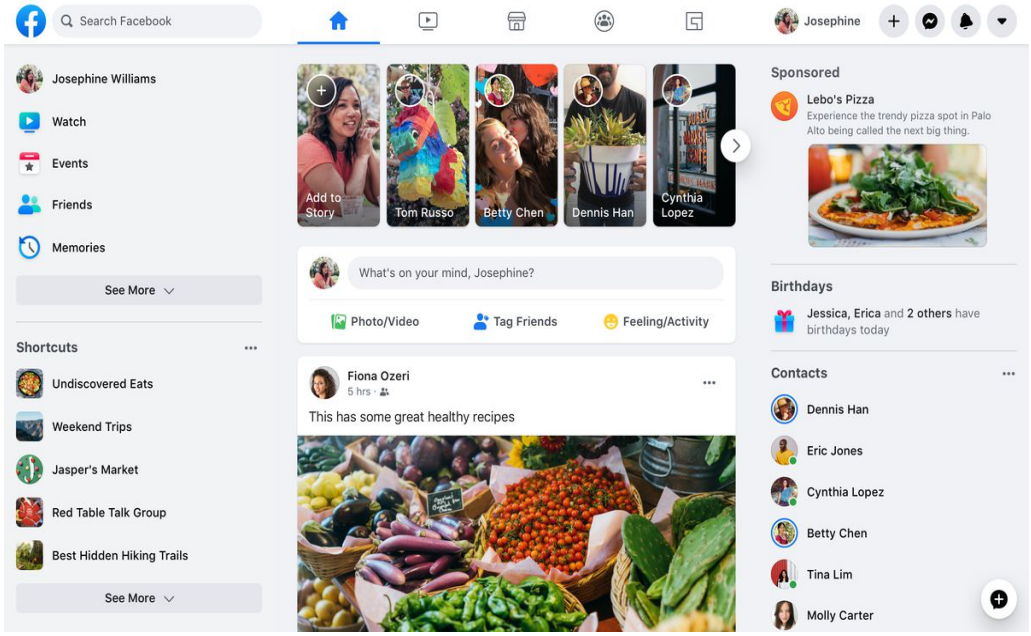


Рисунок 1.4 – Інтерфейс соціальної мережі Facebook

У 2006 році соціальна мережа Facebook стала доступною усім користувачам Інтернету. Це дуже сильно вплинуло на історію соціальних мереж. Підприємці побачили гарний спосіб, як рекламувати свої послуги і збільшити аудиторію.

У 2006 році побачив світ Twitter. Він був інструментом для того, щоб повідомляти новини, змінивши спосіб перегляду телебачення або, навіть, управління революціями.

Twitter став чудовою ідею для ведення мікроблогів. У сьогоденні не існує події, яка б не була написана у Twitter.

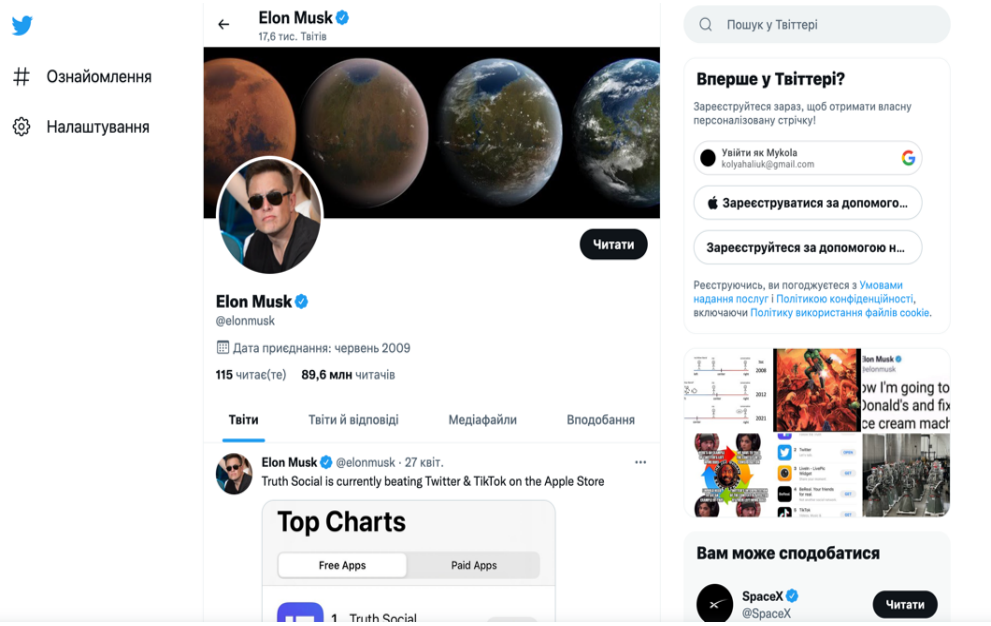


Рисунок 1.5 – Інтерфейс соціальної мережі Twitter

Останнім часом, почало з’являтися все більше соціальних мереж, які націлені на певну аудиторію з різними характеристиками [4]. За останні п’ять років появилися наступні популярні мережі:

Pinterest**.** Це соціальний фото-сервіс для поширення зображень між користувачами. Відвідувачів об`єднують навколо стрічки із пінами (зображення, завантажені на Pinterest) за певними критеріями, предметами, захопленнями або подіями.

Youtube. Платформа, яка дозволяє завантажувати і ділитися контентом у вигляді відео. Також тут розміщено мільйони музичних відео, відеоблоги або телешоу.

LinkedIn. Це спеціальна соціальна мережа, націлена на компанії. Користувач може створити профіль, де він розміщує інформацію про свій досвід, вміння та якості. У такий спосіб тут збираються мільйони фахівців з різних галузь.

Кожного року ми бачимо, як швидко розвиваються соціальні мережі. Їхня історія тільки починається, а вже спричинила тисячі змін у нашому житті. Змінюється також ставлення людей один до одного. Сьогодні це один з основних методів впливу і поширення інформації для широких мас населення.

## 1.2 Аналіз предметної області

Сучасний світ соціальних мереж наповнений набором послуг і функцій, які привертають увагу понад 5-ти мільярдів користувачів мобільних пристроїв у всьому світі.

З появою застосунків для соціальних мереж, які можуть працювати на смартфонах, кінцеві користувачі мають можливість «брати» свої спільноти з собою, куди захочуть. Підприємства скористалися перевагами цієї нової мобільності споживачів, обслуговуючи своїх клієнтів новими, простішими методи взаємодії - «нові способи покупки товарів і послуг» [5].

Соціальні мережі дають змогу покращувати розвиток і підтримувати особисті і ділові відносини за допомогою технологій. Це реалізується через такі веб-застосунки, як Twitter, Facebook та Instagram. Вони дозволяють людям комунікувати один з одним, розвиваючи та покращуючи стосунки, ділитися інформацією, ідеями та повідомленнями.

В основному, соціальні мережі є засобом для того, щоб залишатися на зв’язку з членами сім'ї і друзями. Вони мають змогу ділитися фотографіями та новинами їхнього життя. Користувачі також можуть спілкуватися з незнайомцями, в яких схожі інтереси. Люди можуть шукати один одного за допомогою списків, груп або ж хештегів [6].

Соціальні мережі дуже часто використовують маркетологи для збільшення кількості потенційних клієнтів і пізнаваності бренду. За рахунок збільшення обсягу аудиторії, компанія може збільшити свій прибуток, просувати свою мету, бренд та ідеї [7].

Наприклад, звичайний користувач Twitter може вперше дізнатися про компанію через рекламу у стрічці новин і вирішити все-таки скористатися їхньою послугую. Чим більше людей знають про бренд, тим більше у компанії шансів знайти та утримати нових клієнтів.

Також соціальні мережі слугують способом підвищення коефіцієнту конверсії. Вони надають великий спектр послуг для розміщення реклами саме для потенційних клієнтів. Створення підписки надає доступ до нових, недавніх і старих клієнтів та взаємодію з ними. Публікація у блозі компанії відео або зображень у соціальних мережах дозволяє читачам відреагувати, відвідати веб-сайт компанії та стати клієнтами.

## 1.3 Формування вимог до соціальної мережі

Не кожен застосунок для соціальних мереж, який виходить на ринок, стає популярним. Однак є кілька речей, які підштовхнуть його в правильному напрямку. Розробляючи концепцію нового застосунка для соціальних мереж, важливо уважно вивчити особливості сучасних найпопулярніших програм. Хоча всі вони мають характерні риси, між ними є певні спільні речі. Виділимо кілька ключових функцій кожної соціальної мережі:

Простий і дружній інтерфейс користувача (UI) [8] – одна з перших речей, яка повинна враховуватися під час процесу створення ідей для створення застосунку. Іншими словами - засоби, за допомогою яких користувачі взаємодіють із програмним забезпеченням. Інтерфейс програми містить кілька елементів, вміст і медіа-макет, елементи керування введенням, навігацію тощо. Незалежно від цільової аудиторії, додаток для соціальних мереж повинен мати простий для навігації інтерфейс користувача, який дозволить користувачам швидко та без проблем знайти те, що вони шукають.

Візуально привабливий і доступний дизайн. Безліч разів нові програми соціальних мереж не залучають користувачів, оскільки вони погано

розроблені. Вони можуть містити багато випадкових, суперечливих елементів, які погано з'єднуються і створюють сенсорне перевантаження для користувачів, не даючи їм бути доступними для всіх. Ця програма, як і будь-яка інша, для соціальних мереж повинна мати послідовний, доступний і безперебійний дизайн зі шрифтами та кольоровими схемами, які ретельно підібрані, щоб сприяти згуртованому та приємному досвіду користувача [9].

Безпечний вхід. Користувачі застосунків соціальних мереж діляться різноманітною інформацією публічно чи приватно. Через це їхня інформація має бути максимально захищеною, щоб запобігти атакам зловмисного програмного забезпечення або крадіжці особистих даних. Ключовою особливістю програм соціальних мереж є можливість створення унікального облікового запису користувача з персональними налаштуваннями входу та методами ідентифікації, які дозволяють надавати своє ім'я, контактну інформацію, місцезнаходження, професію та інші персональні дані в тій мірі, яку вони вибирають, а також кілька методів перевірки для входу, наприклад, резервна електронна пошта або автентифікація коду [10].

Елемент мережі. Вони не дарма називають це «соціальними мережами». Серед найбільш потребуючих функцій соціальних мереж є можливість створити особисту або професійну мережу. Вона може складатися з друзів, родини, колег або людей зі схожими інтересами. Вибір залишається за користувачем. Деякі люди створюють облікові записи для особистих мереж, а інші для ділових цілей, однак важливо переконатися, що програма дозволяє користувачам додавати один одного в свої мережі та стежити один за одним [11].

Метод обміну вмістом. Більшість користувачів застосунків соціальних мереж хочуть поділитися інформацією зі своєю мережею. Обмін вмістом є однією з найкращих функцій програми для соціальних мереж, оскільки він полегшує спілкування між людьми та дозволяє користувачам відчувати себе більш пов'язаними, навіть, коли вони знаходяться далеко один від одного.

Спільний доступ до вмісту може включати можливість публікувати та надсилати фотографії чи відео, а також можливість дозволяти користувачам коментувати те, що поширюється.

Система обміну повідомленнями [12]. Подібно до обміну вмістом, одна з головних функцій застосунків соціальних мереж полягає в тому, що вони дозволяють користувачам надсилати один одному публічні та приватні повідомлення. Багато програм соціальних мереж навіть пропонують користувачам брати участь у групових чатах та відео дзвінках. Поки користувачі підключені до мережі WiFi, служби обміну повідомленнями не впливатимуть на тарифні плани користувача. Це набагато простіший і доступніший спосіб залишатися на зв'язку з людьми, аніж міжміські дзвінки чи дорогі текстові повідомлення. Очікується, що завдяки технологіям 5G, ці функції застосунку для соціальних мереж покращяться в найближчі роки.

Універсальність і чуйність. Серед головних функцій застосунків для соціальних мереж є можливість для користувачів отримувати доступ до платформи через настільні та мобільні пристрої, а інтерфейс користувача може адаптуватися та реагувати на різні пристрої та розміри екрана, не втрачаючи жодних елементів і не погіршуючи якість [13]. Різниці в тому, що користувачі бачать, коли вони мають доступ до вашої соціальної програми на своєму планшеті та смартфоном чи ноутбуком, не повинно бути майже ніякої.

### 1.4 Пошук актантів та варіантів використання

Кожний застосунок повинен мати певний перелік типів користувачів, яким буде доступно виконувати ті чи інші дії. Користувач може взаємодіяти із застосунком за допомогою певного переліку дій або кроків відповідно до рівня доступу. Сценарії взаємодії для кожного типу користувача із застосунком має назву – варіант використання або більш поширене поняття як Use Case [14]. Для того, щоб спроектувати деякі варіанти використання програмного

продукту потрібно виділити акторів (користувачі, які будуть взаємодіють із даним застосунком). У соціальній мережі можна виділити два актори – це незареєстрований та зареєстрований користувач. З цього випливає наступна діаграма варіантів використання даного веб-застосунку (див. рисунок 1.6).


Рисунок 1.6 – Діаграма варіантів використання

Розроблювана соціальна мережа повинен містити функціонал по реєстрації користувачів, шифруванні паролів, взаємодії із іншими користувачами, перегляду своїх та дописів друзів, редагуванню профілю та інше. Опис варіантів використання наведено в таблиці 1.1.

Таблиця 1.1 – Таблиця варіантів використання

| Актор | Найменування | Формулювання |
|---|---|---|
| Незареєстрований користувач | Вхід | Перевіряє валідність ведення авторизаційних даних |
| | Реєстрація | Перевіряє правильність заповнення реєстраційної форми і створення нового користувача |
| Зареєстрований користувач | Стежити за користувачем | Дозволяє підписатися на користувача і бачити його дописи в стрічці |
| | Відписатися від користувача | Дозволяє відписатися від користувача |
| | Надсилати повідомлення | Дозволяє надсилати повідомлення іншим користувачам |
| | Переглядати стрічку | Дозволяє переглядати стрічку з дописами користувачів |
| | Створити допис | Дозволяє створювати дописи |
| | Створити час | Дозволяє починати чат з іншими користувачами |

## 1.5 Пошук варіантів використання

Зазвичай соціальна мережа складається з складних та динамічних структур. На їх розробку виділяють не багато часу, щоб застосунок якнайшвидше запрацював і давав певний результат. Часто велика кількість розробників починають написання код без обміркувань, як і що вони збираються розробляти. У такому випадку серверні команди пишуться стихійно, а сутності створюються по мірі необхідності, як результат, архітектура такої системи розвивається в непередбачуваному напрямку. Але якщо провести моделювання і вибрати один підхід до написання коду, це призведи до більш продуктивного результату і така система буде гарантувати, що створений проект буде просто надалі підтримувати. Для цього створимо UML діаграму [15] взаємодії користувача з соціальною мережею, яка наведена на рисунку 1.7.

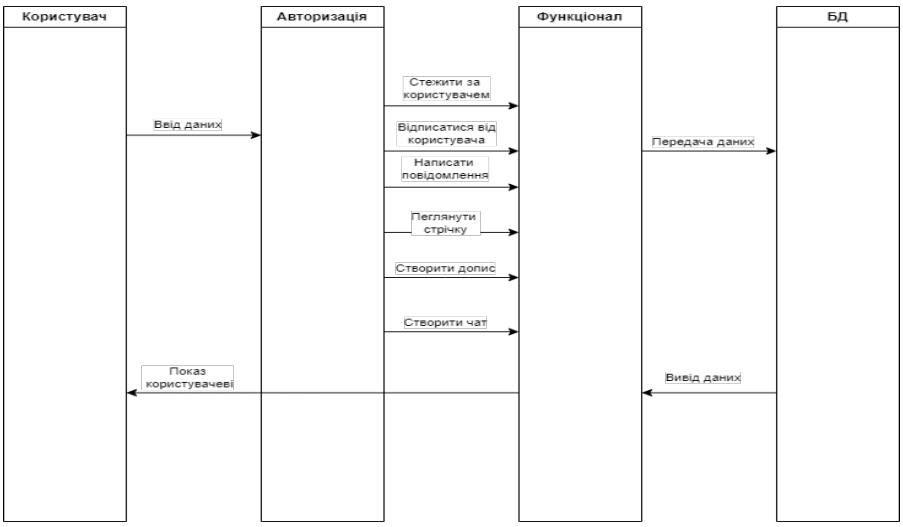


Рисунок 1.7 – UML діаграма взаємодії користувача з соціальною мережею

## 1.6 Вибір середовища розробки

Основний вплив на процес розробки будь-якого застосунку і проектування архітектуру має вибір стеку технологій, які будуть використовуватися в ході імплементації [16].

Процес розробки соціальної мережі включає наступні етапи:

- створення API;
- розробка макету;
- верстка шаблону;
- створення інтерактивних елементів.

Виходячи із зазначених вимог і функціоналу, який повинен бути впроваджений під час розробки, було взято наступний список технологій:

- Node.js;
- MongoDB;
- Express.js;
- React.

У даній роботі буде використовуватися MERN – це набір технологій, які реалізуються мовою Javascript на усіх рівня: інтерфейс користувача – React, а серверна частина – MongoDB, Express.js, Node.js.

MongoDB – це нереаляційна база даних, у якій дані зберігаються у форматі ключ – значення [17]. Є популярним рішенням у сфері веб-розробки.

Express.js – це фреймворк, який містить у собі набір інструментів і засобів, які пришвидшують серверну розробку [18].

Express.js надає багато готових рішень, які пришвидшують розробку сервера та бізнес логіки, наприклад, обробка надісланих форм, робота з куками, Cors-origin resource sharing (CORS).

React – це бібліотека для створення користувацького інтерфейсу веб-застосунків фреймворк. Він реалізований на мові програмування Javascript та вирішує проблеми динамічного оновлення частин вмісту веб-сторінки [19].

Node.js – це популярне рішення для розробки серверної частини, що надає змогу обробляти запити клієнта за певною бізнес логікою та повертати користувачу результат [20].

Node.js реалізована мовою JavaScript, що дозволяє створювати продуктивні серверні застосунки. Node.js використовує платформу V8, яка розроблена компанією Google, в якій реалізовані принципи асинхронного програмування, що дозволяє забезпечити викосу швидкість API [21].

## 1.7 Висновок до першого розділу

В першому розділі було розглянуто історію встановлення соціальних мереж і їхню роль у нашому повсякденному житті. Також, проаналізовано основні аспекти даного типу веб-застосунків і сформовано список вимог. Вибрано та аргументовано середовище розробки і основні технології для реалізації, зроблено пошук ключових акторів та сформовано діаграми варіантів використань, а також зображено структуру соціальної мережі на базі трирівневої архітектури.

# РОЗДІЛ 2. ПРОЕКТУВАННЯ СОЦІАЛЬНОЇ МЕРЕЖІ

## 2.1 Обґрунтування розробленої структури соціальної мережі

У логічній структурі проекту використано клієнт–серверну архітектуру, з використанням існуючих SAAS, таких як MongoDB Atlas. Це хмарна база даних від розробників MongoDB. Дана система спрощує розгортання та керування даними програмами, що дозволяє створювати стійкі і продуктивні застосунки [22]. Інтерфейс даної СУБД зображено на рисунку 2.1.

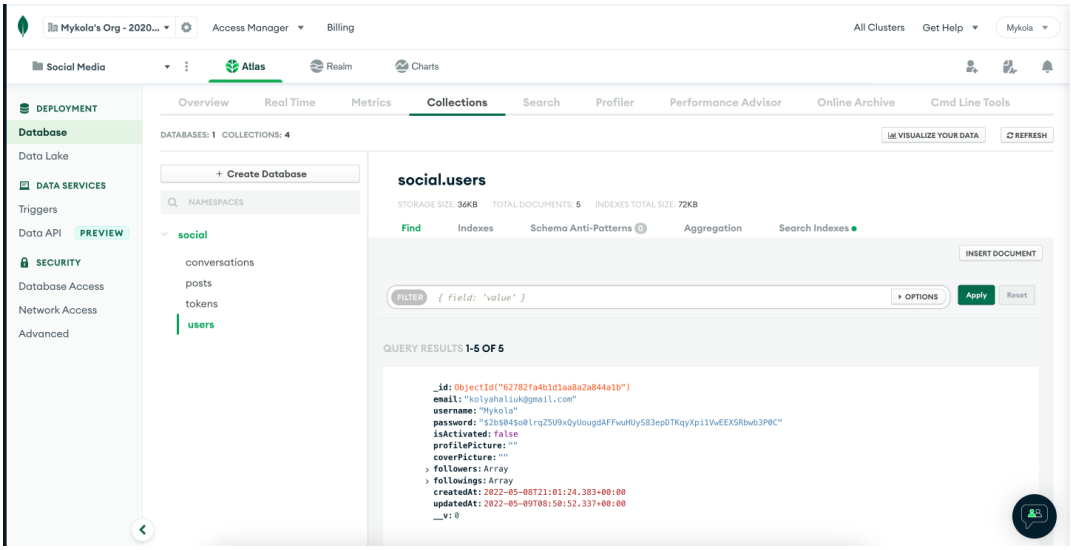


Рисунок 2.1 – Інтерфейс СУБД MongoDB Atlas

Вище зазначена архітектура є популярним шаблоном програмного забезпечення і є основною концепцією для створення веб-застосунків, яка включає взаємодію та обмін даних [23]. Вона передбачає наступні елементи:

- сервери, які надають певні послуги програмам, які звертаються до них;
- клієнти, що користуються сервісами, які надають сервери;
- мережа, що надає взаємодію між сервером і клієнтом.

Кожний сервер і клієнт працює паралельно і незалежно від інших. Немає значення до якого сервера прив’язаний клієнт. Кожний сервер може обробляти запити кількох різних клієнтів.

Варто зазначити, що клієнт і сервер це програмні модулі, які знаходяться на різних комп’ютерах. На рисунку 2.2 зображено особливості взаємодії між візуальним інтерфейсом застосунку, який працює у одному із популярних браузерів.

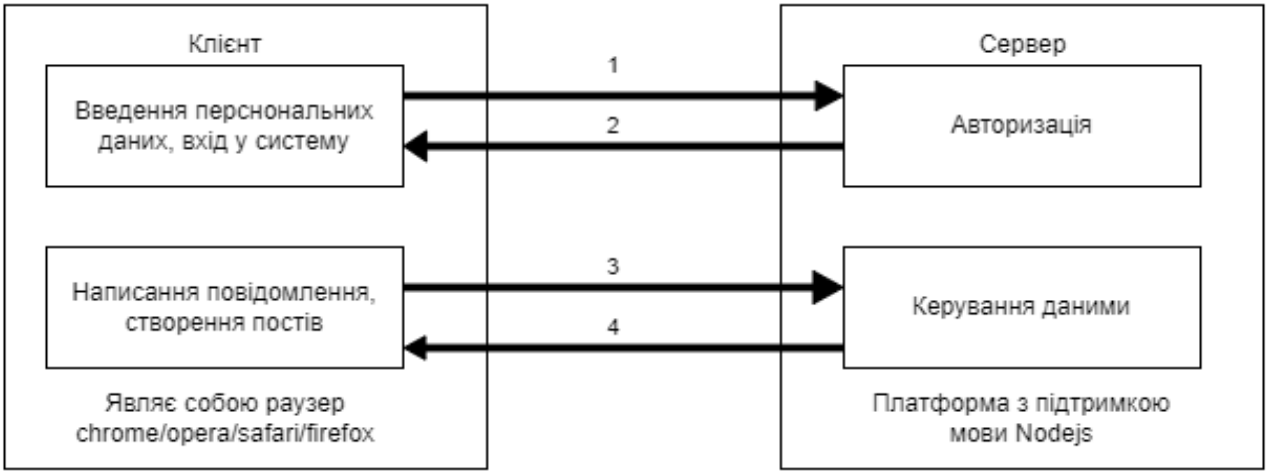


Рисунок 2.2 – Взаємодія між клієнтом і сервером

За рахунок технології AJAX [24], між клієнтом та сервером відбувається обмін даними. Під цифрою 1 на рисунку 2.2 позначено процес передачі зашифрованих реєстраційних даних до сервера від форм реєстрації та вхід користувача в систему, під цифрою 2, якщо отримано підтвердження реєстрації, надання доступу до системи.

Після цього, перейшовши на головну сторінку, користувач може користуватись системою, створюючи дописи, переглядаючи стрічку та надсилаючи повідомлення через модуль, дані яких передаються клієнту з сервера, що позначено цифрою 4. В свою чергу сервер застосунок працюватиме з сервером БД MongoDB.

Під цифрою 1 на рисунку 2.3 позначено процес надсилання захешованих реєстраційних даних до БД. Їхнє отримання позначено цифрою 2, для

порівняння хешів паролів під час входу в систему  механізму створення токенів, що вимагає даних користувача, але збереже ресурси БД і часу очікування користувачів за рахунок відсутності ведення механізму сесій. Під цифрою 3 позначено надсилання повідомлень на сервер, а також отримання цих повідомлень процесами запитів даних для інших користувачів – 4.

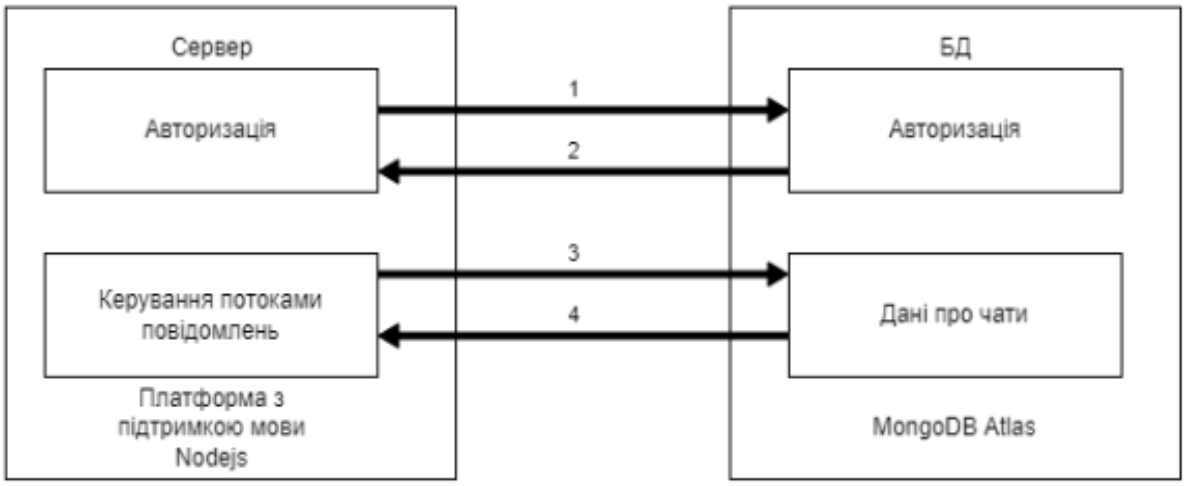


Рисунок 2.3 – Взаємодія між сервером і БД

В результаті проведеної роботи моделювання, нижче було виділено такі структурні елементи для опрацювання.

БД складатиметься з наступних таблиць:

- таблиці даних користувачів;
- таблиці з дописами користувачів;
- таблиці з повідомленнями користувачів;
- таблиці з чатами;
- таблиці з токенами.

Для користувача виділено такі ключові сторінки і їх складові:

- сторінка з формою реєстрації даних та авторизації;
- сторінка з формою для відновлення пароля;
- сторінка для перегляду стрічки дописів користувачів, включає:
- форма друзів, які перебувають онлайн;

- навігацій на панель;
- стрічна з дописами користувачів;
- сторінка для перегляду профіля користувача;
- сторінка для роботи з месенджером, включає:
- форма відображення чатів;
- форма для пошуку користувачів із списку друзів;
- форма для перегляду та створення повідомлень.

Для сервера виділено такі ключові модулі і їх складові:

- модуль для зберігання налаштувань сервера – порт, стрічка підключення до БД;
- модулі для роботи з таблицями БД;
- модуль API для роботи з даними реєстрації користувачів, входу користувачів;
- модуль для роботи з користувачами і їх пошуку по параметру;
- модуль для роботи з дописами;
- модуль для роботи з чатами, їх створення, підписання користувачів на створені чати, отримання повідомлень чату та написання повідомлень.

Розділення застосунку на такі модулі забезпечує можливість добавлення нового функціоналу без зміни роботи вже існуючого, а виділення роботи з інформацією месенджера такою як чати, користувачі та повідомлення в різні блоки та встановлення API шляху доступу до них забезпечить можливість розширення застосунку.

### 2.2 Проектування таблиць БД

MongoDB не вимагає створення строгої схеми полів і їх типів під час роботи з її записами, але це значно спрощує роботу, тому для роботи з MongoDB було вибрано бібліотеку mongoose [25]. В даній бібліотеці є

можливість задати структуру для колекції, в подальшому використовуватимуться таблиці, аналогічно до того, як це робиться в SQL БД.

В застосунку було створено таблиці користувачів, чатів, повідомлень, токенів та дописів. В таблиці 2.1 наведено перелік полів таблиці користувачів соціальної мережі.

Таблиця 2.1 – Поля таблиці користувач

| **Назва поля** | **Тип** | **Дані для зберігання** |
|---|---|---|
| email | String | Пошта користувача |
| password | String | Шифрований пароль користувача |
| username | String | Ім’я користувача |
| profilePicture | String | Шлях до фотографії користувача |
| coverPicture | String | Шлях до фотографії шапки профілю користувача |
| followers | Array | Масив ідентифікаторів користувачів, за якими стежить користувач |
| followins | Array | Масив ідентифікаторів користувачів, які стежать за даним користувачем |
| desc | String | Опис користувача |
| city | String | Місце проживання користувача |
| from | String | Походження користувача |
| relationship | Number | У яких відносинах перебуває користувач. Можливі варіанти 1 – неодружений, 2 – одружений, 3 – не вказано |
| isActivated | Boolean | Вказує чи користувач активував сторінку через електронну пошту |

В таблиці 2.2 наведено перелік полів таблиці, яка містить інформацію про чати месенджера.

Таблиця 2.2 – Поля таблиці повідомлення

| **Назва поля** | **Тип** | **Дані для зберігання** |
|---|---|---|
| text | String | Текст повідомлення |
| converstationId | String | Ідентифікатор чату, де повідомлення було створено |
| sender | String | Ідентифікатор користувача, який надіслав повідомлення |

В таблиці 2.3 наведено перелік полів таблиці чатів.

Таблиця 2.3 – Поля таблиці чати

| **Назва поля** | **Тип** | **Дані для зберігання** |
|---|---|---|
| member | Array | Масив ідентифікаторів користувачів чату |

В таблиці 2.4 наведено перелік полів таблиці дописів.

Таблиця 2.4 – Поля таблиці допис

| **Назва поля** | **Тип** | **Дані для зберігання** |
|---|---|---|
| userId | String | Ідентифікатор автора допису |
| desc | String | Опис допису |
| img | String | Шлях до зображення |
| likes | Array | Масив ідентифікаторів користувачів, яким сподобався допис |

В таблиці 2.5 наведено перелік полів таблиці токенів.

Таблиця 2.4 – Поля таблиці токен

| **Назва поля** | **Тип** | **Дані для зберігання** |
|---|---|---|
| user | ObjectId | Ідентифікатор користувача допису |
| refreshToken | String | Токен для оновлення логіну користувача |

## 2.3 Розробка інтерфейсу користувача та його верстка

Інтерфейс користувача повинен бути за вимогам простоти і зрозумілості у використанні. Із вище проаналізованих систем було вибрано схему кольорів і користувацький інтерфейс месенджера Facebook в якості основи.

Для швидшої і якіснішої реалізації інтерфейсу користувача було використано React UI інструмент – Material UI. Дана бібліотека забезпечує

велику кількість інструментів інтерфейсу користувача, а саме різні компоненти, форми, обгортки для вирівнювання елементів сторінки та багато іншого [26]. Реалізації форми входу продемонстровано на рисунку 2.4.

```
import { Box, Button, Grid, TextField, Typography } from "@mui/material";

export const Login = () => {
  return (
    <Grid>
      <Grid item xs={4}>
        <Box>
          <Typography>єДрузі</Typography>
          <Typography>
            Спілкуйтеся з друзями та навколишнім світом на Social...
          </Typography>
        </Box>
      </Grid>
      <Grid item xs={4}>
        <Grid>
          <Grid item xs={12}>
            <TextField label="Email"/>
          </Grid>
          <Grid item xs={12}>
            <TextField label="Пароль" type="password"
            />
          </Grid>
          <Grid item xs={12}>
            <Button>Увійти</Button>
          </Grid>
          <Grid item xs={12}>
            <Typography >
              Забули пароль?
            </Typography>
          </Grid>
          <Grid item xs={12}>
            <Button>Створити новий акаунт</Button>
          </Grid>
        </Grid>
      </Grid>
    </Grid>
  );
}
export default Login;

```

Рисунок 2.4 – Реалізація компонента входу з використанням бібліотеки MUI

Створивши компонент, його потрібно об’єднати з серверною стороною. Для цього використаємо бібліотеку MobX. Дана бібліотека досить просте рішення для керування станами компонентів, які можна масштабувати. Це дозволяє розробнику використовувати стан програми за межами інтерфейсу користувача, роблячи код незалежним і повторюваним. Також вона дозволяє реалізувати спостереження за компонентами по принципі публікації/підписки. Підписки обробляються автоматично, бібліотека відстежує, де використовуються зареєстровані параметри і лише тоді виконує побічні

ефекти, наприклад ререндиринг, якщо вони змінюються [27]. Приклад реалізації наведено на рисунку 2.5.

```
import { makeAutoObservable } from "mobx";
import AuthService from "../services/AuthService";
import axios from "axios";
import { API_URL } from "../http";

export default class AuthStore {
  user = {};
  isAuth = false;
  isLoading = false;

  constructor() {
    makeAutoObservable(this);
  }

  setAuth(bool) {
    this.isAuth = bool;
  }

  setUser(user) {
    this.user = user;
  }

  setLoading(bool) {
    this.isLoading = bool;
  }

  async login(data) {
    const response = await AuthService.login(data);
    localStorage.setItem("token", response.data.accessToken);
    this.setAuth(true);
    this.setAuth(response.data.user);
  }
}
```

Рисунок 2.5 – Створення глобального стану для зберігання даних користувача

Для реалізації серверних запитів використаємо бібліотеку Axios. Це HTTP-клієнт, який дозволяє надсилати асинхронні запити використовуючи REST і виконувати базові операції CRUD. Дана бібліотека часто використовується з вище зазначеними технологіями. Axios надає змогу реалізувати так звані перехоплювачі (interceptors). Це функції, яка викликається для оновлення або зміні кожного запиту перед його надсиланням або поверненням [28].

У даному випадку перехоплювачі використовуються для додавання токену для авторизації користувача підчас певного запиту, тому що незареєстровані користувачі не мають доступу до усіх можливостей веб-застосунку (див. рисунок 2.6). Також підчас отримання відповіді, перевіряємо чи статус відповіді рівний 401, якщо так потрібно перевірити чи токен

користувач дійсний і оновити його, якщо ні означає що користувач не авторизований.

```
import axios from 'axios';

export const API_URL = 'http://localhost:8080/api';
const $api = axios.create({
  withCredentials: true,
  baseURL: API_URL,
});

$api.interceptors.request.use((config) => {
  config.headers.Authorization = `Bearer ${localStorage.getItem('token')}`;
  return config;
});

$api.interceptors.response.use((config) => config, async (error) => {
  const originalRequest = error.config;

  if (error.response.status === 401 && error.config && !error.config._isRetry) {
    try {
      const response = await axios.get(`${API_URL}/auth/refresh`, { withCredentials: true });
      localStorage.setItem('token', response.data.accessToken);

      return $api.request(originalRequest);
    } catch (e) {
      console.error('User is not authorized!');
    }
  }
  throw error;
});

export default $api;
```

Рисунок 2.6 – Реалізація перехоплювачів

Тож, після реалізації вище зазначених налаштувань і впровадження компонентів із бізнес логікою отримаємо сторінку входу користувача подібної до форми Facebook (див. рисунок 2.7).

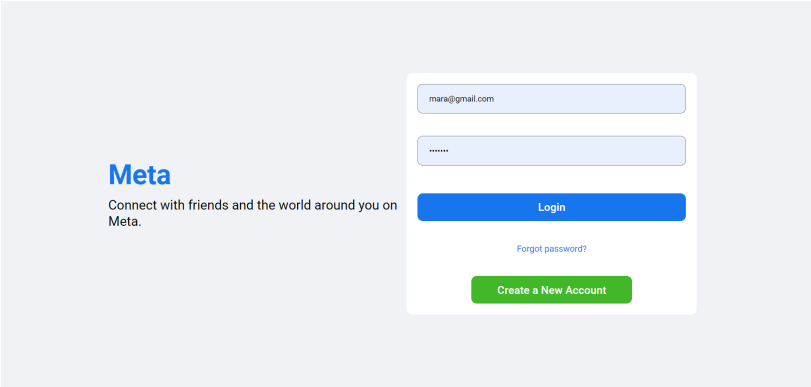


Рисунок 2.7 – Сторінка входу в застосунок

Аналогічним чином створена сторінка реєстрації користувача. Вигляд форми входу в систему зображено на рисунку 2.8, а код реалізації даного компоненту розміщено у додатку А.

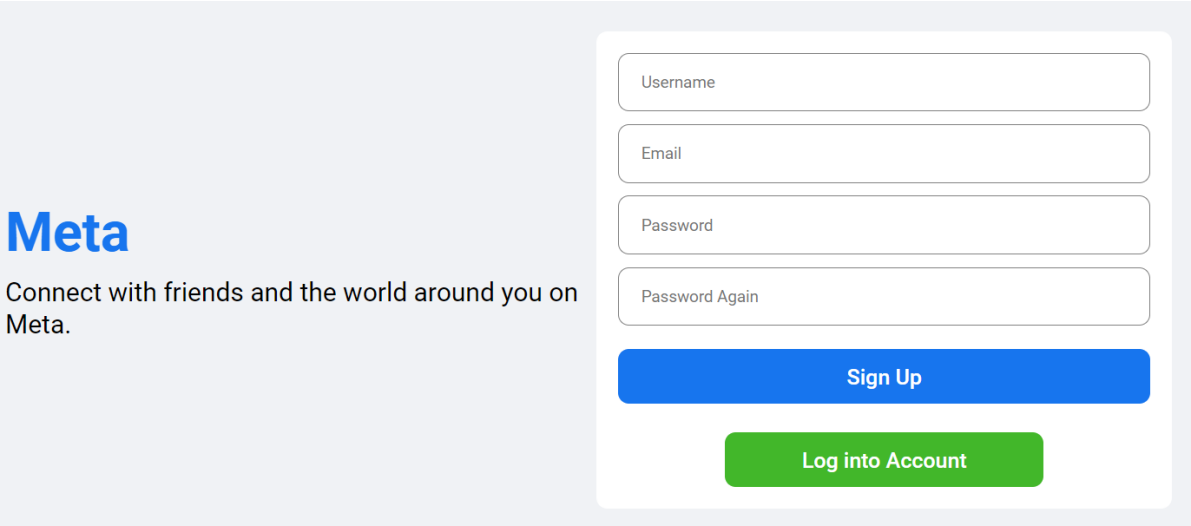


Рисунок 2.8 – Сторінка реєстрації в застосунок

Після реєстрації або ж входу ми переходимо на домашню сторінку, яка включає різні інструменти. Домашня сторінка дозволяє побачити стрічку дописів, яка включає свої дописи і друзі. Також в лівому меню розміщується посилання на сторінки чатів, відео, груп, закладок, робіт, подій, курсів. Трохи нижче ми може побачити список свої друзів. У правому меню розміщуються друзі онлайн і різні сповіщення. JSX код сторінки наведено в додатку Б. Вигляд сторінку зображено на рисунку 2.9.

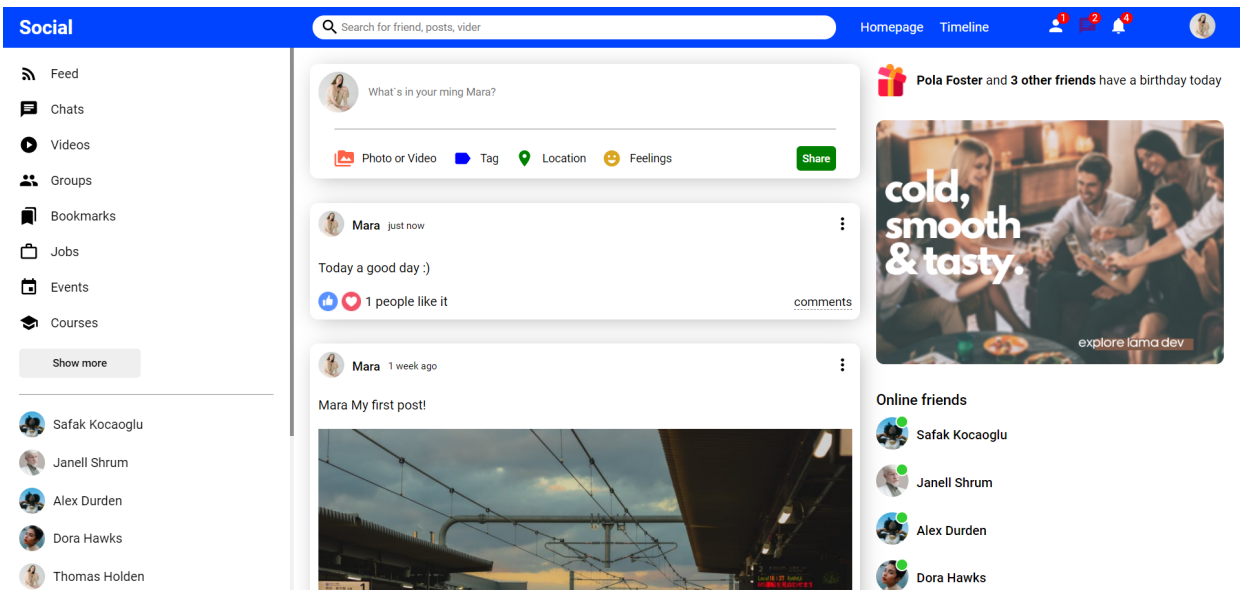


Рисунок 2.9 – Домашня сторінка

Одною з основних сторінок даного застосунку є месенджер. Сторінка роботи з чатами розділена на три блоки. Блок чатів користувачів, який складається з списку друзів, з якими існує розмова, і з блоком пошуку. Посередині розміщується сам чат, який працює з використанням технології Soket.io [29], яка дозволяє користувача бачити надіслані іншим користувачем повідомлення без обновлення сторінки. І останній блок – це список наших друзів. В результаті отримали наступну сторінку, яка зображена на рисунку 2.10. JSX код наведено в додатку В.

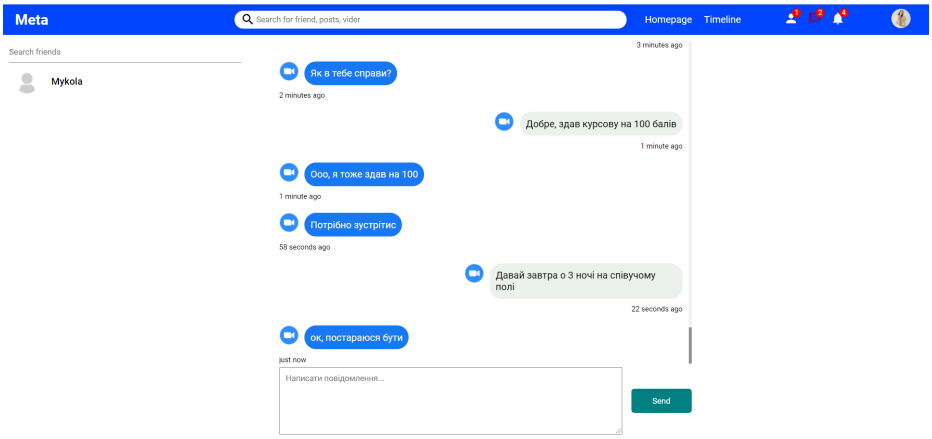


Рисунок 2.10 – Сторінка месенджера

Також створена сторінка профілю, яка відображає загальну інформацію про користувача, дозволяє підписатися на нього і побачити його дописи. Дана сторінка зображена на рисунку 2.11, а код розміщений у додатку Г.

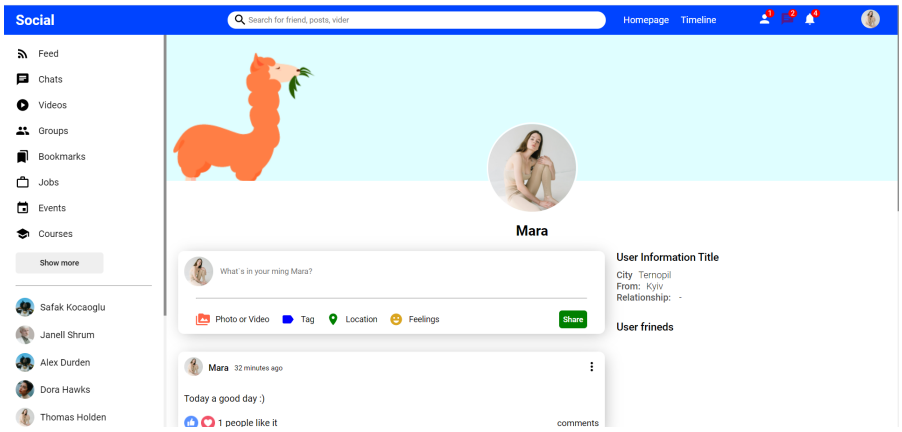


Рисунок 2.11 – Сторінка профілю

## 2.4 Розробка серверної частини

Відповідно до алгоритмів роботи, необхідно реалізувати сервер обробки даних, який буде приймати запити від клієнтської частини та, виконуючи обробку даних, забезпечувати її потрібною інформацією. Для вирішення цієї задачі, з сторони імплементації, буде доцільно використати шаблон програмування MVC.

MVC – один з патернів проектування, що складається з кількох невеликих шаблонів. Під час реалізації MVC код застосунку розбивається на три компоненти: модель представлення даних, візуальний інтерфейс користувача і логіка комунікації клієнта з системою. За рахунок цього, редагування одного з компонентів завдає мінімального впливу на інші компоненти або не робить цього зовсім [30].

Цей патерн використовується для того, щоб розділити серверну логіку та дані від візуальної частини. Як результат ми отримаємо змогу повторно використовувати код і спрощується його підтримка.

Концепція MVC спричинює поділ коду на компоненти, в яких відбувається обробка даних відповідно до бізнес-логіка (див. рисунок 2.12).

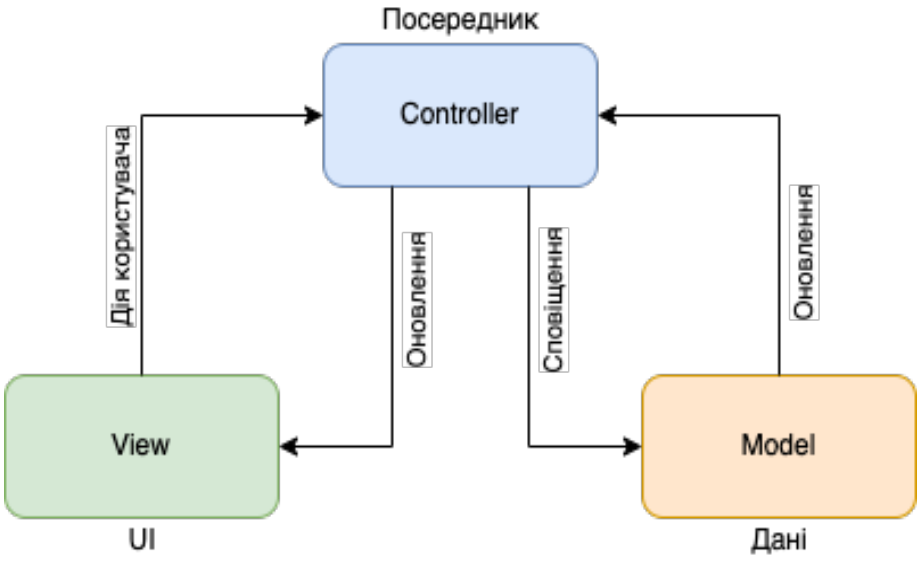


Рисунок 2.12 – Взаємодія компонентів у MVC

Model – є це об’єкт, який містить в собі дані та методи їх, повертає набір даних на певний запит. Модель не містить інформації, які саме дані будуть відображені і яким способом, а також не взаємодіє з користувачем на пряму.

View – забезпечує візуалізацію інформації, яку отримала від сервера. Дані можуть відображатися по різному в залежності від варіанту використання і місця. Одні й ті самі дані можуть бути візуалізовані у різний спосіб, наприклад, масив об'єктів з даними можна подати користувачеві як таблицю, так і список, а для комунікації з різними API вони можуть бути у форматі JSON або XML.

Controller – з’єднує користувача та сервер, використовує дані та їхню візуалізацію для впровадження потрібних сценаріїв. Зазвичай, у контролері відбувається валідація вхідних даних та авторизація, яка перевіряє чи користувач має доступ на виконання певних дій у системі.

Розробка серверної і клієнтської сторони єдино мовою програмування надає наступні переваги:

- однин розробник може одночасно виконувати завдання на обох сторонах застосунку;
- певну логіки можна винести на сервер з візуальної частини або ж навпаки;
- спільне представлення даних (JSON);
- спільні інструменти для розробки (IDE);
- на серверній і клієнтській стороні можна застосувати один підхід до написання коду;

Як вище зазначено, для реалізації серверної чистини буде використано Node.js. На структуру проекту впливають власні перевага, вподобання, використана архітектура проекту і хід впровадження компонентів.

При створенні сервера використано бібліотеку express, яка забезпечує маршрутизацію запитів в середині сервера та виконання прописаних скриптів

в залежності від шляху запиту. Приклад реалізації даного функціоналу зображено на рисунку 2.13.

```
const Router = require("express").Router;
const router = new Router();
const userController = require("../controllers/user-controller");
const postController = require("../controllers/post-controller");
const { body } = require("express-validator");
const authMiddleware = require("../middllewares/auth-midalleware");

router.post("/registration", body("email").isEmail(), body("password").isLength({ min: 3, max: 32}), userController.registration);
router.post("/login", userController.login);
router.post("/logout", authMiddleware, userController.logout);
router.get("/activate/:link", authMiddleware, userController.activate);
router.get("/refresh", authMiddleware, userController.refresh);
router.get("/users", authMiddleware, userController.getUsers);
router.get("/user/", authMiddleware, userController.get);
router.put("/user/:id", authMiddleware,userController.updateUser);
router.get("/user/friends/:userId", authMiddleware, userController.getFriendsByUserId);
router.put("/user/:id/follow", authMiddleware, userController.follow);
router.put("/user/:id/unfollow", authMiddleware, userController.unfollow);

router.post("/post/create", authMiddleware, body("desc").isLength({ max: 500}), postController.create);
router.put("/post/:id", authMiddleware, body("desc").isLength({ max: 500}), postController.update);
router.delete("/post/:id", authMiddleware, postController.delete);
router.put("/post/:id/like", authMiddleware, postController.like);
router.get("/post/:id", authMiddleware,  postController.get);
router.get("/post/timeline/:userId", authMiddleware, postController.getTimeLine);
router.get("/post/profile/:username", authMiddleware, postController.getTimeLineByName);

module.exports = router;
```

Рисунок 2.13 – Реалізація маршрутизації запитів

Для кожного запиту можна додати ланцюжок обробників, так званих middleware. Це функції, які будуть виконуватися підчас життєвого циклу запиту до сервера. У даному випадку один з них це функція перевірки доступу користувача до команди.

Розглянемо реалізацію алгоритму реєстрації користувача у застосунку. Дана команда починає свій шлях з користувацького інтерфейсу, коли він заповняє форму реєстрації і у випадку коли дані валідні і паролі збігаються надсилає дані на сторону сервера. Алгоритм процесу реєстрації від натискання кнопки до запису даних у БД зображений в блок-схемі на рисунку 2.14.

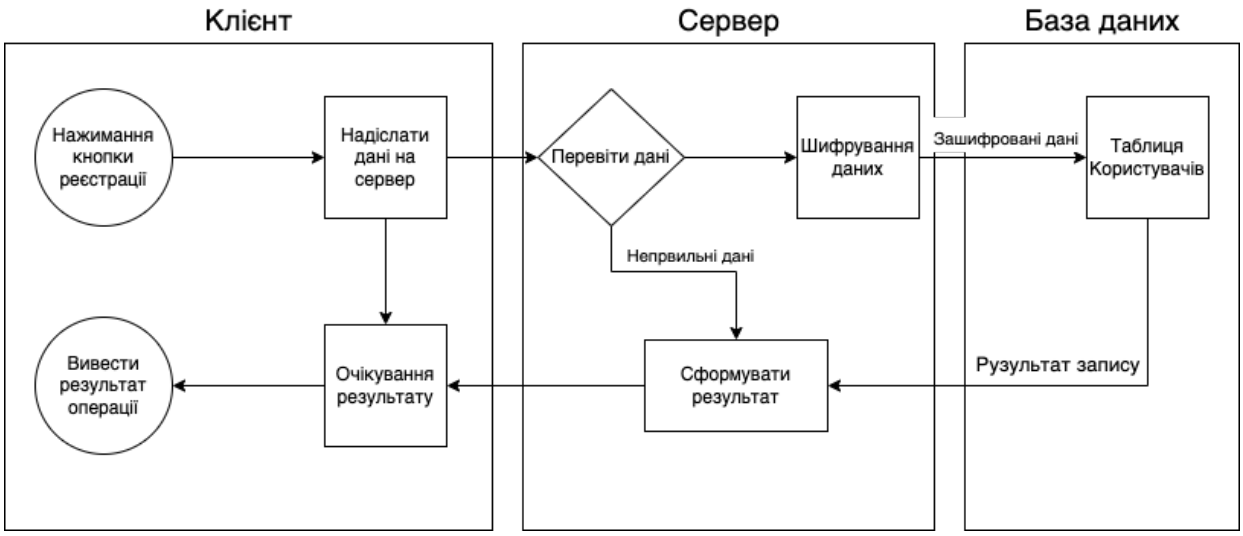


Рисунок 2.14 – Алгоритм реєстрації

Для безпеки даних користувача особисті дані, такі як пароль, потрібно хешувати. На даних момент існує багато готових бібліотек, у яких реалізовано подібні алгоритми, одна із них - Bcrypt. Хешування виконує одностороннє перетворення паролю, перетворюючи пароль в інший рядок, який називається хешованим паролем. Хешування називається одностороннім, тому що отримати оригінальний текст з хешу практично неможливо. На рисунку лістингу 2.15 наведено реалізацію алгоритму реєстрації на стороні сервер.

```
async registration(dtoIn){
    const candidate = await UserModel.findOne({email: dtoIn.email});
    if (candidate) {
      throw ApiError.BadRequest(`Користувач з поштою ${dtoIn.email} існує`);
    }
    const hashPassword = await bcrypt.hash(dtoIn.password, 3);
    const activationLink  =  uuId.v4();
    const user = await UserModel.create({ ...dtoIn, password: hashPassword });
    await mailService.sendActivationMail(dtoIn.email, `${process.env.API_URL}/api/activate/${activationLink}`);

    const userDto = new UserDto(user);
    const tokens = tokenService.generateTokens({ ...userDto });
    await tokenService.saveToken(userDto.id, tokens.refreshToken);

    return {
      ...tokens,
      user: userDto,
    }
  }
```

Рисунок 2.15 – Реалізація реєстрації

## 2.5 Розробка месенджера з використання Socket.io

Сьогодні будь-яку соціальну мережу важко уявити без месенджерів. Вони дозволяють нам обмінюватися повідомленнями, фотографіями та відео, створювати групові чати та здійснювати відео дзвінки.

Для реалізації даного функціоналу використовують технологію Socket.io. Це бібліотека, яка надає змогу обмінюватися повідомленнями в режимі реального часу між клієнтами, які підключені до сервера. Для її реалізації потрібно два елементи: клієнт, який запускається в браузері та сервер, який буде обробляти події, які відбуваються. Ці дві частини мають схожий підхід до імплементації. Бібліотека Socket.IO, як і Node.js – подієво-орієнтована. Крім того, дана бібліотека містить інші функції, наприклад, підключення до кількох сокетів або асинхронний ввід і вивід [31].

Дана технологія дозволить там отримувати повідомлення без перезавантаження сторінки, що є дуже зручним у наш час. Для того, щоб обмінюватися повідомленнями потрібно добавити кілька подій, як на серверній стороні, так і на клієнтській. Приклад реалізація зображено на рисунку 2.16.

```
io.on("connection", (socket) => {
  //when ceonnect
  console.log("A user connected.");
  //take userId and socketId from user
  socket.on("addUser", (userId) => {
    addUser(userId, socket.id);
    io.emit("getUsers", users);
  });
  //send and get message
  socket.on("sendMessage", ({ senderId, receiverId, text }) => {
    const user = getUser(receiverId);
    io.to(user.socketId).emit("getMessage", {
      senderId,
      text,
    });
  });
  //when disconnect
  socket.on("disconnect", () => {
    console.log("a user disconnected!");
    removeUser(socket.id);
    io.emit("getUsers", users);
  });
});
```

Рисунок 2.16 – Реалізація подій у Socket.IO

## 2.6 Висновок до другого розділу

У другому розділі кваліфікаційної роботи спроектовано структуру бази даних, визначено основні сутності, зв’язки між ними і їхню структуру. Аргументовано архітектурні рішення і підхід до створення клієнтської і серверної сторони. Проаналізовано основні помилки підчас реалізації подібних веб-застосунків. Розроблено клієнтську частину з використанням бібліотек: React.js, MobX і Material UI. У свою чергу серверну частину створено на базі Node.js з використанням бібліотеки Express.js, впроваджено CRUD команди для більшості сутностей. Також, імплементовано месенджер на основні технології Socet.io. Створено авторизацію за допомогою JWT токенів, які дозволяють бути залогіниним протягом місяця у системі і гарантують безпеку особистих даних.

# РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

## 3.1 Долікарська допомога при ураженні електричним струмом

Удар електричним струмом є поширеною травмою і часто може закінчитися летальним випадком. Ураження електричним струмом відбувається підчас контакту тіла людини з джерелом електричної енергії під напругою. Це реакція організму людини на проходження електричного струму через тіло. Вона проявляється по різному, від легких потрясінь до небезпечних здоров'ю травм, які можуть вплинути на тканини в організмі.

Шкода завдана електричним струмом залежать від кількох факторів: наскільки висока була напруга, яка область тіла була вражена і від виду струму. Фізичні наслідки для людини можуть коливатися від опіків частин тіла до серйозних вражень внутрішніх органів [32].

Часто люди стикаються з підвищеним ризиком ураження високою напругою. Низька напруги не наносить серйозних травм для люди, а з іншої сторони висока напруги, яка має більше 500 В, може призвести до серйозного пошкодження тканин.

У дітей або підлітків при враженні електричним струмом в діапазоні від 110 до 200 В можуть вникнути значні травми. Зазвичай це трапляється підчас порушення техніки безпеки при роботі з електричними приладами, які є в побуті. Це можуть бити електричні шнури, подовжувачі, несправні розетки та багато чого іншого.

Виділяють чотири основні фактори від яких залежить вплив електричного струму на організм:

- величина струму, що протікає через тіло;
- органи, через які проходить струм;
- час, протягом якого струм вражає тіло;
- частота струму.

Фізичні наслідки на пряму залежать від величини струму, яка вражає тіло людини. Струм менший за 1 мА не завдає жодної шкоди людина фізичного ефекту. При 1 мА можуть відчуватися слабкі поколювання, а при 5 мА – легкий поштовх. Однак при струмі ведичною від 6 до 25 мА людина може відчувати больовий шок і деяку втрату контролю над м’язами.

При ураженні електричним струмом від 50 до 150 мА може спричинити у людини сильний біль, м’язове скорочення і навіть призвести до зупинку дихання. У певних випадках можливий летальний результат для людини. Струм від 1000 до 4300 мА призводить до ймовірної смерть, тому що дана напруга спричиняє пошкодження нервових зав’язків, м’язових скорочення та порушення ритму серця.

До 10 000 мА електричний струм спричиняє важкі опіки шкіри людини та зупинку серця. Висока ймовірність смерті.

Найпоширеніші ознаки та симптоми враження електричним струмом включають:

- втрата свідомості;
- ускладнення або зупинка дихання;
- опіки, які виникають там, де струм входить і виходить з тіла;
- зупинка серця;
- слабкий і непостійний пульс або його припинення.

Перш за все, що потрібно зробити при першій допомозі, це відключити джерело живлення. Вимкнути електропостачання, від’єднати електроприлад від джерела електричного струму або вимкнути блок запобіжників, якщо він знаходиться неподалік. Не потрібно намагатися підходити близько до жертви, якщо не переконані, що це безпечно і живлення вимкнено.

Потрібно бути обережним у вологих місцях, тому що вода є електричним провідником і рятівник може стати теж жертвою. Якщо людина не впевнена щодо вологості поверхні, потрібно відключити основне електропостачання будинку. У випадку коли це неможливо, використати

підручний предмет, який не є провідником і відокремити людину від джерела струму. Це може бути дерев'яна або пластмасова річ.

Після того як постраждалого відокремили від джерела електрики, потрібно викликати швидку і надати першу медичну допомогу. Далі потрібно визначити стан жертви. Перевірити, чи людина у свідомості і дихає. У складних випадках у жертви може бути слабкий пульс або його відсутність. Можливо, що дихання зупинилося. Якщо людина втратила свідомість і перестала дихати, потрібно почати серцево-легеневу реанімацію. Руки розташовуємо в центрі грудної, одна на іншу. Сильно і швидко натиснути 30 раз приблизно до третини діаметра грудної клітки. Після кожного натискання на грудну клітку робиться два рятувальні вдихи. Потрібно відкинути голову потерпілого назад і підняти підборіддя. Затиснути ніс і створити повне ущільнення. Далі подути потерпілому в рот і подивитися, чи підніметься грудна клітка. Потрібно продовжувати робити натискань на грудну клітку та вдих, поки не прибуде медична допомога або людина не почне сама дихати. Якщо потерпілий живий, перемістити його у зручне йому положення подальше від небезпеки. Можна запобігти шоку, поклавши людину рівно на землю, з головою трохи нижче тіла [33].

Якщо людина при свідомості, нормально дихає і на тілі є опіки, потрібно накрити їх звичайною харчовою плівкою або іншою неклейкою пов'язкою, але без мазі чи лосьйону. Якщо кровотеча у потерпілого, може знадобитися компресія та джгут.

При роботі з соціальною мережею користувач повинен знати і вміти як правильно поводитися з ПК, тому що людина перебуває у непосередньому контакті з джерелом напруги. Удар електричним струмом є потенційно смертельною травмою. Негайна медична допомога важлива, щоб запобігти серйозним травмам і смерті.

Для запобігання уражень електричним струмом при роботі за ПК слід встановити додаткові захисні пристрої, що забезпечують недоступність токо-

провідних частин для дотику. З метою зменшення небезпеки можна використовувати розділовий трансформатор для розв'язки з основною мережею.

Удар електричним струмом є потенційно смертельною травмою. Негайна медична допомога важлива, щоб запобігти серйозним травмам і смерті.

### 3.2 Вимоги ергономіки до організації робочого місця оператора ПК

Робоче місце – це ділянка простору, яка облаштована необхідним обладнанням, відповідно до трудової діяльності, для виконання поставлених завдань.

Правильно побудоване робоче місце повинне забезпечувати:

- найкраще розміщення обладнання і предметів праці;
- не допускати дискомфорту;
- підвищувати продуктивність праці;
- зменшувати втому працівника.

Розмір робочого місця повинне бути таким, щоб людина не виконувала лишніх рухів і не відчувала дискомфорту під час виконання роботи. Також для працівника важливо мати змогу змінити робочу позу, наприклад, положення тулуба, рук або ніг. Потрібно мінімізувати або звести до нуля всі незручності положення тіла [34].

Різні дослідження заявляють, що при правильному проектуванні робочого місця продуктивність людини може зрости від 15-25%.

Такі фактори як рівень освітлення, вологість повітря, температура, шум, вібрація, токсичність, мають значний вплив на умови життєдіяльності і працездатності людини.

Антропометричні вимоги визначають відповідність робочого місця до фізіологічних параметрів тіла людини як зріст і розміри тіла. Індикатором

цього є правильна робоча поза, відсутність дискомфорту, оптимальні зони досягнення, раціональні рухи.

Психофізіологічні та фізіологічні вимоги формулюють відповідність обладнання і робочого місця можливостям співробітника щодо розуміння, обробки даних, пошук і реалізації рішень.

Організація робочого місця передбачає наступні пункти:

- раціональне положення робочого місця у приміщенні;
- вибір робочих меблів відповідно до фізіологічних характеристик працівника;
- правильне компонування і розміщення обладнання на робочих місцях;
- урахування особливостей та характеру професійної діяльності.
- До загальних принципів організації робочого місця відносять:
- робоче місце повинне містити тільки ті предмети, які беруть учать у робочому процесі, але не заважати йому;
- предмети, які часто використовуються у роботі, розміщуються ближче, ніж ті речі, якими користуються рідше;
- предмети, які беруться лівою рукою, повинні розміщуватися зліва, а предмети, які використовуються правою рукою — справа;
- якщо при роботі з предметом працівник використовує дві руки, то він розміщується з урахуванням зручності захоплювання його двома руками;
- робоче місце не повинно бути засмічене;
- необхідна оглядовість повинна бути забезпечена при правильній організації робочого місця.

Робоча поза – це найбільш тривале положення тіла працівника протягом робочого дня. При зручній робочій позі забезпечується стійкість положення тулуба, ніг, рук, голови і витрачається мінімальний запас енергії та максимальну продуктивність [35].

Сидячки і стоячи – дві найпопулярніших пози у робочому процесі. При проектуванні робочого місця потрібно враховувати, що з фізичним навантаженням бажана поза стоячи, а при малих зусиллях – сидячи.

При роботі стоячи, людина стомлюється більше ніж сидячи. У відсотковому еквіваленті це на 10% більше енергії. При додатковому навантаженні підвищується артеріальний і венозний тиск крові, розширення вен, пошкоджуються ступені та викривляється хребет.

У свою чергу при сидячій роботі нижня частина тіла розслаблена, а основне навантаження спрямоване на м'язи шиї, спини, таза, стегон. При неправильній сидячій позі розвивається застій крові у ногах, а якщо пальці виконують багато роботи можливе запалення суглобів.

Організація робочого місця при використанні персонального комп’ютера повинна відповідати усім ергономічним вимогам. Ключові ергономічні вимоги до проектування робочого місця оператора ПК зображені на рисунку 3.1.

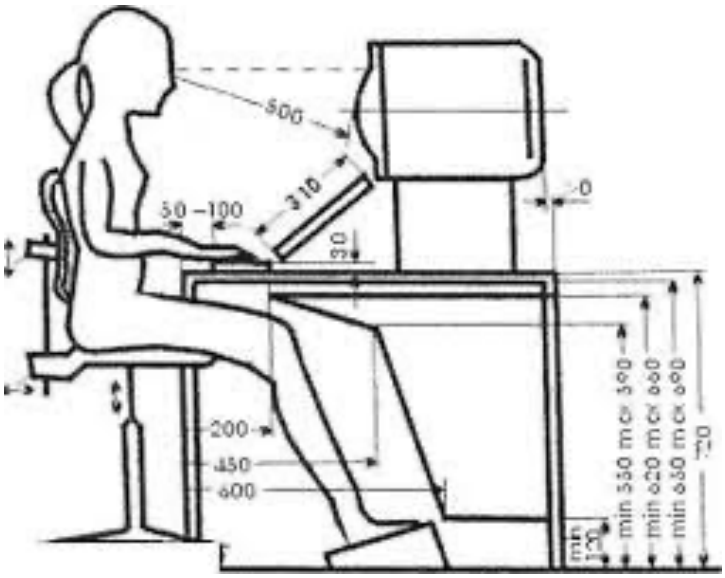


Рисунок 3.1 – Робочий стіл і розміщення користувача ПК

При роботі з персональним комп’ютером потрібно:

- зменшувати кількість статичних напружень;
- розподіляти кількість і час статичних напружень;

- змінювати робочі пози під професійної діяльності.

Саме вибір правильної робочої пози визначається від впливу багатьох факторів. Одні з найважливіших це – кількість зусиль яка прикладається, велична робочої зони, відношення висоти робочої поверхні і ростом працівника.

При використанні соціальної мережі користувач повинен дотримуватися вище зазначених правил, які будуть сприяти комфортній і продуктивній роботі. Важливо регулярно робити короткі перерви. Часта зміна заняття – кращий спосіб уникнути можливих неприємностей.

# ВИСНОВКИ

У результаті виконання кваліфікаційної роботи освітнього рівня «Бакалавр» було розроблено соціальну мережу. Даний веб-застосунок створений засобами MonogDB та серверної мови програмування Node.js.

В першому розділі було розглянуто історію встановлення соціальних мереж і їхню роль у нашому повсякденному житті. Також, проаналізовано основні аспекти даного типу веб-застосунків і сформовано список вимог. Вибрано та аргументовано середовище розробки і основні технології для реалізації, зроблено пошук ключових акторів та сформовано діаграми варіантів використань, а також розроблено структуру соціальної мережі на базі трирівневої архітектури.

В другому розділі кваліфікаційної роботи спроектовано структуру бази даних, визначено основні сутності, зв’язки між ними і їхню структуру. Аргументовано архітектурні рішення і підхід до створення клієнтської і серверної сторони. Проаналізовано основні помилки підчас реалізації подібних веб-застосунків. Розроблено клієнтську частину з використанням бібліотек: React.js, MobX і Material UI. У свою чергу серверну частину створено на базі Node.js з використанням бібліотеки Express.js, впроваджено CRUD команди для більшості сутностей. Також, імплементовано месенджер на основні технології Socet.io. Створено авторизацію за допомогою JWT токенів, які дозволяють бути залогіниним протягом місяця у системі і гарантують безпеку особистих даних.

Після впровадження певного функціоналу, проводилося тестування усіх сценаріїв використання веб-застосунку.

У розділі «Безпека життєдіяльності, основи охорони праці» описано небезпеку яку несе електричний струм і наведено порядок необхідних дій для надання долікарської допомоги у різних ситуаціях. Також подано основні вимоги і поради ергономіки до організації робочого місця оператора ПК.

# ПЕРЕЛІК ДЖЕРЕЛ

1 Беркій Т. М. СОЦІАЛЬНІ МЕРЕЖІ: РІЗНІ АСПЕКТИ ВПЛИВУ НА ЛЮДИНУ [Електронний ресурс] / Т. М. Беркій. – 2019. – Режим доступу до ресурсу: https://ukrainepravo.com/legal_publications/essay-on-it-law/it_law_berkiy_Social_networks_and_there_involves/.

2 Кеннеді Д. Жорсткий SMM. Вичавити з соцмереж максимум / Д. Кеннеді, К. Уелш-Філліпс. – Паблішер: Київ, 2017. – 344 с. – ISBN 978-5-9614-6546-4.

3 Samur A. The History of Social Media: 29+ Key Moments [Електронний ресурс] / Alexandra Samur. – 2018. – Режим доступу до ресурсу: https://blog.hootsuite.com/history-social-media/.

4 Алексеєнко Ю. О. ТЕОРІЯ ТА ІСТОРІЯ СОЦІАЛЬНИХ КОМУНІКАЦІЙ / Ю. О. Алексеєнко // Вчені записки ТНУ імені В. І. Вернадського. Серія: Філологія. Журналістика / Ю. О. Алексеєнко. – Запоріжжя: MIT Press, 2021. – (32). – С. 204–208.

5 Крамаренко А. Соціальні медіа та бізнес: можливості і загрози [Електронний ресурс] / Анна Крамаренко // 2016 – Режим доступу до ресурсу: https://www.researchgate.net/publication/328532025_Socialni_media_ta_biznes_mozlivosti_i_zagrozi.

6 Коброслі, А. Х. "Соціальні мережі та психологічне благополуччя підлітків: переваги та ризики." Теоретичні і прикладні проблеми психології 2 (2019): 194-203.

7 Савицька Н. Л. Маркетинг у соціальних мережах: стратегії та інструменти на ринку B2C [Електронний ресурс] / Наталія Леонидівна Савицька. – 2017. – Режим доступу до ресурсу: https://mdt-opu.com.ua/index.php/mdt/article/view/5.

8 Петліна А. О. Дослідження процесу розробки UI/UX web-додатків : пояснювальна аписка до атестаційної роботи здобувача вищої освіти на

другому (магістерському) рівні, спеціальність 186 Видавництво та поліграфія / А. О. Петліна ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Харків, 2021. – 66 с.

9 Гринчак О. Веб-технології та дизайн / Олександ Гринчак. // вісник хмельницького національного університету. – 2021. – С. 22–26.

10 Tetskyi, A. "Аналіз проблем і можливостей забезпечення безпеки WEB-застосунків, створених за допомогою систем керування вмістом." Системи управління, навігації та зв'язку. Збірник наукових праць 1.53 (2019): 133-136.

11 Кокарча, Ю. А. "Віртуальна спільнота як елемент інформаційного суспільства в соціальних медіаресурсах." Соціально-політичні, економічні та гуманітарні виміри європейської інтеграції України: зб. наук. пр. VII Міжнар. наук.-практ. конф., 2019. – 144 с.

12 Фенін, О. М. Система обміну повідомленнями : дипломний проект ... бакалавра : 6.050102 Комп'ютерна інженерія / Фенін Олексій Михайлович. – Київ, 2019. – 90 с.

13 Бітаєва О. Розробка методології та рекомендацій з розробки "доступних" UI/UX веб застосунків : дис. канд. техн. наук : 121 / Бітаєва Олександра – Київ, 2021. – 56 с.

14 Голубенко Е. Как и зачем писать Use Cases [Електронний ресурс] / Елена Голубенко. – 2016. – Режим доступу до ресурсу: https://dou.ua/lenta/articles/use-cases/.

15 Марголін О. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс] / Олександр Марголін. – 2021. – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/uml-diagrams.html.

16 Алексенко, О.В. Технології програмування та створення програмних продуктів: конспект лекцій для студ. напряму підготовки 6.050101

"Комп'ютерні науки" усіх форм навчання / О.В. Алексенко. - Суми: СумДУ, 2013. - 133 с.

17 MongoDB [Електронний ресурс] // Вікіпедія, 2017. – Режим доступу: https://uk.wikipedia.org/wiki/MongoDB

18 Mardan, Azat. "Using Express. js to create Node. js web apps." Practical Node. js. Apress, Berkeley, CA, 2018. 51-87.

19 Mardan, Azat. "Using Express. js to create Node. js web apps." Practical Node. js. Apress, Berkeley, CA, 2018. 51-87.

20 React Документація [Електронний ресурс] // 18.1.0. – 2022. – Режим доступу до ресурсу: https://reactjs.org/.

21 Гордійчук І. Секрети швидкодії JavaScript: V8 та приховані класи [Електронний ресурс] / Ігор Гордійчук. – 2021. – Режим доступу до ресурсу: https://codeguida.com/post/3070.

22 Get Started with Atlas [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.mongodb.com/docs/atlas/getting-started/.

23 Змерзлий І. Клієнт-серверна архітектура та ролі серверів. [Електронний ресурс] / Ivan Змерзлий. – 2017. – Режим доступу до ресурсу: клієнт-серверна-архітектура-та-ролі-серверів-9893d8048229

24 Martsoukos G. Введення в AJAX для фронтенд-розробників [Електронний ресурс] / George Martsoukos. – 2017. – Режим доступу до ресурсу: https://webdesign.tutsplus.com/uk/tutorials/an-introduction-to-ajax-for-front-end-designers--cms-25099.

25 Munro J. Вступ до Mongoose для MongoDB та Node.js [Електронний ресурс] / Jamie Munro. – 2017. – Режим доступу до ресурсу: https://code.tutsplus.com/uk/articles/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527.

26 Material UI - Документація [Електронний ресурс] – Режим доступу до ресурсу: https://mui.com/.

27 Aleem I. Introduction to MobX with React [Електронний ресурс] / Isiaka Aleem. – 2019. – Режим доступу до ресурсу: https://blog.logrocket.com/introduction-to-mobx-with-react/.

28 Документація Axios [Електронний ресурс] – Режим доступу до ресурсу: https://axios-http.com/.

29 Калашников Ю. Створення простої програми для чату за допомогою node.js та socket.io [Електронний ресурс] / Юрій Калашников. – 2019. – Режим доступу до ресурсу: https://medium.com/freecodecamp-russia-русскоязычный/создание-простого-приложения-для-чата-с-помощью-node-js.

30 1. Model-View-Controller [Електронний ресурс] / Вікіпедія – Режим доступу: https://ru.wikipedia.org/wiki/Model-View-Controller – Загол. з екрану.

31 Кіптенко, Є.А. Інформаційна система обміну повідомленнями між розробниками програмного забезпечення [Текст]: робота на здобуття кваліфікаційного ступеня бакалавра; спец.: 122 - комп`ютерні науки (інформатика) / Є.А. Кіптенко; наук. керівник І.В.Шелехов. - Суми: СумДУ, 2020. - 57 с.

32 Г.М. Тіхомірова. Удар струмом: перша допомога, наслідки після ураження електричним струмом [Електронний ресурс] / Г.М. Тіхомірова – Режим доступу до ресурсу: http://tomrda.gov.ua/news/578646863743857435/.

33 Перша допомога при ураженні електричним струмом [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://bozhedarivska-selrada.gov.ua/news/1576497483/.

34 Ергономічні вимоги до організації робочих місць [Електронний ресурс] – Режим доступу до ресурсу: https://pidru4niki.com/14821111/bzhd/ergonomichni_vimogi_organizatsiyi.

35 Кравченко О. Охорона праці в офісі. Вимоги до робочого місця офісного працівника [Електронний ресурс] / Олена Кравченко. – 2022. – Режим доступу до ресурсу: https://gc.ua/uk/oxorona-praci-v-ofisi-vimogi-do-robochogo-miscya-ofisnogo-pracivnika/.

# ДОДАТКИ

**Додаток А**

## Лістинг коду сторінки реєстрації

```
import axios from "axios";
import { useRef, useContext } from "react";
import "./register.css";
import { useHistory } from "react-router";
import { Link } from "react-router-dom";
export default function Register() {
  const username = useRef();
  const email = useRef();
  const password = useRef();
  const passwordagain = useRef();
  const history = useHistory();
  const handleClick = async (e) => {
    e.preventDefault();
    if (passwordagain.current.value !== password.current.value) {
      passwordagain.current.setCustomValidity("Password don`t match");
    } else {
      const user = {
        username: username.current.value,
        email: email.current.value,
        password: password.current.value,
      };
      try {
        await axios.post("/auth/register", user);
```

```
      history.push("/login");
    } catch (error) {
      console.log(error);
    }
  }
};
return (
  <div className="login">
    <div className="loginWrapper">
      <div className="loginLeft">
        <h3 className="loginLogo">Meta</h3>
        <span className="loginDesc">
          Connect with friends and the world around you on Meta.
        </span>
      </div>
      <div className="loginRight">
        <form className="loginBox" onSubmit={handleClick}>
          <input
            placeholder="Username"
            required
            ref={username}
            className="loginInput"
```

```
            <input
              placeholder="Email"
              required
              ref={email}
              className="loginInput"
              type="email"
            />
            <input
              placeholder="Password"
              required
              ref={password}
              className="loginInput"
              type="password"
              minLength={6}
            />
            <input
              placeholder="Password Again"
              required
              ref={passwordagain}
              className="loginInput"
              type="password"
              minLength={6}
            />
            <button className="loginButton">Sign Up</button>
            <Link to="/login" style={{textDecoration: "none", display:
"flex", marginTop: "15px", justifyContent: "center"}}
```

```
<button className="loginRegisterButton">Log into Account</button>

            </Link>

          </form>

        </div>

      </div>

    </div>);

}
```

**Додаток Б**

## Лістинг коду домашньої сторінки

```
import Topbar from "../../components/topbar/Topbar";
import Sidebar from "../../components/sidebar/Sidebar";
import Feed from "../../components/feed/Feed";
import Rightbar from "../../components/rightbar/Rightbar";
import "./home.css"


export default function Home() {
  return (
    <>
      <Topbar />
      <div className="homeContainer">
        <Sidebar />
        <Feed/>
        <Rightbar/>
      </div>
    </>
  );
}
```

**Додаток В**

## Лістинг коду сторінки чатів

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

import Conversation from
"../../components/conversations/Conversation";

import Topbar from "../../components/topbar/Topbar";

import Message from "../../components/messege/Messege";

import axios from "axios";

import "./messenger.css";

import ChatOnline from "../../components/chatOnline/ChatOnline";

import { AuthContext } from "../../contex/AuthContext";

import { io } from "socket.io-client";

export default function Messenger() {

  const [converstation, setConverstation] = useState([]);

  const [currentChat, setCurrentChat] = useState(null);

  const [messages, setMessages] = useState([]);

  const [newMessage, setNewMessage] = useState("");

  const [arrivalMessage, setArrivalMessage] = useState([]);

  const [onlineUsers, setOnlineUsers] = useState([]);

  const socket = useRef("");

  const { user } = useContext(AuthContext);

  const scrollRef = useRef();

  useEffect(() => {

    socket.current = io("ws://localhost:8900");

    socket.current.on("getMessage", (data) => {
```

```
        setArrivalMessage({
          sender: data.senderId,
          text: data.text,
          createdAt: Date.now(),
        });
      });
    }, []);

    useEffect(() => {
      arrivalMessage &&
        currentChat?.members.includes(arrivalMessage.sender) &&
        setMessages((prev) => [...prev, arrivalMessage]);
    }, [arrivalMessage, currentChat]);

    useEffect(() => {
      socket.current.emit("addUser", user._id);
      socket.current.on("getUsers", (users) => {
        setOnlineUsers(
          user.followins.filter((f) => users.some((u) => u.userId ===
f))
        );
      });
    }, [user]);

    useEffect(() => {
```

```
    const getConverstation = async () => {
      try {
        const res = await await axios.get("/converstation/" +
user._id);
        setConverstation(res.data);
      } catch (error) {
        console.log(error);
      }
    };
    getConverstation();
  }, [user]);

  useEffect(() => {
    const getMessages = async () => {
      try {
        const res = await axios.get("/messages/" + currentChat?._id);
        setMessages(res.data);
      } catch (error) {
        console.log(error);
      }
    };
    getMessages();
  }, [currentChat]);

  useEffect(() => {
```

```
    scrollRef.current?.scrollIntoView({ behavior: "smooth" });
  }, [messages]);


  const handleSend = async (e) => {
    e.preventDefault();
    const message = {
      sender: user._id,
      text: newMessage,
      converstationId: currentChat._id,
    };


    const receiverId = currentChat.members.find((m) => m !==
user._id);
    socket.current.emit("sendMessage", {
      senderId: user._id,
      receiverId,
      text: newMessage,
    });
    try {
      const res = await axios.post("/messages", message);


      setMessages([...messages, res.data]);
      setNewMessage("");
    } catch (error) {
      console.log(error);
```

```
  }
};

return (
  <>
    <Topbar />
    <div className="messenger">
      <div className="chatMenu">
        <div className="chatMenuWrapper">
          <input
            type="text"
            placeholder="Пошук друзів"
            className="chatMenuInput"
          />
          {converstation.map((c) => (
            <div onClick={() => setCurrentChat(c)}>
              <Conversation conversation={c} currentUser={user} />
            </div>
          ))}
        </div>
      </div>
      <div className="chatBox">
        <div className="chatBoxWrapper">
          {currentChat ? (
```

```
                <>
                  <div className="chatBoxTop">
                    {messages.map((m) => (
                      <div ref={scrollRef}>
                        <Message message={m} own={m.sender === user._id}
/>
                      </div>
                    ))}
                  </div>
                  <div className="chatBoxBottom">
                    <textarea
                      placeholder="Написати повідомлення..."
                      className="chatMessageInput"
                      value={newMessage}
                      onChange={(e) => setNewMessage(e.target.value)}
                    ></textarea>
                    <button className="chatSubmitButtom"
onClick={handleSend}>
                      Надіслати
                    </button>
                  </div>
                </>
              ) : (
                <span className="noConverstation">
                  Open a conversation to start a chat
```

```
            </span>
          )}
        </div>
      </div>
      <div className="chatOnline">
        <div className="chatOnlineWrapper">
          <ChatOnline
            onlineUsers={onlineUsers}
            currentUserId={user._id}
            setCurrentChat={setCurrentChat}
          />
        </div>
      </div>
    </div>
  </>
);
```

**Додаток Г**

## Лістинг коду сторінки профілю

```
import { useState, useEffect } from "react";
import axios from "axios";

import "./profile.css";
import Topbar from "../../components/topbar/Topbar";
import Sidebar from "../../components/sidebar/Sidebar";
import Feed from "../../components/feed/Feed";
import Rightbar from "../../components/rightbar/Rightbar";
import { useParams } from "react-router";

export default function Profile() {
  const [user, setUser] = useState({});
  const params = useParams();
  useEffect(() => {
    const fetchUser = async () => {
      const res = await
axios.get(`/users?username=${params.username}`);
      setUser(res.data);
    };
    fetchUser();
  }, [params.username]);
  const PF = process.env.REACT_APP_PUBLIC_FOLDER;

  if (!user.username) {
```

```
  return null;
}
return (
  <>
    <Topbar />
    <div className="profile">
      <Sidebar />
      <div className="profileRight">
        <div className="profileRightTop">
          <div className="profileCover">
            <img
              src={
                user.coverPicture
                  ? PF + user.coverPicture
                  : PF + "person/noCover.png"
              }
              alt=""
              className="profileCoverImg"
            />
            <img
              src={
                user.profilePicrture
                  ? PF + user.profilePicrture
                  : PF + "person/noAvatar.png"
              }
```

```
                alt=""
                className="profileUserImg"
              />
            </div>
          </div>
          <div className="profileInfo">
            <h4 className="profileInfoName">{user.username}</h4>
            <h4 className="profileInfoDesc">{user.description}</h4>
          </div>
          <div className="profileRightBottom">
            <Feed username={params?.username} />
            <Rightbar user={user} />
          </div>
        </div>
      </div>
    </>
  );
}
```