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

Факультет інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка мережі соціальної взаємодії з використанням технологій React та Node.js

Виконав(ла): студент(ка) 4 курсу, групи СП-42
спеціальності 121
«Інженерія програмного забезпечення»
(шифр і назва спеціальності)

_____ Шиманський Р. В.
(підпис) (прізвище та ініціали)

Керівник _____ Мудрик І. Я.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю. М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М. Р.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2024

АНОТАЦІЯ

Кваліфікаційна робота на здобуття освітнього рівня бакалавра. Тернопільський національний технічний університет ім. І. Пулюя, кафедра програмної інженерії, спеціальність 121 «Інженерія програмного забезпечення». ТНТУ, 2024. Виконав студент групи СП-42 Шиманський Роман Вікторович.

Сторінок __, таблиць _, рисунків __, додатків _, презентація слайдів __.
Тема: Розробка мережі соціальної взаємодії з використанням технологій React та Node.js.

Веб-застосунок для комунікації та соціальної взаємодії, який є месенджером з розширеним функціоналом, залученням штучного інтелекту та спеціалізованими можливостями спілкування. Основою цього застосунку є чат з реальними користувачами та взаємодія з вбудованим штучним інтелектом. Застосування буде цікавим для різних соціальних верств населення та професій.

Для досягнення цілей в розробці обрано технологію React мовою програмування JavaScript, яка дозволяє створювати масштабовані веб-додатки із високою продуктивністю.

Першому розділі виконано аналіз предметної області, аналіз аналогів та конкурентів та вставнолено вимоги до системи типу «соцмережі».

У другому розділі виконано проєктування, конструювання та тестування програмної системи з демонстрацією результатів.

У третьому розділі описано основні принципи Охорони праці та безпеки життєдіяльності в галузі.

Ключові слова: розробка сайту, соцмережа, взаємодія, React, API, клієнт-сервер, фронтенд, розробка, Node.js.

ABSTRACT

A student Bachelor's certification work in Ternopil Ivan Pulyj National Technical University, department of software engineering, specialty 121 "Software engineering". TNTU, 2024.

Pages __, tables _, figures __, appendices _, slide presentation __.

Topic: Development of a social interaction network using React and Node.js technologies.

A web application for communication and social interaction, which is a messenger with advanced functionality, the involvement of artificial intelligence and specialized communication capabilities. The basis of this application is chat with real users and interaction with built-in artificial intelligence. The application will be interesting for different social strata of the population and professions.

To achieve the development goals, the React technology was chosen in the JavaScript programming language, which allows you to create scalable web applications with high performance.

The first section performed an analysis of the subject area, an analysis of analogues and competitors, and inserted requirements for a "social network" type system.

In the second section, the design, construction and testing of the software system with the demonstration of the results was performed.

The third chapter describes the main principles of labor protection and life safety in the industry.

Keywords: site development, social network, interaction, React, API, client-server, frontend, development, Node.js.

ЗМІСТ

ВСТУП

Перш ніж розпочинати розробку, важливо чітко визначити свої цілі та пріоритети для соціальної мережі. Це допоможе вам зосередитися на тому, що важливо, і уникнути витрачання часу на розробку функцій, які не потрібні користувачам. Розробка власної соціальної мережі - це складний, але потенційно дуже корисний проект, який потребує ретельного планування та виконання.

Ця дипломна робота присвячена розробці веб-сайту типу «соцмережі» для соціальної взаємодії групи осіб з спільними інтересами (наприклад для іт-шників). з використанням технологій веб-розробки React, API, Node.js та архітектури клієнт-сервер. [2] Основні функції включають обмін файлами, комунікацію в реальному часі, інтеграцію з іншими інструментами для спілкування та команди.

Метою цієї роботи є створення веб-додатку власної соцмережі як платформи для спілкування та обміну інформацією між людьми навколо спільних інтересів.

Завдання в процесі створення соціальної мережі передбачають аналіз конкурентів та вимог до системи (функціональний та нефункціональних), проєктування системи, конструювання клієнтської та серверної частини, розгортання та тестування реалізованих рішень. Також частина завдань передбачає виконання творчої роботи – моделювання та реалізацію користувацького інтерфейсу.

Розроблений додаток повинен враховувати вимоги потенційних користувачів і цільових аудиторій аби забезпечити умови використання. Особливу увагу варто зосередити на універсальності розробки та якісних характеристиках, які забезпечить ефективне та різнобічне тестування.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

У наш час соціальні мережі стали невід'ємною частиною життя мільярдів людей. Вони дають можливість спілкуватися з друзями та родичами, ділитися новинами та думками, знаходити однодумців, а також стежити за подіями, що відбуваються у світі.

Для ефективного спілкування та зручності використання веб-сервісу варто ретельно аналізувати діяльність основних конкурентів та існуючі продукти на ринку. Також варто порівняти їхні переваги на недоліки для виділення ключових позицій, які б могли покращити роботу розроблюваної системи.

Призначення розроблюваного ПЗ - забезпечення швидкого обміну повідомленнями, файлами великого розміру, роботою з штучним інтелектом. [6]

Застосунок представлятиме собою платформу для спілкування та спільної роботи, яка поєднуватиме в собі можливості чату, онлайн-дошки для конференцій. Основні функції застосунку будуть наступними:

- управління соціальною мережею;
- спрощення взаємодії та обміну інформацією між учасниками;
- публікація коротких повідомлень та взаємодія з ними;
- обмін данними між користувачами системи.

До характеристик системи можна віднести:

- Користувачі: зареєстровані користувачі з особистими профілями.
- Публікації: можливість створення публікацій до 120 символів з можливістю лайків та коментування.
- Лайки: користувачі можуть лайкати публікації інших користувачів.
- Аутентифікація. Вхід та перевірка існування користувача, даних для авторизації.
- REST API: весь функціонал системи надається через REST API.

Основне вікно складається зі списку чатів або каналів, що дозволяє вибирати потрібний, і панелі користувацьких інструментів (з вибором візуального

оформлення програми, статусу користувача, пошуку користувачів і кнопки налаштувань). Чат представляє собою список повідомлень, статус користувача, з яким спілкуєшся, і поле для введення повідомлень.

Додаток поєднує в собі особливості Твітера і Телеграму, надаючи можливість студентам обмінюватися ідеями, думками, досвідом та навчальними матеріалами. Він дозволяє створювати публікації, ділитися посиланнями, відправляти особисті повідомлення, створювати групові чати та спільноти за інтересами, а також отримувати оновлення від інших студентів і професіоналів з різних галузей. [5]

Веб-додатки стають все більш популярними завдяки своїм численним перевагам. Їхня гнучкість, простота використання та доступність роблять їх ідеальним рішенням для багатьох завдань.

Веб-додатки мають ряд суттєвих переваг перед традиційними програмами, зокрема:

Постійне оновлення: Завдяки веб-технологіям, веб-додаток можна легко оновлювати на сервері, без необхідності завантаження та встановлення нових версій користувачами. Це гарантує, що користувачі завжди мають доступ до найновіших функцій та виправлень помилок.

Сумісність: Веб-додатки не залежать від жодної конкретної операційної системи або пристрою. Їх можна використовувати на будь-якому комп'ютері, планшеті чи смартфоні з доступом до глобальної мережі. Це робить їх зручними для користувачів, яким не потрібно турбуватися про те, чи сумісний додаток з їхнім пристроєм.

Ці переваги роблять веб-додатки чудовим вибором для компаній та організацій, які прагнуть:

- забезпечити користувачам постійний доступ до оновлених функцій та можливостей.
- досягти ширшої аудиторії, адже їх можна використовувати на будь-яких пристроях.

- знизити витрати на розробку та обслуговування програмного забезпечення, адже не потрібно створювати та підтримувати окремі версії для різних платформ.

Швидкість роботи веб-додатку є його сильною стороною. Сторінки завантажуються швидко, без помітних затримок, що забезпечує плавний та приємний досвід користування. Оптимізація під мобільні пристрої робить веб-додаток зручним для використання на смартфонах та планшетах. Інтерфейс адаптується до розміру екрану, а елементи управління легко доступні. [8]

В сукупності всі ці чинники роблять цю предметну область важливою для вивчення в навчальних та практичних цілях. Розробка додатку типу соціальної мережі може мати значний вплив на суспільство, економіку та технології, надаючи корисні інструменти для різноманітних потреб користувачів. [7]

1.2 Аналіз ринку та існуючих розробок

Оскільки обрана предметна область є доволі важливою у сучасному суспільстві, готові рішення є розповсюджені на ринку.

Однак розробка успішного додатку типу соцмережі - це складне завдання, яке потребує значних інвестицій та зусиль. Важливо провести ретельне дослідження ринку, визначити цільову аудиторію, розробити унікальну пропозицію та створити зручний та інтуїтивно зрозумілий інтерфейс.

Рисунок 1.1 – Найпопулярніші додатки для спілкування в світі 2023

Аналогів є доволі багато, проте у них відрізняються спільні між собою функції. Зважаючи на поставлені вимоги до проекту, можна проаналізувати декілька конкурентів у цій сфері. Для аналізу було обрано найпопулярніші варіанти мереж соціальної взаємодії:

1. Назва: Telegram
   - Розробник: Telegram Messenger LLP, Telegram FZ LLC.
   - Архітектура: Кросплатформений месенджер (мобільні та веб-додатки). існують версії для мобільних та десктопних пристроїв, також браузерна, що є клієнтськими складовими, які взаємодіють з серверними
   - Мова реалізації: Python, C++, Java, JavaScript

     Перелік функцій та характеристик:

     - Обмін файлами;
     - Групові чати, канали;
     - Дзвінки;
     - Чат-боти;
     - Наявність наліпок та іншого користувацького вмісту.

До переваг варто віднести: високий рівень шифрування для забезпечення конфіденційності даних; широкі можливості відправки різноманітного контенту;

зручний інтерфейс та висока швидкість доставки повідомлень; підтримка ботів для різних цілей: від розваг до бізнес-процесів.

1) Шифрована переписка один до одного та у групових чатах.
2) Можливість відправки повідомлень, фото, відео, аудіо та файлів різних форматів.
3) Створення групових чатів до 200 000 учасників.
4) Користувальницькі профілі з ім'ям, прізвищем, місцем проживання.
5) Канали для масової розсилки інформації та сповіщень.
6) Використання ботів для автоматизації дій та отримання інформації.
7) Відео та голосові дзвінки з криптографічним шифруванням.
8) Секретні чати з автоматичним видаленням повідомлень.
9) Можливість використання на різних пристроях одночасно.
10) Хмарне збереження файлів;
11) Таймлайн з оновленнями від підписаних користувачів.
12) Швидка доставка повідомлень;
13) Велика популярність в Україні та світі.

Недоліки Telegram наступні:

1) Обмежена інтеграція з іншими сервісами порівняно з деякими конкурентами;
2) Обмежені можливості рекламної та бізнесової активності;
3) Обмежений розмір обмінюваних файлів (до 2Гб);
4) Наявність платного вмісту.

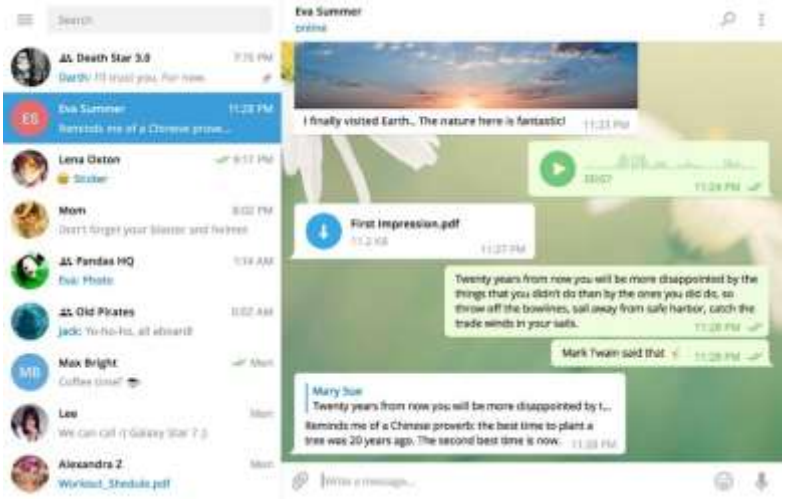


Рисунок 1.2 – Вигляд вікна програми Telegram

2. Назва: Facebook Massanger від розробників Meta Platforms, Inc.

Архітектура: Веб-додаток, мобільні додатки. Мова реалізації: PHP, Hack, JavaScript.

Перелік функцій та характеристик:

1. Профілі користувачів з інформацією про ім'я, прізвище, місце проживання.

2. Можливість публікувати пости до 63 206 символів, включаючи зображення та відео.

3. Лайки, коментарі та спільна робота з публікаціями.

4. Аналітика активності користувачів та рекламних кампаній.

5. Аутентифікація через JWT.

Аналіз переваг та недоліків:

- Переваги Massanger: Велика активна аудиторія, розвинуті можливості рекламування, широкий функціонал, активна аудиторія.
- Недоліки Massanger: Суттєва конкуренція, складність управління приватністю даних, залежність від алгоритмів сортування контенту, недостатній контроль над спамом та недостовірною інформацією.

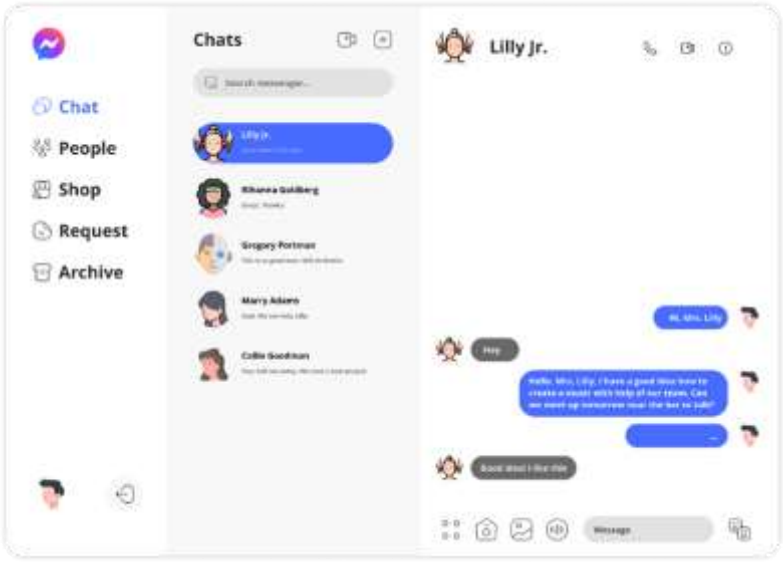


Рисунок 1.3 – Вигляд вікна програми Telegram

3. Назва: Viber від Розробників: Viber Media, Inc.

Архітектура: існують версії для мобільних та десктопних пристроїв, що є клієнтськими складовими, які взаємодіють з серверними.

Мова реалізації: Java, C, Python, C++ і Objective C.

Перелік функцій, характеристик:

- Обмін файлами;
- Групові чати, канали (менший функціонал, ніж у Telegram);
- Дзвінки;
- Чат-боти (обмеженіші ніж у Telegram);
- Наявність наліпок та іншого користувацького вмісту.

Аналіз переваг та недоліків:

Переваги:

- Інтуїтивний інтерфейс.

Недоліки:

- Нестабільна робота;
- Погана оптимізація;
- Великий обсяг додатку;
- Проблеми хмарного збереження;
- Мала кількість можливостей інтеграції з іншими додатками чи сайтами;

- Наявність великої кількості рекламного вмісту;
- Велика кількість спаму. Недостатній контроль над спамом та недостовірною інформацією.

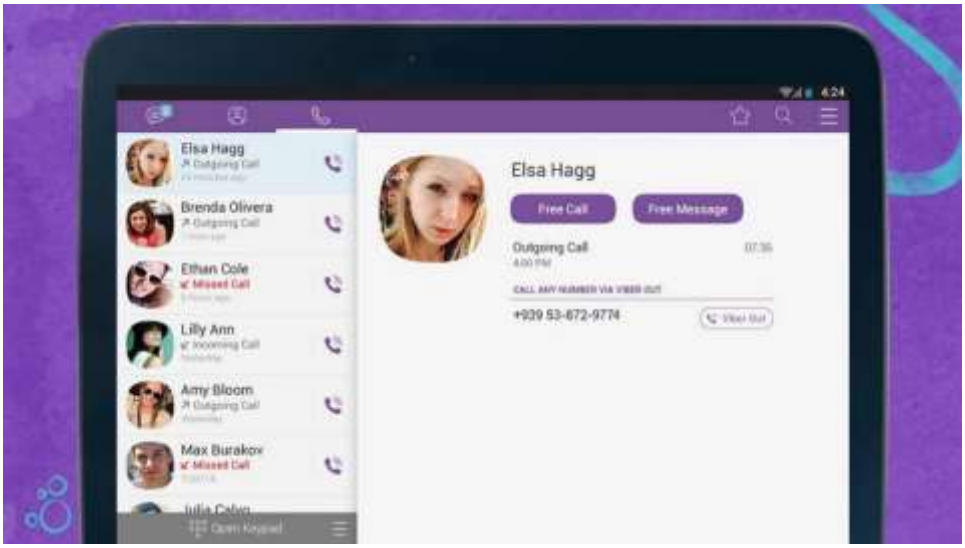


Рисунок 1.4 – Вигляд вікна програми Viber

4. Назва: GPT-Chat. Чат-бот, який є яскравим і вдалим прикладом моделі штучного інтелекту із великою базою даних. Розробник: Open AI.

Архітектура: веб-сайт відіграє роль клієнтської складової, запити обробляються на сервері.

Мова реалізації: Python.

Перелік функцій, характеристик:

- Підтримка розмови на будь-які теми;

- Платна версія має можливості працювати із посиланнями, зображеннями.

- Спеціалізація в широкій сфері;

- Генерація тексту;

- Навчання моделі

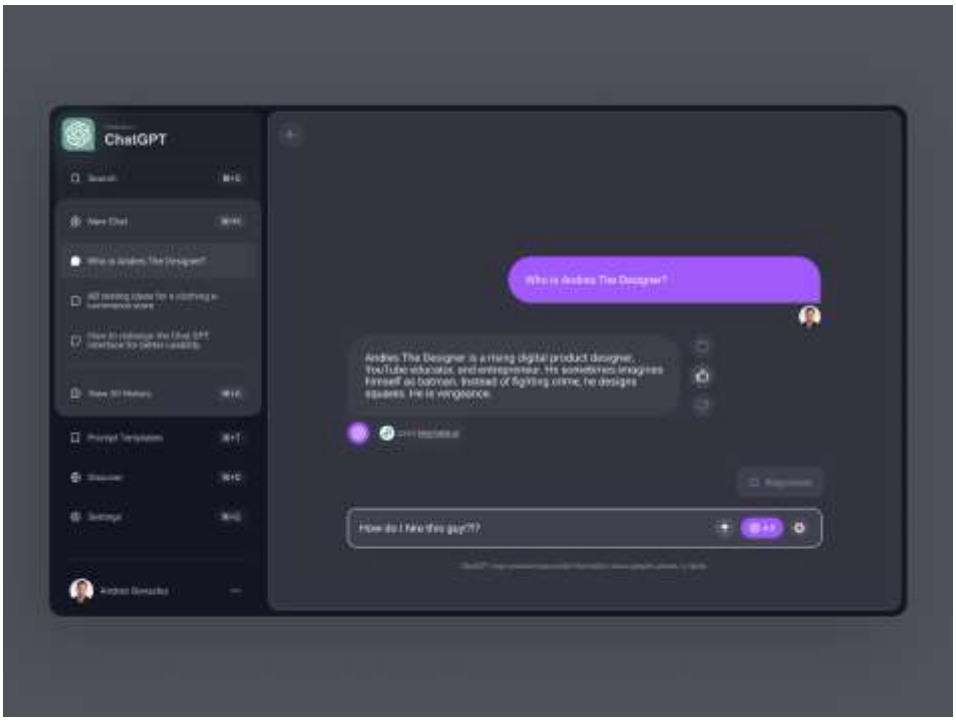


Рисунок 1.5 – Вигляд вікна програми GPT-chat

Аналіз переваг та недоліків:

Переваги: велика база даних, надання корисної інформації (якщо вона не є новинкою), швидкодія, відповідає в реальному часі.

Недоліки: перенавчання моделі; робота із нетекстовими даними доступна лише після отримання платної версії; не завжди можна отримати достовірну чи правильну відповідь; застаріла база даних в безкоштовній версії.

1.3 Обґрунтування вибору архітектури та технологій реалізації

Вибір технологій для реалізації клієнтської частини системи соцмережі з використанням React має кілька вагомих переваг, які роблять цей фреймворк оптимальним вибором [4].

Для написання веб-додатку обрано наступні технології:

- Мова програмування: Java-Script.
- Frontend: React, Redux.
- Backend: Node.js, Express, Mongoose, Socket.io.

Ось основні обґрунтування вибору React:

1) Компонентний підхід. React дозволяє розбивати інтерфейс на малі, повторно використовувані компоненти. Це сприяє кращій організації коду та його масштабованості. Компоненти можна легко комбінувати та повторно використовувати в різних частинах додатку.
2) Висока продуктивність. React використовує віртуальний DOM (Virtual DOM) [1], який мінімізує операції прямого маніпулювання DOM. Це забезпечує високу продуктивність додатку, оскільки зміни в інтерфейсі обробляються швидше.
3) Сильна підтримка та велика спільнота. React активно підтримується Facebook і має велику спільноту розробників. Це означає, що для React доступно багато бібліотек, інструментів та навчальних матеріалів, що полегшує розробку та вирішення проблем.
4) Екосистема та інтеграція. React легко інтегрується з іншими бібліотеками та фреймворками. Для управління станом можна використовувати Redux або MobX, для маршрутизації - React Router, для асинхронних запитів - Axios тощо.
5) Легкість навчання. React має відносно невелику криву навчання. Основні концепції, такі як компоненти, стани та властивості, легко зрозуміти навіть новачкам. Крім того, велика кількість документації та прикладів допомагають швидко опанувати фреймворк.
6) Сумісність з TypeScript. React добре сумісний з TypeScript, що дозволяє додати статичну типізацію в JavaScript-код. Це покращує читабельність та надійність коду, зменшуючи кількість потенційних помилок.

7) SEO-оптимізація. З допомогою бібліотеки Next.js можна створювати серверні рендеринг додатки на базі React, що покращує SEO-оптимізацію та швидкість завантаження сторінок.
8) Гнучкість та налаштування. React надає розробникам велику гнучкість у виборі архітектури та підходів до розробки. Це дозволяє адаптувати додаток під конкретні потреби проекту.
9) Розширюваність: За допомогою бібліотек і фреймворків, таких як Redux або MobX, можна забезпечити розширюваність та масштабованість додатку, особливо при роботі з великим обсягом даних та функціоналом.

React є потужним, продуктивним і гнучким інструментом для розробки клієнтських додатків. Його компонентний підхід, висока продуктивність, велика спільнота та легкість навчання роблять його оптимальним вибором для реалізації системи соцмережі. Вибір React забезпечує не лише зручність розробки, але й можливість масштабування та підтримки додатку у довгостроковій перспективі.

Написання соціальної мережі за допомогою React є зручним. React надає можливість ефективного відображення даних, зокрема профілів користувачів, новин, фотографій тощо, що є ключовими елементами соціальної мережі [4].

Завдяки можливостям React, можна створювати динамічні інтерфейси, такі як реактивні елементи, анімація та жива оновлення даних без перезавантаження сторінки. Управління станом, де React пропонує зручні засоби для управління станом додатку, наприклад, за допомогою контексту, хуків та управління станом компонентів. Також React дозволяє створювати реактивні елементи інтерфейсу, що реагують на зміни стану, вводу користувача та інших подій без зайвих зусиль [5].

Загалом, використання React для написання соціальної мережі дозволяє створити сучасний, швидкий та функціональний додаток з багатим функціоналом і зручним інтерфейсом для користувачів.

Для серверної частини розроблюваного ПЗ використано технології Node.js, яка є потужною технологією для розробки серверної частини додатків, і його вибір для розробки соцмережі має кілька важливих переваг.

Node.js використовує однопоточну архітектуру з подіями, що дозволяє ефективно обробляти багато одночасних запитів без блокування. Асинхронна природа Node.js робить його ідеальним для додатків з високою навантаженістю, таких як соцмережі, де одночасно можуть відбуватися численні запити на читання та запис.

Також, Node.js побудований на V8 JavaScript engine [6], що забезпечує високу продуктивність і швидкість виконання коду. Це важливо для обробки великої кількості запитів та взаємодій у соціальній мережі. Завдяки використанню JavaScript/TypeScript як на серверній, так і на клієнтській частині, Node.js дозволяє розробникам використовувати один стек технологій. Це спрощує процес розробки, покращує продуктивність команди та зменшує кількість помилок, пов'язаних з контекстними перемиканнями між різними мовами програмування.

Node.js має величезний репозиторій пакетів npm (Node Package Manager), що містить понад мільйон бібліотек та модулів. Це дозволяє швидко знаходити та інтегрувати необхідні інструменти та функціональність, скорочуючи час на розробку та підвищуючи ефективність. Node.js легко масштабувати як вертикально, так і горизонтально. Він підтримує розподілену архітектуру і може легко масштабуватися за допомогою кластеризації або розгортання на різних серверах.

Соцмережі часто потребують реалізації функцій реального часу, таких як чати, повідомлення або оновлення новинних стрічок. Node.js має вбудовану підтримку WebSockets, що дозволяє створювати додатки з функціональністю реального часу ефективно та просто. Node.js добре підходить для реалізації мікросервісної архітектури, що дозволяє розбити додаток на незалежні сервіси. Це покращує модульність, спрощує підтримку та забезпечує можливість розгортання окремих компонентів незалежно один від одного.

Node.js активно підтримується спільнотою та великими компаніями, такими як Google, Microsoft та Amazon. Це забезпечує регулярні оновлення, покращення безпеки та нові функціональні можливості.

Загалом, Node.js є відмінним вибором для розробки серверної частини соціальної мережі завдяки своїй високій продуктивності, асинхронності, масштабованості та великій екосистемі. Його використання забезпечує швидкий і ефективний розвиток, підтримку функцій реального часу та можливість розробки повного стеку на одній мові програмування, що значно спрощує процес розробки та підтримки додатку.

Засоби реалізації архітектури додатку «соцмережі»:

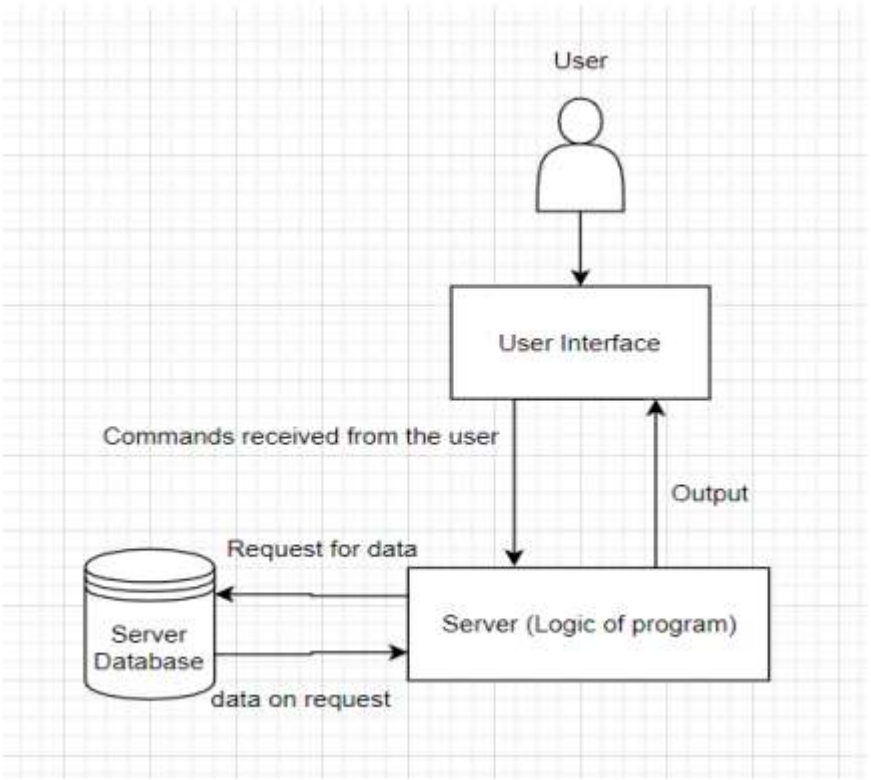


Рисунок 1.2 – Структура схема розроблюваного ПЗ

Головне вікно складається із списку чатів/каналів, відповідно, можна вибирати певний, і панелі користувацьких інструментів (із вибором інтерфейсу програми, стану користувача, пошуку користувачів, і кнопки налаштування). Чат являє собою список повідомлень, стан користувача, з яким спілкуєшся, і поле вводу повідомлення.

Варто врахувати до уваги з самого початку, що ці функції зазвичай мають вищу складність та вартість реалізації: інтеграція API сторонніх розробників, платіжні інтеграції, шифрування бази даних, розширена панель адміністратора з

потужною аналітикою та численними функціями, спеціальна механіка взаємодії між користувачами (точнісінько як у соціальних мережах).

Типові випадки використання системи:

1) Створення нової публікації:

- Користувач авторизується.
- Обирає опцію "Створити нову публікацію".
- Вводить текст публікації та натискає кнопку "Опублікувати".

2) Лайк публікації:

- Користувач переглядає публікацію.
- Натискає кнопку "Лайк" для публікації, яку бажає підтримати.

3) Коментування публікації:

- Користувач переглядає публікацію.
- Відкриває форму коментаря та вводить текст.
- Натискає кнопку "Надіслати" для опублікування коментаря.

4) Звідний обмін даними:

- Користувач вибирає опцію "Надіслати файл" у чаті.
- Обирає файл зі свого пристрою (текстовий документ, зображення тощо).
- Натискає кнопку "Відправити".
- Передає файл через мережу Інтернет на сервер.
- Сервер приймає файл та зберігає його у відповідному сховищі даних.
- Отримувач отримує повідомлення про надходження файлу у чаті.
- Він може завантажити та переглянути отриманий файл.

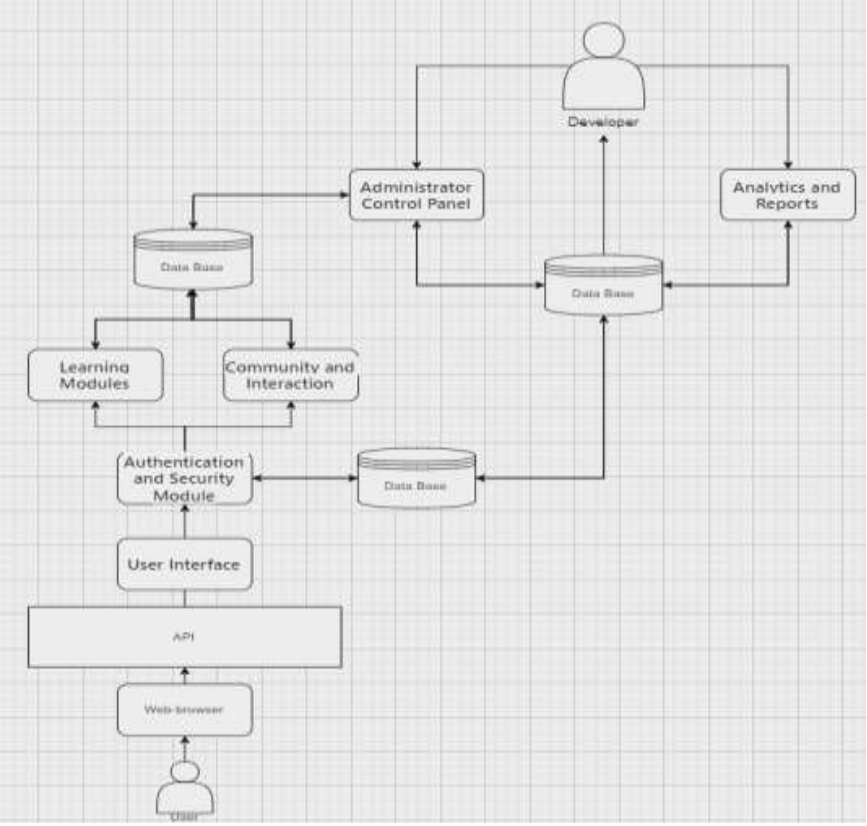


Рисунок 1.6 – Типовий приклад використання

1.4 Опис вимог до системи та користувачів

Користувачі системи виявлено наступні:

1) Адміністратор каналу/чату, що створив чат, наділений правами змінювати їхню структуру, ієрархію її учасників та склад. Відповідає за склад каналу/групи, дії модераторів, вміст.
2) Модератор чату. користувач системи, учасник чату, якому адміністратор надав права модератора каналу/чату. Відповідає за контроль та модерацію обговорень в чатах, дотримання його учасниками правил спілкування, вміст каналу.

3) Користувач системи, можливий учасник каналів/чатів або їхній адміністратор чи модератор. Відповідає за обмін повідомленнями, створення або управління чатами/каналами.
4) Неавторизований користувач: це користувач, який має змогу зареєструватися чи авторизуватися та переглянути оголошення з головної сторінки.

Функціонал системи побудований за форматом перехід між вікнами. Головне вікно складається із списку каналів чатів. Відповідно можна вибирати певний, і панелі користувацьких інструментів (із вибором візуалу програми, стану користувача, пошуку користувачів, і кнопки налаштунків).

Чат являє собою список повідомлень, стан користувача, з яким спілкуєшся, і поле вводу повідомлення.

Функціональні можливості роботи системи наступні:

1) Користувач починає новий чат, ввівши його унікальне ім’я в програмі у панель пошуку, і написавши повідомлення. Розв’язується діалог, а згодом з’являється потреба у надсиланні великорозмірного файлу.
2) Надсилання, вкладення успішно доставляється (вибравши відповідні параметри).
3) Чат користувачів вирішив спільно переглянути подію. Адміністратор (або модератор, чи навіть простий користувач) чату починає трансляцію, ввівши посилання на ресурс, або транслюючи власний екран.
4) Користувач створює канал, натиснувши на відповідні кнопки. Згодом налаштовує його і додає вміст, приймає або відхиляє нових учасників.
5) Користувач вирішив переглянути ситуацію на ринку і для цього звернувся до спеціального чат-бота, який, відповідно, йому дасть власну, натреновану великою кількістю даних, інформацію по темі спілкування.

1.5 Розробка концептуальної моделі предметної області

Призначення програмного забезпечення: забезпечити швидкий обмін повідомленнями, передачу файлів великого розміру, а також роботу зі штучним інтелектом.

Основне вікно складається зі списку чатів або каналів, що дозволяє вибирати потрібний, і панелі користувацьких інструментів (з вибором візуального оформлення програми, статусу користувача, пошуку користувачів і кнопки налаштувань). Чат представляє собою список повідомлень, статус користувача, з яким спілкуєшся, і поле для введення повідомлень.

Планування передбачає визначення функціоналу: вхід/реєстрація, створення та перегляд профілів, додавання друзів, створення постів, коментування, лайки, чати та повідомлення.

Створення моделі та макету: інструменти для створення прототипів, такі як Figma або Sketch, щоб візуалізувати інтерфейс користувача.

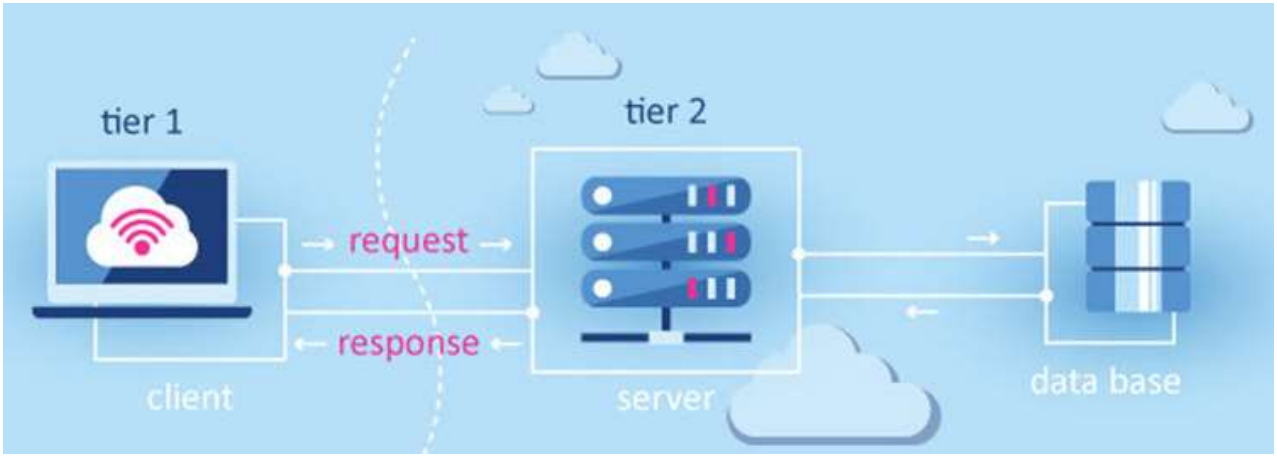


Рисунок 1.7 – Схема багатошарової архітектури

2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО КОМПЛЕКСУ

2.1 Проєктування програмного забезпечення

Опис архітектурної моделі проекту представляє собою детальний план, який описує структуру, компоненти та взаємозв'язки системи. Він слугує дорожньою картою для розробників, гарантуючи, що всі частини системи працюють узгоджено та ефективно.

Опис архітектурної моделі містить загальний огляд системи: Опис мети, функціональності та ключових характеристик системи. Логічна архітектура - опис компонентів системи, їх функцій та взаємодій. Фізична архітектура - опис апаратного та програмного забезпечення, на якому буде працювати система. Опис технологій, які будуть використовуватися для розробки та реалізації системи.

Проєктування системи соцмережі з використанням React починається з аналізу вимог та планування, включаючи визначення основних функцій (реєстрація користувачів, створення постів, коментування, лайки тощо). Далі створюються ескізи та прототипи для UI/UX дизайну. Використовується технологічний стек з React для фронтенду, Node.js з Express для бекенду, і MongoDB для бази даних. Реалізуються основні компоненти та налаштовується інтеграція з бекендом, а також забезпечується аутентифікація через JWT. Проект тестується, деплоїться на платформу хостингу (наприклад, Heroku), і налаштовується моніторинг та зворотний зв'язок для подальшого покращення і масштабування [5].

Підхід до виконання проекту визначає, як буде реалізовано проект. Він описує процеси, методи та інструменти, які будуть використовуватися для досягнення цілей проекту.

2.2 Підхід до виконання проекту

Методологія управління проектами: Agile методологія поділяє процес розробки на короткі ітерації, що дозволяє швидко отримувати відгуки користувачів та вносити зміни до продукту. Це особливо важливо для соціальних мереж, які постійно розвиваються та адаптуються до нових тенденцій. Ця методологія зосереджується на створенні функцій, які мають найбільшу цінність для користувачів. Це допомагає команді розробників зосередитися на тому, що важливо, і уникнути витрачання часу. Agile методологія включає в себе постійне вдосконалення продукту. Основні риси цієї методології, які впливають на успішність проєкту розробки власної соцмережі наступні:

- управління ризиками: Опис потенційних ризиків проекту та стратегій їх пом'якшення.
- управління ресурсами: Опис того, як будуть виділятися та використовуватися ресурси проекту.
- управління комунікаціями: Опис того, як буде здійснюватися комунікація між учасниками проекту.
- управління зацікавленими сторонами: Опис того, як буде управлятися взаємодією з зацікавленими сторонами проекту.

Опис архітектурної моделі та підхід до виконання проекту є важливими документами, які гарантують успішне виконання проекту. Вони забезпечують чітке розуміння цілей проекту, його структури, процесів та ресурсів, необхідних для його досягнення.

Був проведений аналіз головних акторів системи. Аналіз для встановлення ролей в системі включав в себе докладне вивчення функціональностей та обов'язків, які пов'язані з цими ролями. В якості основних кандидатів для ролей в системі було обрано Незареєстрованого користувача(Гостя), Зареєстрованого користувача та розробника. Акторів системи зображено на рисунку 2.1.

Для візуального моделювання веб-додатку використано UML-діаграми.

Діаграма Use Case для веб-застосунку демонструє різні сценарії взаємодії між користувачами системи та самою системою. На діаграмі продемонстровано основних акторів: Адміністратор і Користувач.

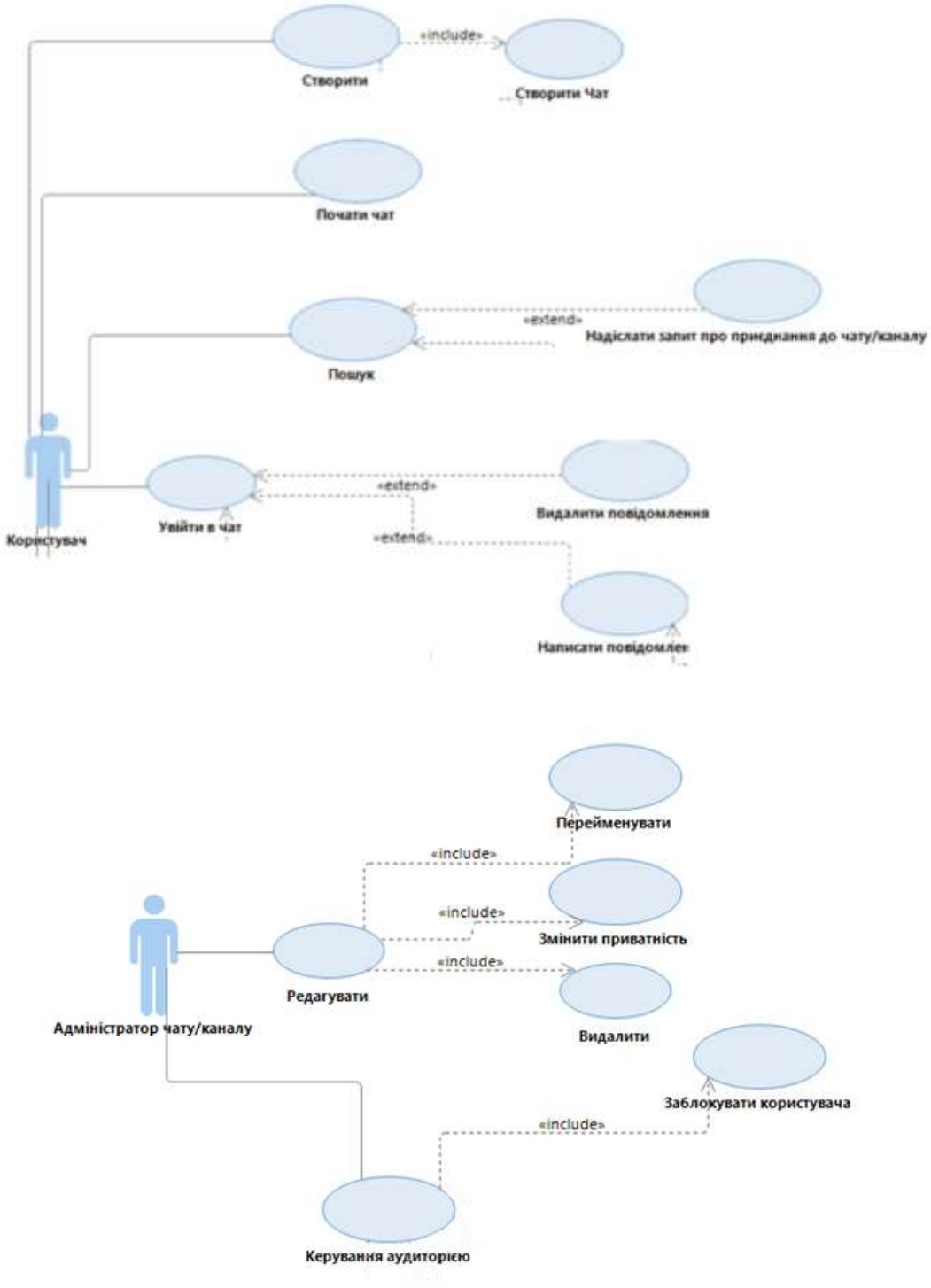


Рисунок 2.1 – Діаграма варіантів використання розроблюваної системи для користувачів

Сценарії до базових Usecases:

1) Користувач вибрав опцію створити чат/канал. Йому потрібно ввести його назву, тип(закритий/відкритий), після чого йому присвоюється статус Адміністратора чату/каналу.
2) Користувачу не сподобалось власне повідомлення і він вирішив його видалити. Для цього в контекстному меню тексту вибрав опцію видалення повідомлення, яке було успішно видалено.
3) Користувач захотів вийти з чату/каналу. Для цього вибрав в опціях “Покинути”. Користувача видалено із учасників чату/каналу.
4) Користувач вибрав опцію почати новий чат. Він увів в пошукове поле нік користувача, з яким йому слід почати спілкування, і написав повідомлення, тим самим почавши чат.
5) Користувач, зайшовши в груповий чат, написав повідомлення, ввівши його у поле для вводу повідомлень. Повідомлення успішно надіслалось усім учасникам чату.
6) Користувач вводить в пошукове поле посилання на чат/канал, якщо їм присвоєно тип “Відкритий”, то після приєднання він стає його учасником, інакше, адміністратор вибирає чи прийняти, чи відхилити запит про приєднання користувача. У разі відмови користувача не буде додано до чату/каналу, і не буде повідомлено йому про це.
7) Користувач, зайшовши в груповий чат, написав повідомлення у вигляді відповіді на інше повідомлення, ввівши його у поле для вводу повідомлень і вибравши цільове. Повідомлення успішно надіслалось усім учасникам чату.
8) Користувач вибрав опцію “домашній кінотеатр” починає трансляцію, ввівши посилання на відеоресурс. Далі запрошує коритсувачів. Вони успішно приєднались і переглядають фільм.
9) Користувач натиснув на опцію “Надіслати файл” біля поля вводу повідомлень. Вибрав ціль і натиснув кнопку для надсилання. Файл через певний проміжок часу було успішно надіслано.

10) Адміністратор чату вирішив зняти обмеження на заборону спілкування Користувача, для цього заходить в доступну лише йому панель керування каналом/чатом і серед обмежених Користувачів вибирає потрібного і знімає обмеження.
11) Адміністратор чату побачив повідомлення невідповідного вмісту. Вибравши відповідну дану йому опцію, обмежив введення його повідомлень на певний проміжок часу.
12) Адміністратор чату побачив повідомлення невідповідного вмісту. Вибравши відповідну дану йому опцію, позбавив його доступу до чату.
13) Адміністратор чату має намір призначити модератора чату. Вибрав відповідного користувача і вибравши потрібно опцію, користувач стає модератором чату.
14) Адміністратор чату/каналу має намір змінити назву, для цього заходить в доступну лише йому панель керування каналом/чатом і вводить в поле назви її оновлену версію, після чого повертається на головне меню каналу/чату із уже новим ім’ям.
15) Адміністратор чату/каналу має намір змінити тип приватності, для цього заходить в доступну лише йому панель керування каналом/чатом і вибирає відповідний пункт (Закритий/Відкритий), після чого тип доступу успішно змінений.
16) Адміністратор чату/каналу має намір видалити його, для цього заходить в доступну лише йому панель керування каналом/чатом і вибирає відповідну опцію, після чого чат/канал видаляється НАЗАВЖДИ.
17) Адміністратор чату/каналу вирішив розблокувати Користувача в чаті/каналі, для цього заходить в доступну лише йому панель керування каналом/чатом і вибирає серед списку заблокованих користувачів потібного і натиснувши на відповідну опцію, Користувача повернено на чат/канал, а у нього він зявився у списках.
18) Адміністратор чату вирішив позбавити Модератора цього статусу , для цього заходить в доступну лише йому панель керування чатом, і серед

списку модераторів обирає потрібного і видаляє, статус змінено на учасника.

2.3 Проєктування архітектури

Створення додатку соціальної мережі з використанням React включає декілька основних кроків: планування, налаштування середовища розробки, створення основної архітектури, реалізацію функціоналу, налаштування бекенду та тестування.

Одним з ключових інструментів при проєктуванні є діаграма класів, яка надає візуальне представлення структури системи та її компонентів.

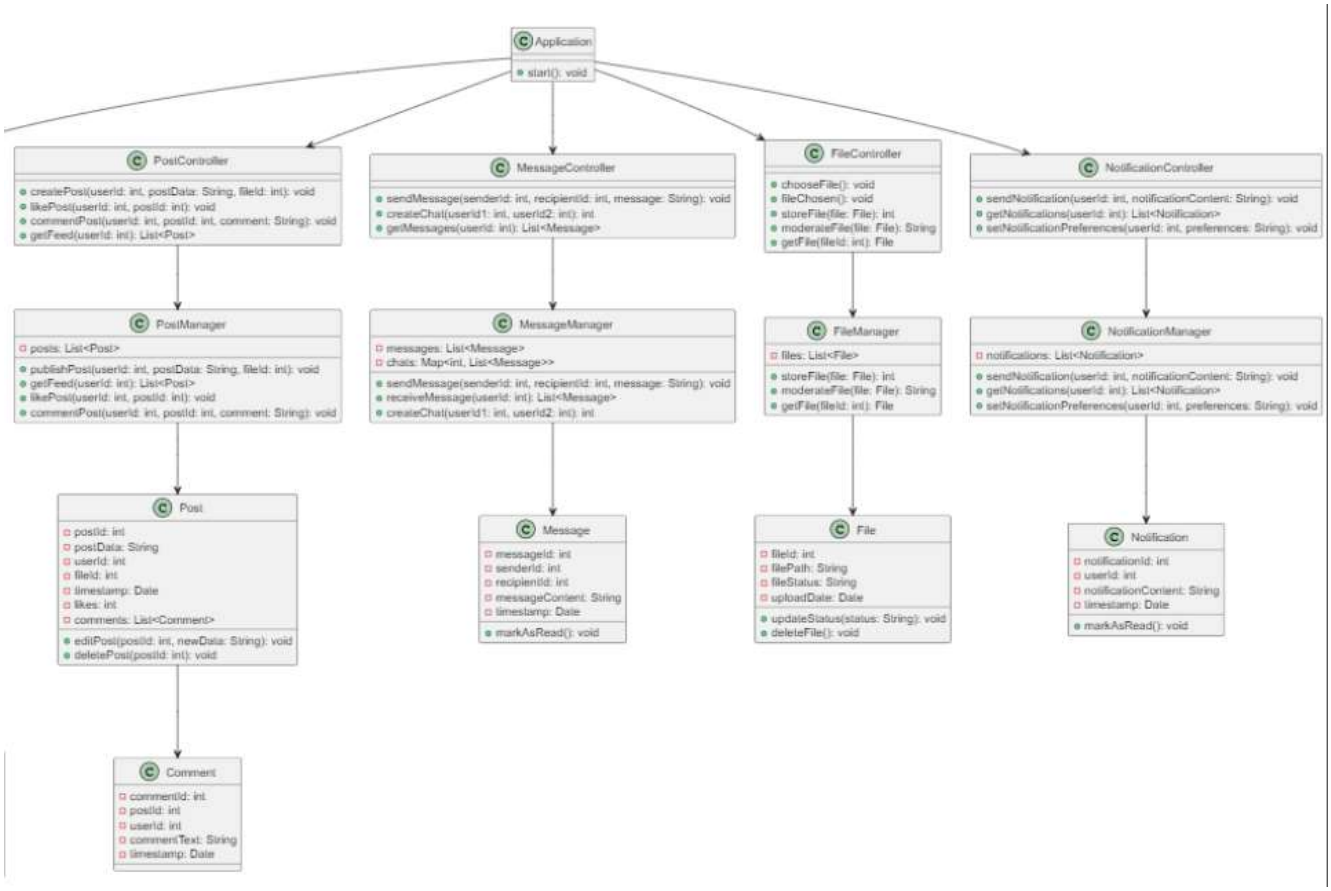


Рисунок 2.2 Фрагмент «Діаграма Класів» для серверної частини системи

Для проектованого веб-додатку основними варіантами використання є написання повідомлення та відправка його в чаті. Далі продемонстровано обєкти в системі для реалізації. В системі втзначено наступні класи:

Application - клас, що запускає та координує роботу різних компонентів системи. Методи: • start(): Запускає додаток.

UserController- Клас, що відповідає за управління користувачами. Методи: registerUser(username: String, password: String): bool: Реєструє нового користувача. loginUser(username: String, password: String): bool: Авторизує користувача. editProfile(userId: int, profileData: String): void: Редагує профіль користувача. addFriend(userId: int, friendId: int): void: Додає друга до списку друзів. searchUser(query: String): List<int>: Шукає користувачів за запитом. createGroup(userId: int, groupName: String): int: Створює нову групу.

PostController - клас, що відповідає за управління постами. Методи: createPost(userId: int, postData: String, fileId: int): void: Створює новий пост. likePost(userId: int, postId: int): void: Додає лайк до посту. commentPost(userId: int, postId: int, comment: String): void: Додає коментар до посту. getFeed(userId: int): List<Post>: Отримує стрічку новин користувача.

MessageController - клас, що відповідає за управління повідомленнями. Методи: sendMessage(senderId: int, recipientId: int, message: String): void: Надсилає повідомлення. createChat(userId1: int, userId2: int): int: Створює новий чат. getMessages(userId: int): List<Message>: Отримує повідомлення користувача.

FileController - клас, що відповідає за управління файлами. Методи: chooseFile(): void: Вибирає файл. fileChosen(): void: Підтверджує вибір файлу. storeFile(file: File): int: Зберігає файл. moderateFile(file: File): String: Проводить модерацію файлу. getFile(fileId: int): File: Отримує файл за ідентифікатором.

NotificationController - клас, що відповідає за управління сповіщеннями. Методи: sendNotification(userId: int, notificationContent: String): void: Надсилає сповіщення. getNotifications(userId: int): List<Notification>: Отримує сповіщення користувача. setNotificationPreferences(userId: int, preferences: String): Встановлює налаштування сповіщень.

SessionManager - клас, що відповідає за управління сесіями користувачів. Поля: activeSessions: Map<int, Object>: Карта активних сесій. Методи: startSession(userId: int): void: Запускає сесію для користувача. endSession(userId: int): void: Завершує сесію користувача. getSessionData(userId: int): Object: Отримує дані сесії користувача.

PostManager - клас, що відповідає за управління постами. Поля: posts: List<Post>: Список постів. Методи: publishPost(userId: int, postData: String, fileId: int): void: Публікує пост.getFeed(userId: int): List<Post>: Отримує стрічку новин користувача. likePost(userId: int, postId: int): void: Додає лайк до посту. commentPost(userId: int, postId: int, comment: String): void: Додає коментар до посту.

Post - клас, що представляє пост. Поля: postId: int: Ідентифікатор посту. postData: String: Дані посту. userId: int: Ідентифікатор користувача, який створив пост. fileId: int: Ідентифікатор файлу. timestamp: Date: Дата та час створення посту. likes: int: Кількість лайків. comments: List<Comment>: Список коментарів. Методи: editPost(postId: int, newData: String): void: Редагує пост. deletePost(postId: int): void: Видаляє пост.

Comment - клас, що представляє коментар. Поля: commentId: int: Ідентифікатор коментаря. postId: int: Ідентифікатор посту. userId: int: Ідентифікатор користувача, який залишив коментар. commentText: String: Текст коментаря. timestamp: Date: Дата та час створення коментаря.

MessageManager - клас, що відповідає за управління повідомленнями. Поля: messages: List<Message>: Список повідомлень. Методи: sendMessage(senderId: int, recipientId: int, message: String): void: Надсилає повідомлення. receiveMessage(userId: int): List<Message>: Отримує повідомлення користувача. createChat(userId1: int, userId2: int): int: Створює новий чат.

Message - клас, що представляє повідомлення. Поля: messageId: int: Ідентифікатор повідомлення. senderId: int: Ідентифікатор відправника. recipientId: int: Ідентифікатор отримувача. messageContent: String: Текст повідомлення. timestamp: Date: Дата та час створення повідомлення. Методи: markAsRead(): void: Позначає повідомлення як прочитане.

NotificationManager - клас, що відповідає за управління сповіщеннями. Поля: notifications: List<Notification>: Список сповіщень. Методи: sendNotification(userId: int, notificationContent: String): void: Надсилає сповіщення getNotifications(userId: int): List<Notification>: Отримує сповіщення користувача. setNotificationPreferences(userId: int, preferences: String): void: Встановлює налаштування сповіщень.

Notification-клас, що представляє сповіщення. Поля: notificationId: int: Ідентифікатор сповіщення. userId: int: Ідентифікатор користувача. notificationContent: String: Текст сповіщення. timestamp: Date: Дата та час створення сповіщення. Методи: markAsRead(): void: Позначає сповіщення як прочитане.

User - клас, що представляє користувача. Поля: userId: int: Ідентифікатор користувача. username: String: Ім'я користувача. passwordHash: String: Хеш пароля. profileData: String: Дані профілю. friendsList: List<int>: Список друзів. groups: List<int>: Список груп. Методи: register(username: String, password: String): bool: Реєструє користувача. login(username: String, password: String): bool: Авторизує користувача. editProfile(profileData: String): void: Редагує профіль. addFriend(friendId: int): void: Додає друга. joinGroup(groupId: int): void: Приєднується до групи. leaveGroup(groupId: int): void: Покидає групу.

Group - клас, що представляє групу. Поля: groupId: int: Ідентифікатор групи. groupName: String: Назва групи. members: List<int>: Список членів групи. Методи: addMember(userId: int): void: Додає члена до групи. removeMember(userId: int): void: Видаляє члена з групи. postMessage(userId: int, message: String): void: Публікує повідомлення в групі

2.4 Конструювання розроблюваного ПЗ

Конструювання включає написання коду, розробку алгоритмів, тестування та інтеграцію окремих модулів у єдину систему.

На цьому етапі розробники реалізують функціональність, визначену на стадії проектування, використовуючи програмні мови та інструменти.

Важливими аспектами цього процесу є забезпечення якості коду, відповідність стандартам програмування, а також проведення модульного та інтеграційного тестування для виявлення та усунення помилок.

Налаштування середовища розробки: встановлення Node.js та npm: вони потрібні для роботи з React. Створюється з новим проекто з Create React App:

```
npx create-react-app R-social-network
cd R-social-network
```

Спосіб конструювання системи соцмережі з використанням React включає кілька ключових етапів. Починається з налаштування середовища розробки, зокрема, встановлення Node.js та створення проекту за допомогою Create React App. Далі визначається структура проекту, розробляються та стилізуються основні компоненти (реєстрація, авторизація, профіль користувача, стрічка новин, коментарі). Потім інтегрується бекенд, побудований на Node.js з Express і MongoDB, з реалізацією RESTful API для взаємодії з фронтендом. Застосовується керування станом за допомогою Redux або Context API. Аутентифікація реалізується через JWT для безпечного доступу до ресурсів авторизованих користувачів. Після розробки проводиться тестування додатку, налаштовується CI/CD для автоматичного розгортання (наприклад, на Heroku), та забезпечується моніторинг продуктивності і логування для підтримки та масштабування системи.

2.2.1 Реалізація серверної частини системи

Налаштування бекенду передбачає вибір технології бекенду з використанням використовувати Node.js, Express, MongoDB, або інші бібліотеки технології. Далі відбувається створення API з RESTful для взаємодії з додатком. Аутентифікація

використовує JWT-токен для захисту маршрутів та зберігання інформації про користувачів.

Реалізація функціоналу:

Вхід та реєстрація: використовуються форми з валідацією.

Профілі користувачів: створено сторінки для відображення профілів та редагування інформації.

Пости: компонент для створення нових постів та відображення стрічки повідомлень.

Чати: реалізовано систему повідомлень, використовуючи технології оновлення реального часу.

Система складається з різних модулів: класи для конфігурації, контролери, класи для передачі даних, моделі (сутності), репозиторії, сервіси, допоміжні класи, класи для валідації даних та ресурси.

На рисунку 2.3 зображено створені модулі.

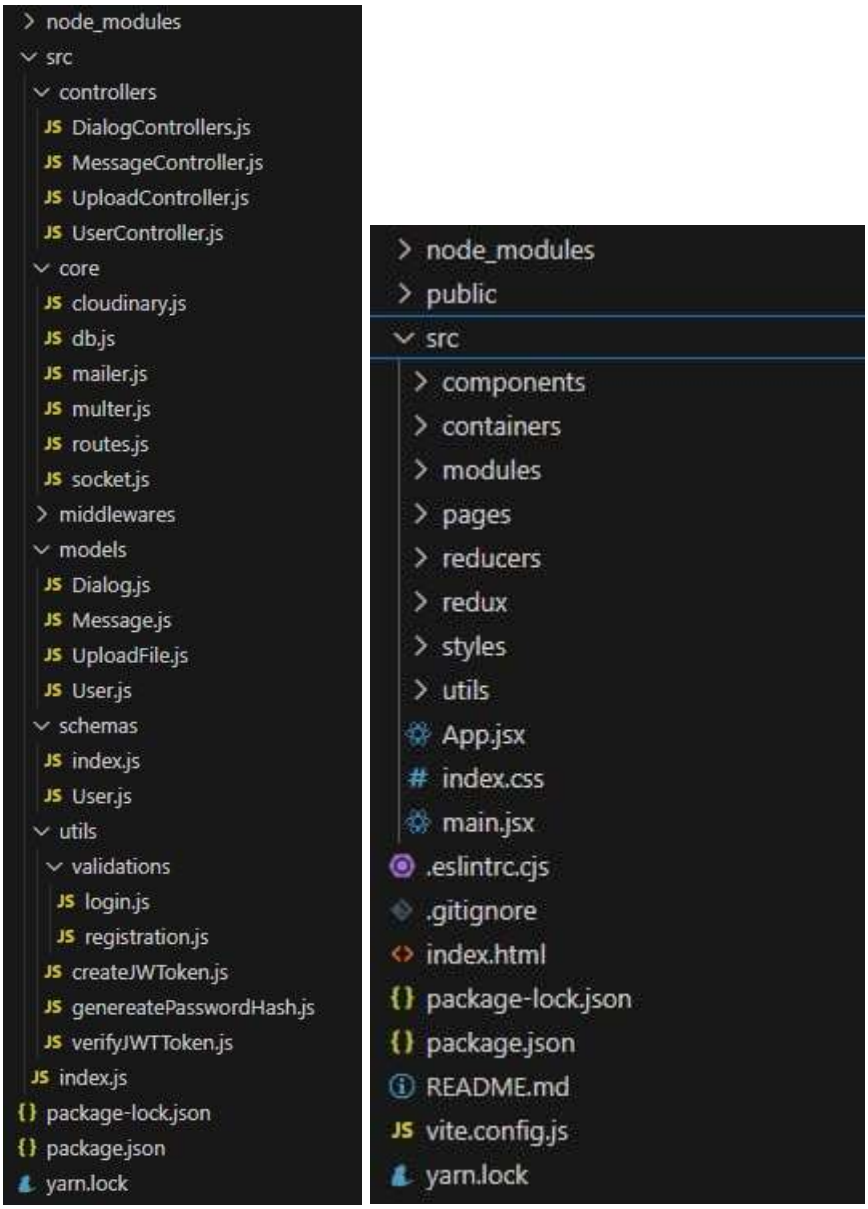


Рисунок 2.3 – Директорії backend та frontend-частин

Приклад коду для створення компонентів наведений нижче:

App.js

```jsx
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import HomePage from './pages/HomePage';
import LoginPage from './pages/LoginPage';
import ProfilePage from './pages/ProfilePage';
import Navbar from './components/Navbar';

function App() {
  return (
    <Router>
      <div>
        <Navbar />
        <Switch>
          <Route path="/" exact component={HomePage} />
          <Route path="/login" component={LoginPage} />
          <Route path="/profile/:id" component={ProfilePage} />
        </Switch>
      </div>
    </Router>
  );
}

export default App;
```

Рисунок 2.4 – Лістинг частини коду файлу App.jsx

```
import axios from 'axios';

const API_URL = 'http://localhost:5000/api/posts';

export const getPosts = async () => {
  const response = await axios.get(API_URL);
  return response.data;
};

export const createPost = async (post) => {
  const response = await axios.post(API_URL, post);
  return response.data;
};
```

Рисунок 2.5 – Лістинг частини коду файлу App.jsx

```jsx
import React, { useEffect, useState } from 'react';
import Post from '../components/Post';
import PostForm from '../components/PostForm';
import { getPosts } from '../services/postService';

function HomePage() {
  const [posts, setPosts] = useState([]);

  useEffect(() => {
    fetchPosts();
  }, []);

  const fetchPosts = async () => {
    const data = await getPosts();
    setPosts(data);
  };

  return (
    <div>
      <PostForm fetchPosts={fetchPosts} />
      {posts.map(post => (
        <Post key={post.id} post={post} />
      ))}
    </div>
```

Рисунок 2.6 – Лістинг частини коду файлу HomePage.jsx головної

```
import axios from 'axios';

const API_URL = 'http://localhost:5000/api/posts';

export const getPosts = async () => {
  const response = await axios.get(API_URL);
  return response.data;
};

export const createPost = async (post) => {
  const response = await axios.post(API_URL, post);
  return response.data;
};
```

Рисунок 2.7 – Лістинг частини коду файлу PostService.jsx для постів

2.2.2 Розробка Клієнтської частини системи

Розробка клієнтської частини системи соцмережі з використанням React включає кілька важливих кроків: створення проекту, налаштування проекту, встановлення залежностей необхідні бібліотеки, такі як react-router-dom для маршрутизації, axios для запитів до сервера, та інші залежності, що задіяні в обміні меседжами.

Структура проекту організована так, щоб компоненти, сторінки, та інші файли були зрозуміло розподілені:

```
/src
  /components
    - Navbar.js
    - Post.js
    - Profile.js
  /pages
    - HomePage.js
    - ProfilePage.js
    - LoginPage.js
  /services
    - api.js
  /styles
    - main.css
  - App.js
  - index.js
```

```
const Message = ({avatar, user, text, date, isMe, attachments}) => {
    return (
        <div className={classNames('message', {'message-isme' : isMe})}>
            <div className="message-avatar">
                <img src={avatar} alt={`${user.fullname}`} className='user-avatar' />
            </div>

            <div className="message-content">
                <div className="message-info">
                    <div className="message-author">{user.fullname}</div>
                    <div className="message-date">{formatDistanceToNow(date, {addSuffix: true, locale: uk})}</div>
                </div>
                <div className="message-bubble">
                    <p className="message-text">{text}</p>
                    <div className="message-attachments">
                    {attachments && attachments.map((item) => (
                    <div className="message-attachments-item">
                        <img src={item.url} alt={item.filename} />
                    </div>
                ))}
            </div>
                </div>
            </div>

        </div>
    );
}
```

Рисунок 2.3 – Лістинг коду React-component повідомлення

Це базовий приклад виконанн клієнтської частини соцмережі з використанням React-component для демонстрації повідомлення. В майбутньому його можна розширювати та вдосконалювати проект, додаючи нові функції та покращуючи інтерфейс.

2.5 Тестування додатку

Тестування здійснюється з допомогою магнуального методу проходження по usecase-переліку функціонлаьних можливостей системи а також на виявлення багів.

Для тестування компонентів використовуються компоненти бібліотеки для тестування к Jest і React Testing Library. Однак не всі способи реалізовано в ході виконання роботи.

Також варто звернути увагу на Інтеграційне тестування, щоб переконайтеся, що всі частини додатку працюють разом без проблем в різних середовищах (браузерах) на різних платформах. Це так звана кросплатформеність.

Вигляд додатку зображено на наступних рисунках.

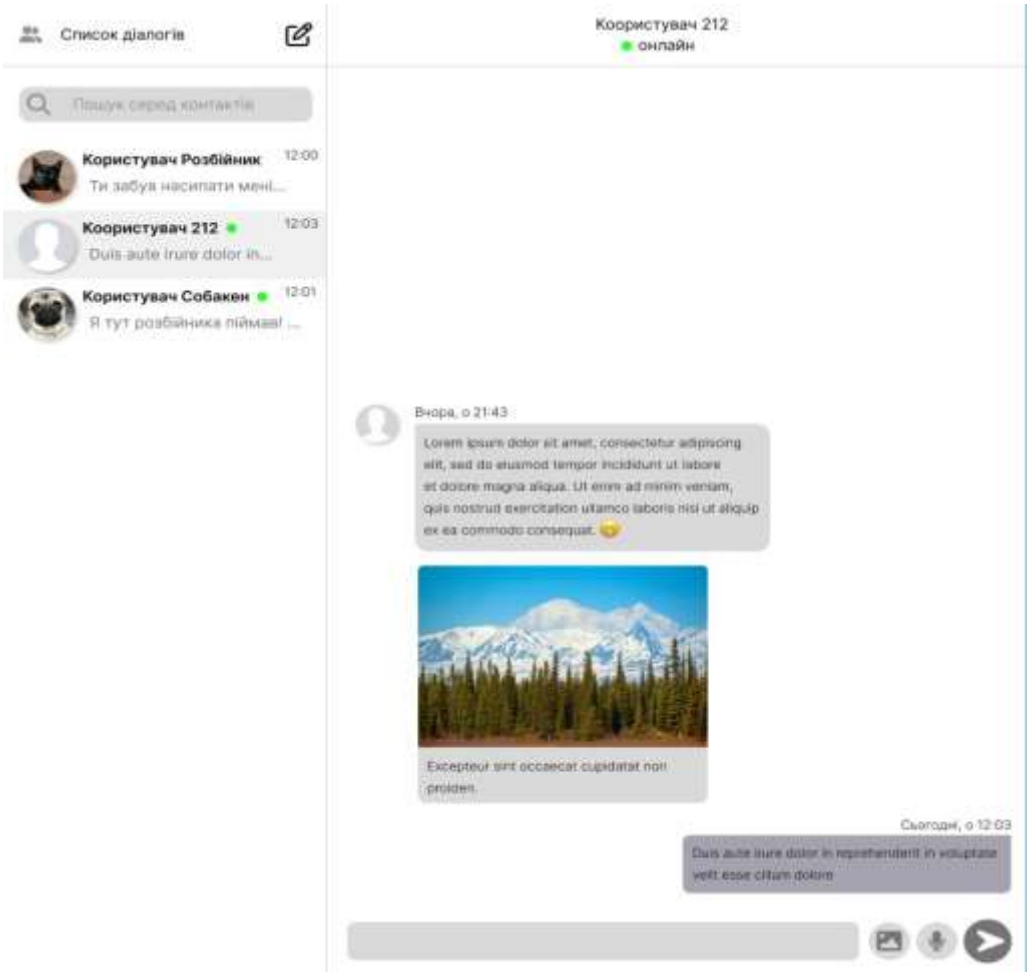


Рисунок 2.4 – Головна сторінка веб-додатку

Розроблене програмне забезпечення повністю задовольняє вимоги та специфікації проекту. Всі ключові функції виконані відповідно до поставлених задач, і додаток ефективно обробляє дані користувачів, квитків та круїзів. Система забезпечує надійну та безпечну роботу, що підтверджується результатами тестування.

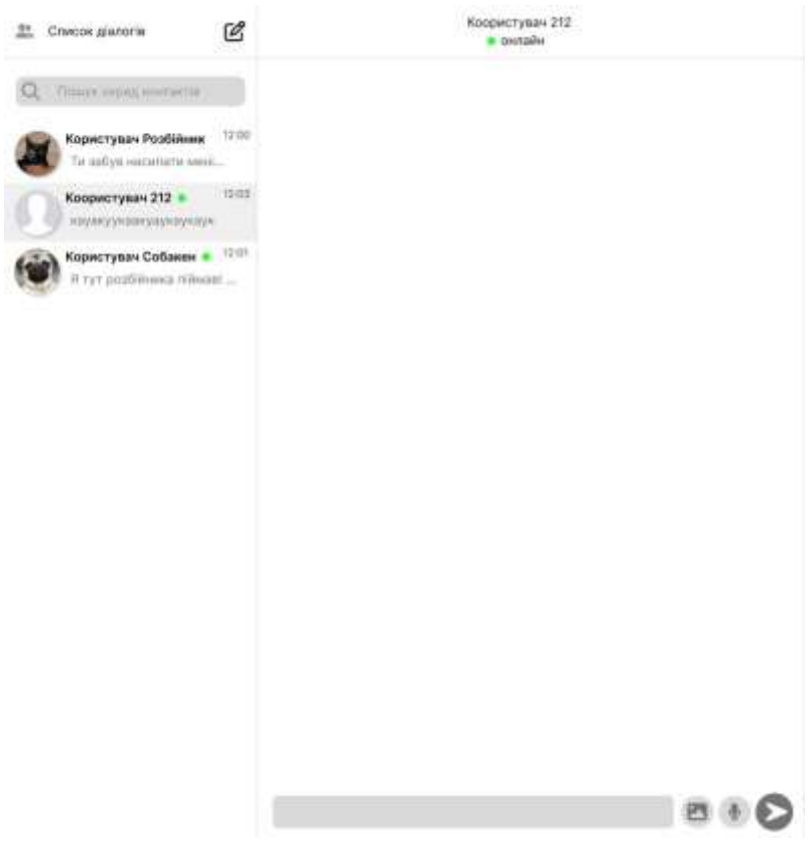


Рисунок 2.5 – Головна сторінка веб-додатку з повідомленням в чаті

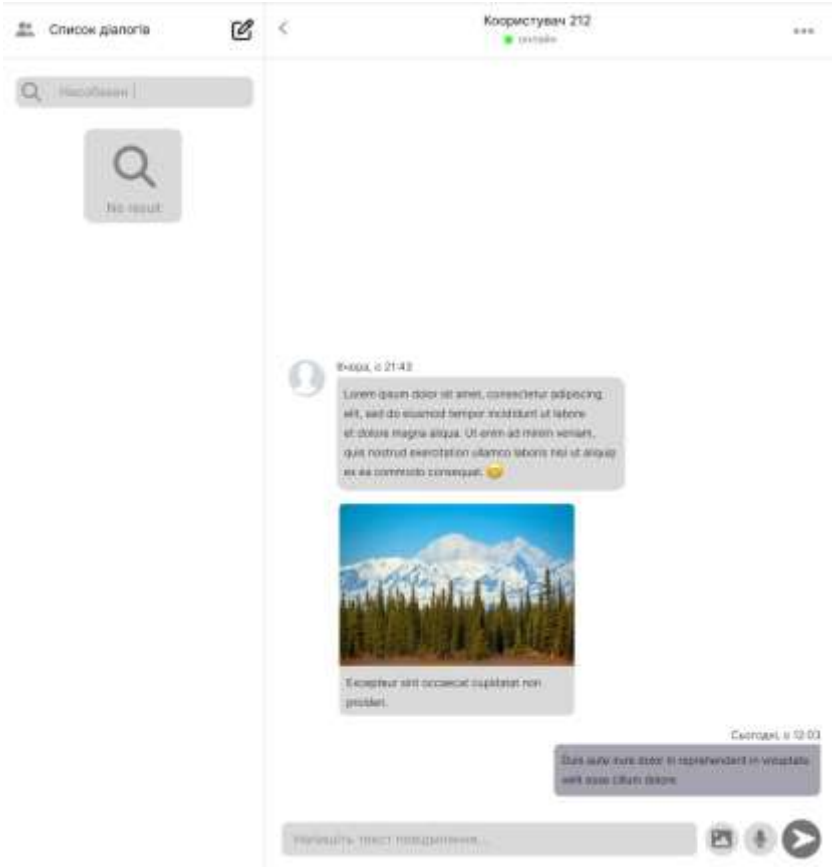


Рисунок 2.6 – Здійснення пошуку в чаті

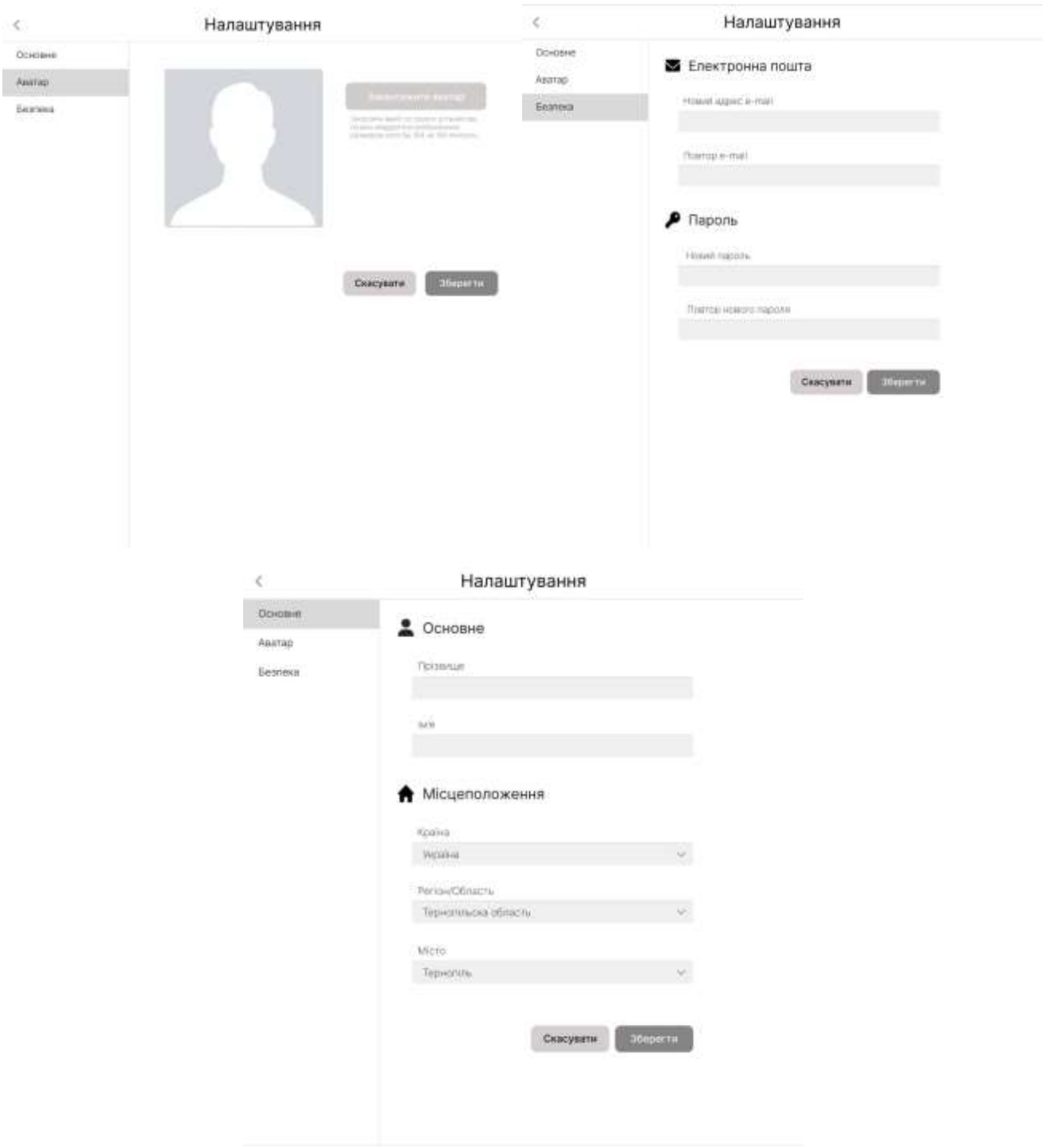


Рисунок 2.7 – Вікна налаштувань профілю користувача соцмережі

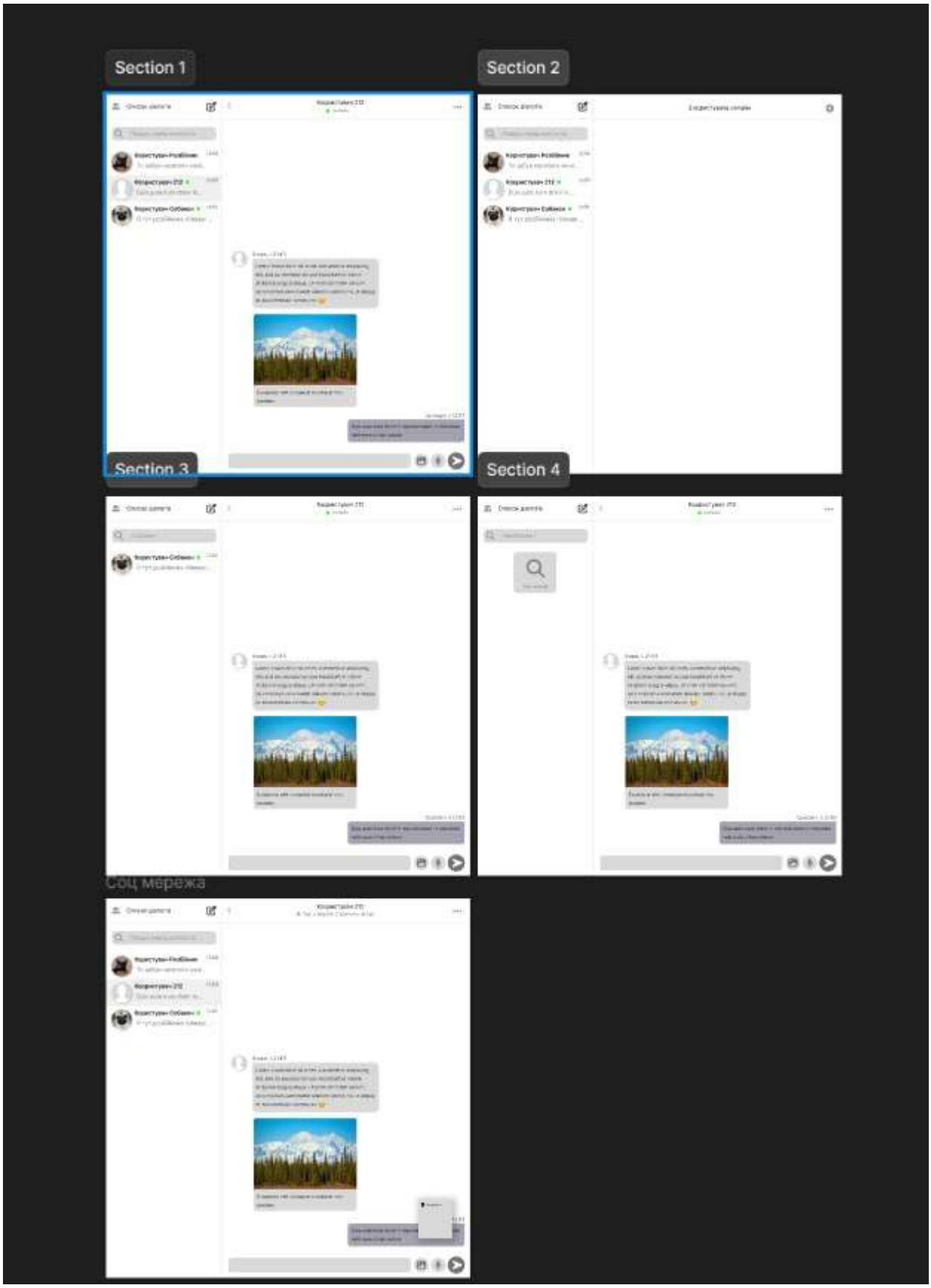


Рисунок 2.8 – UI-дизайн вікон системи меседжера для спілкування

Special Requirements: Немає спеціальних вимог.

Technology and Data Variations List: Інформація про завдання передається з файлу типу JSON.

Frequency of Occurrence: 80%, оскільки нові завдання створюються регулярно під час роботи над проєктом.

Розробка власної соціальної мережі надає численні переваги, включаючи повний контроль над функціональністю та дизайном, що дозволяє створювати унікальний користувацький досвід.[20] Вона забезпечує захист даних і конфіденційність, дозволяючи впроваджувати сучасні засоби автентифікації та передачі даних. Власна платформа також забезпечує гнучкість у розвитку та масштабуванні, дозволяючи швидко реагувати на змінні потреби ринку та додавати нові функції. Крім того, існує можливість інтегрувати корпоративний стиль під потреби користувачів, що підвищує лояльність користувачів і створює тісніший зв'язок з аудиторією [12].

3 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

3.1 Поведінкові реакції населення у надзвичайних ситуаціях

Кожна людина в побуті, на роботі, на природі може потрапити в такі ситуації, для яких притаманна несподіваність, тривалість та інтенсивність впливу несприятливих факторів навколишнього середовища, а іноді наявність загрози самому життю. Ситуації називають екстремальними, тому що вони виходять за межі звичайних.

Люди, які опинилися в зоні катастрофи або стихійного лиха, тобто в екстремальних умовах (в умовах НС), підлягають впливу різноманітних психотравматичних факторів, що обмовлені загрозою для життя [15]. Вони відчувають знач ні емоційні перевантаження, що можуть призводити до розвитку у них різноманітних захворювань. Людина опиняється в екстремальних ситуаціях з різних причин, але частіше за все це відбувається з власної вини як результат відсутності досвіду безпечної поведінки в природному та соціальному середовищі або зневага до норм, правил безпеки, непередбачливості або легковажності. Деякі люди не знають, як діяти в тій чи іншій життєвій ситуації. В результаті, опинившись у незвичному становищі, коли потрібні рішучі дії, люди виявляються зовсім безпомічними.

Зазначені небезпеки мають ту особливість, що характерізуються досить широким спектром чинників, які викликають ці небезпеки:

- вони виникають в результаті різного роду конфліктів, як соціально-політичного походження так і іншого (економічного, організаційного та ін..);
- їх породжують різного роду залежності – тютюнопаління, алкоголізм, наркоманія, ігроманія, інтернет-залежність тощо;
- до цих небезпек відносяться соціальні хвороби та епідемії – грип, гепатит, туберкульоз, СНІД та інші;
- вони включають екстремальні ситуації криміногенного характеру;

- в певній мірі серед них також потрібно розглядати небезпеки, викликані стрімкими темпами урбанізації, що веде до підвищення рівня злочинності, забруднення повітря, води, шумового, вібраційного, електромагнітного забруднення та інших небезпек.

Для того щоб зменшити ймовірність опинитися в екстремальній ситуації й збільшити шанси на збереження здоров'я і власного життя, необхідно знати і враховувати фактори ризику, які супроводжують наше життя.

По-перше, виробити уміння передбачати можливі виникнення небезпечних для життя ситуацій і зі знанням справи їх уникати.

По-друге, опинившись в екстремальній ситуації, швидко оцінити її і власні можливості, прийняти грамотне рішення та діяти.

В екстремальній ситуації у людини неминуче виникає особливе емоційне напруження або стрес. Це сукупність захисних фізіологічний реакцій, які виникають в організмі людини у відповідь на дію несприятливих зовнішніх факто- рів. Вплив стресу на поведінку й можливість людини надзвичайно індивідуальний. Є люди, які діють найефективніше саме в стані високого емоційного напруження в небезпечних для життя ситуаціях. А інших подібні ситуації деморалізують. Настає своєрідний психологічний шок з'являється сильна загальмованість, поспішність, метушливість, нездатність до розумних дій. При стресовому стані об'єм уваги звужений, переключення її загальмоване, м'язи напружені, рухи стають різкими, сточними, погано скоординованими, порушується пам'ять. Людина забуває послідовність дій, неправильно оцінює ситуацію, припускається грубих помилок [16]. Будь-яка емоція супроводжується активізацією нервової системи й появою в крові біологічно активних речовин, які змінюють діяльність внутрішній органів: кровообігу, дихання, травлення та ін. Однією з таких речовин с гормон надниркових залоз - адреналін.

Захисні й профілактичні дії імунної систем можуть бути придушені стресом, і організм втрачає здатність захистити себе від мікроорганізмів (віру- сів, бактерій). Звідси люди, які часто перебувають у стресовому стані, більшою мірою схильні до інфекційних захворювань, наприклад грипу.

Щоб витримати стрес, потрібно не боятися його, бути готовим до нього. Тренування стійкості може значно знизити нервове напруження, а отже, і нераціональну затрату енергетичних резервів організму. Для цього необхідно як на роботі, так і в повсякденному житті вчитися володіти собою. Слід контролювати свою поведінку, свою реакцію на все, що може спричиняти неадекватні емоції [16].

Багато людей вважають, що для зміцнення здоров'я потрібно всіляко уникати негативних емоцій. Але іноді тільки максимальна мобілізація психофізіологічних функцій допомагає людині вийти з критичної ситуації. Стрес впливає не тільки на психіку, але й на весь організм. Психосоматичні порушення спричиняють розвиток таких серйозних захворювань, як стенокардія, інфаркт міокарда, інсульт, гіпертонічна та виразкова хвороби, рак.

Короткочасні стреси мобілізують сили організму. Стрес невід'ємний елемент адаптаційних механізмів. А вони в свою чергу с фундаментальною властивістю живої матерії, запорукою еволюційного удосконалення будь-яких біо-систем.

Стресова дія збуджує активність організму і працездатність зростає, але чим довше, тим повільніше, досягаючи максимального рівню. Коли резерви вичерпані, а стрес наростає, працездатність знижується.

Тривалі стреси призводять до різного виду захворювань. Після того, як вичерпані всі резерви протистояння стресу, в результаті зриву пристосувальних механізмів людського організму виникають неврози. Наслідком довготривалих стресів с фобії (боязнь когось або чогось) [17].

Отже, перебування в екстремальних умовах передбачає підвищений фак тор ризику, необхідність прийняття адекватного рішення, високу відповідаль ність за виконання задачі, наявність неочікуваних перешкод.

3.2 Особливості безпеки праці під час вантажно-розвантажувальних робіт.

Для безпечного виконання вантажно-розвантажувальних робіт слід дотримуватися відповідних вимог правил, які затверджені українським законодавством в сфері безпеки та охорони праці.

Створення безпечних умов праці при вантажно-розвантажувальних роботах відбувається з урахування наступних вимог Відповідно правил згідно затверджених у статті 28 Закону України «Про охорону праці» [19] правил охорони праці під час вантажно-розвантажувальних робіт :

• роботодавець повинен забезпечити відповідний санітарно-гігієнічний стан виробничих приміщень;

• створити службу охорони праці (відповідно до вимог законодавства);

• організувати опрацювання і затвердити нормативні акти про охорону праці, що діють на підприємстві;

• розробити та затвердити інструкції з охорони праці; забезпечити проведення попереднього та періодичних медичних оглядів; розробити і затвердити перелік робіт з підвищеною небезпекою;

• організовувати проведення атестації робочих місць за умовами праці;

• одержати дозвіл на виконання робіт підвищеної небезпеки та на експлуатацію (застосування) машин, механізмів, устаткування підвищеної небезпеки;

• забезпечити працівників спецодягом, спеціальним взуттям та іншими ЗІЗ.

Під час організації та ведення технологічних процесів, пов’язаних із застосуванням шкідливих речовин, необхідно дотримуватися правил, що затверджені нормативними актами з цього питання.

• необхідно створити відповідний мікроклімат у виробничих приміщеннях;

• природне та штучне освітлення на робочих місцях повинно відповідати вимогам законодавства; до експлуатації допускаються справне устаткування, механізми або пристрої, що відповідають вимогам безпеки;

• роботи по очищенню цистерн повинні виконувати не менше ніж три працівники, один із яких перебуває в цистерні, а двоє працівників, що спостерігають за виконанням роботи (спостерігачі), повинні перебувати поза цистерною. Між спостерігачами і працівником у цистерні повинен бути встановлений переговірний зв'язок або визначені сигнали, які передаються за допомогою страхувального каната (мотузки), що забезпечує підйом працівника нагору на його вимогу;

• навантаження (розвантаження) балонів з горючими газами, а також легкозаймистих рідин, речовин і матеріалів, здатних накопичувати на своїй поверхні заряди статичної електрики, та в спецвзутті, підбитому металевими (крім латунних) цвяхами або підковами.

Для безпечного виконання вантажно-розвантажувальних робіт слід дотримуватися відповідних вимог правил, які затверджені українським законодавством в сфері безпеки та охорони праці.

Правила охорони праці під час вантажно-розвантажувальних робіт, затверджені наказом Міністерства енергетики та вугільної промисловості України [17] 19 січня 2015 року № 21, поширюються на всіх суб'єктів господарювання (роботодавців та працівників) незалежно від форм власності та організаційно-правової форми, які в процесі своєї діяльності виконують: вантажно-розвантажувальні роботи, навантаження (розвантаження) та вивантаження.

Варто зауважити, що Правила [19] містять вичерпний перелік вимог щодо безпеки та охорони праці при виконанні відповідних робіт:

• до місць проведення вантажно-розвантажувальних робіт, території підприємства, виробничих будівель і споруд;

• до застосування і розміщення підіймально-транспортного устаткування, механізмів, пристосувань та огороджень, організації робочих місць;

• безпеки до транспортування вантажів, до внутрішньозаводського й цехового транспорту; до транспортування вантажів колісними та гусеничними тракторами;

• до транспортування вантажів авто- і електронавантажувачами;

• до транспортування вантажів підлоговими рейковими передатними електровізками; транспортування матеріалів, напівфабрикатів і готової продукції на підприємствах, в організаціях поліграфічної промисловості і книжкової торгівлі.

Загальні вимоги до вантажопідйомних механізмів (кранів), до вантажно-розвантажувального майданчику, рухомого складу та обслуговуючого його персоналу повинні виконуватися згідно з діючими правилами та інструкціями.

ВИСНОВКИ

У процесі виконання дипломної роботи на тему "Розробка мережі соціальної взаємодії з використанням технологій React та Node.js" було досягнуто поставлених цілей. Розроблено MVP веб-додатоку, що демонструє можливості використання його як соцмережі для певних категорій населення чи соціальних груп.

Загалом, веб-додаток має вдалий інтерфейс з точки зору зручності та простоти використання. Проте кольорова гама потребує доопрацювання, щоб покращити візуальне сприйняття та зробити користування додатком більш комфортним. До рекомендацій варто віднести: редизайн кольорової гами веб-додатку, щоб вона стала більш гармонійною та приємною для очей. Варто провести тестування оновленого дизайну з користувачами, щоб отримати їхню оцінку та внести необхідні правки.

Роз’яснено основні аспекти задуманого програмного продукту, наведено аналоги, їхні переваги та недоліки, в загальних відомостях розписано основні функції системи, її користувачі, схему, сценарій роботи, засоби апаратної та програмної реалізації, також дано звіт про популярні додатки для спілкування в країнах світу, інтерфейс одного із них для прикладу і таблицю з ключовими прецедентами системи.

Розробку за супровід здійснено відповідно до повного циклу виконання проекту з використанням та сучасного підходу Agile. Здійснено аналі ринку цих послуг, огляд конкурентів, їх переваг та недоліків. Обрано та спроектовано архітектуру системи, вибрано технології для реалізації веб-додатку. Здійснено конструювання соцмережі та тестування результатів розробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Abramov, D., & Clark, A. (n.d.). Redux: A predictable state container for JavaScript apps. Режим доступу: https://redux.js.org/
2. MongoDB, Inc. (n.d.). MongoDB: The most popular database for modern apps. Retrieved from. Режим доступу: https://www.mongodb.com/
3. Relation Data Model [Електронний ресурс] – Режим доступу ресурсу: https://www.tutorialspoint.com/dbms/relational_data_model.htm.
4. Посібник: знайомство з React [Електронний ресурс] – Режим доступу до ресурсу: https://uk.reactjs.org/tutorial/tutorial.html.
5. Communication: Online vs. Face-to-Face Interactions [Електронний ресурс]. – Режим доступу : https://psychminds.com/communication-online-vs-face-to-faceinteractions/
6. Magolan G., Bell J., Guijarro D., Peretti A., Housley P. Nest.js: A Progressive Node.js Framework, Kindle Edition. Bleeding Edge Press, 2018. 350 p.
7. О. Голинська, І. Мудрик. Роль CRM-системи у сучасних бізнес-процесах. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології “Тернопільського національного технічного університету імені Івана Пулюя, 2022, 25-25, ТНТУ.
8. Функціональне тестування з Sentry. Functional Software, Inc. (Sentry). (n.d.). Sentry: Application Monitoring and Error Tracking Software. Retrieved from Режим доступу: https://sentry.io/
9. Bradshaw S., Brazil E., Chodorow K. MongoDB: The Definitive Guide, 3rd Edition. O'Reilly Media, 2019. 514 p.
10. Методичні вказівки до виконання атестаційної роботи магістра спеціальності 121 – Інженерія програмного забезпечення (Освітньо-професійна програма - «Програмне забезпечення систем», Освітньо-наукова програма -«Інженерія програмного забезпечення») для студентів усіх форм навчання / Упор.: М.Р.

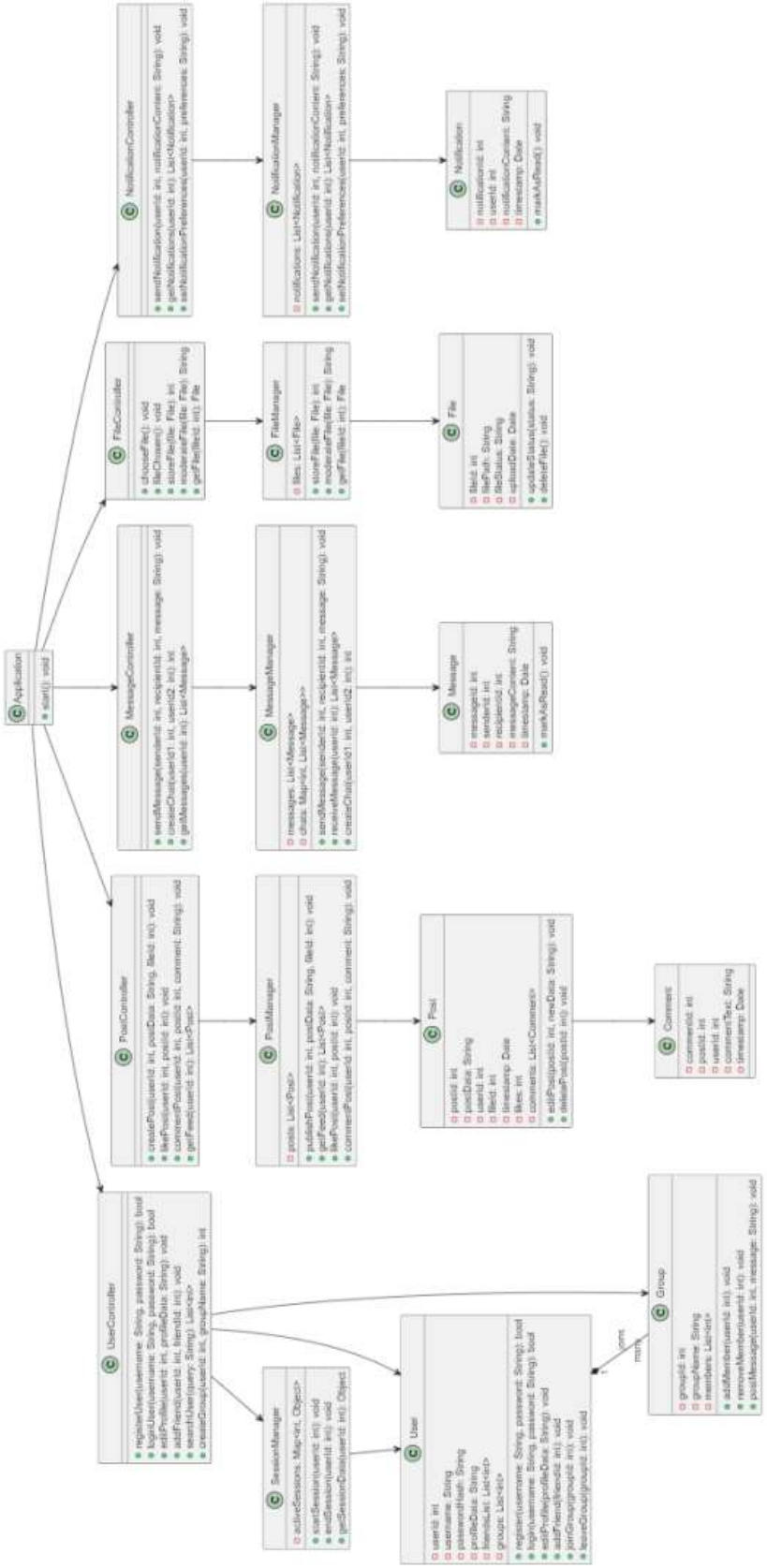
Петрик, Д.М. Михалик, О.Ю. Петрик, Г.Б. Цуприк - Тернопіль: ТНТУ, 2020-51с

11. Навчальний посібник" Сучасні технології ООП-проектування та автоматичного генерування програмного коду". Упор.: М.Р. Петрик, І.Я. Мудрик, О.Ю. Петрик, Ю.М. Стоянов - Тернопіль: ТНТУ, 2018-49с
12. Nadiia, Y., Nadiia, M., Lesia, O., & Ivan, Y. (2023). RADIAL-BASIS NEURAL NETWORKS FOR ENTERPRISES ACTIVITY PREDICTION. European Science, 3(sge17-03), 42–48. https://doi.org/10.30890/2709-2313.2023-17-03-012
13. Гандзюк М.П., Желібо Є.П., Халімовський М.О. Основи охорони праці: Підручник. 5-е вид. / За ред. М.П. Гандзюка. – К.: Каравела, 2011. – 384 с.
14. О Мельник, М Петрик. Розробка архітектури інформаційної системи ділових комунікацій на основі оптимізацій стрімінгової платформи. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології “Тернопільського національного технічного університету імені Івана Пулюя 121-121. ТНТУ
15. Тарасюк В.С., Матвійчук М.В., Паламар І.В., Корольова Н.Д., Кучанська Г.Б., Новицький Н.О. Медицина надзвичайних ситуацій. Організація надання першої медичної допомоги: Підручник. - К.: «Медицина», 2011. – 526 с.
16. Гребінь А. А. Соціальні мережі – сучасне середовище спілкування з користувачами. [Електронний ресурс]. URL: http://conference.nbuv.gov.ua/report/view/id/615 (дата звернення: 13.04.2022).
17. Сєріков Я. О. Безпека життєдіяльності та охорона праці / Я. О. Сєріков, Л. Ф. Коженевські, М. В. Хворост. – Харків, 2021. – 255 с.
18. Гаврилець І. Г. Психофізіологія людини в екстремальних ситуаціях: навч. посібн. Київ: Випол, 2006. 188 с.
19. Наказ Міністерства енергетики та вугільної промисловості України «Про затвердження Правил охорони праці під час вантажно-розвантажувальних робіт» № 21 від 19.01.2015 . Режим доступу: https://zakon.rada.gov.ua/laws/show/z0124-15#n15

ДОДАТКИ

ДОДАТОК А

Розширені діаграма класів сутностей системи.



ДОДАТОК Б

Лістинги коду.

Файл socket.js----------

```
import socket from 'socket.io';
import http from 'http';
export default (http: http.Server) => {
  const io = socket(http);
  io.on('connection', function(socket: any) {
    socket.on('DIALOGS:JOIN', (dialogId: string) => {
      socket.dialogId = dialogId;
      socket.join(dialogId);
    });
    socket.on('DIALOGS:TYPING', (obj: any) => {
      socket.broadcast.emit('DIALOGS:TYPING', obj);
    });
  });

  return io;
};
```

Файл routes.js----------

```
import bodyParser from "body-parser";
import express from "express";
import socket from "socket.io";
import { updateLastSeen, checkAuth } from "../middlewares";
import { loginValidation, registerValidation } from "../utils/validations";

import multer from "./multer";
```

```
import {
  UserCtrl,
  DialogCtrl,
  MessageCtrl,
  UploadFileCtrl,
} from "../controllers";

const createRoutes = (app: express.Express, io: socket.Server) => {
  const UserController = new UserCtrl(io);
  const DialogController = new DialogCtrl(io);
  const MessageController = new MessageCtrl(io);
  const UploadFileController = new UploadFileCtrl();

  app.use(bodyParser.json());
  app.use(checkAuth);
  app.use(updateLastSeen);

  app.get("/", (_: express.Request, res: express.Response) => {
    res.send("Hello, World!");
  });

  app.get("/user/me", UserController.getMe);
  app.get("/user/verify", UserController.verify);
  app.post("/user/signup", registerValidation, UserController.create);
  app.post("/user/signin", loginValidation, UserController.login);
  app.get("/user/find", UserController.findUsers);
  app.get("/user/:id", UserController.show);
  app.delete("/user/:id", UserController.delete);
```

```
  app.get("/dialogs", DialogController.index);
  app.delete("/dialogs/:id", DialogController.delete);
  app.post("/dialogs", DialogController.create);

  app.get("/messages", MessageController.index);
  app.post("/messages", MessageController.create);
  app.delete("/messages", MessageController.delete);

  app.post("/files", multer.single("file"), UploadFileController.create);
  app.delete("/files", UploadFileController.delete);
```

Файл Status.js----------

```
import React from "react";
import { Status as StatusBase } from "components";
import { connect } from "react-redux";

const Status = ({ currentDialogId, user, dialogs }) => {
  if (!dialogs.length || !currentDialogId) {
    return null;
  }

  const currentDialogObj = dialogs.filter(
    dialog => dialog._id === currentDialogId
  )[0];

  let partner = {};

  if (currentDialogObj.author._id === user._id) {
```

```
    partner = currentDialogObj.partner;
  } else {
    partner = currentDialogObj.author;
  }

  return <StatusBase online={partner.isOnline} fullname={partner.fullname} />;
};

export default connect(({ dialogs, user }) => ({
  dialogs: dialogs.items,
  currentDialogId: dialogs.currentDialogId,
  user: user.data
}))(Status);
```

Файл messages.js----------

```
import { messagesApi } from "utils/api";

const Actions = {
  setMessages: items => ({
    type: "MESSAGES:SET_ITEMS",
    payload: items
  }),
  addMessage: message => (dispatch, getState) => {
    const { dialogs } = getState();
    const { currentDialogId } = dialogs;
```

```
  if (currentDialogId === message.dialog._id) {
    dispatch({
      type: "MESSAGES:ADD_MESSAGE",
      payload: message
    });
  }
},
fetchSendMessage: ({ text, dialogId, attachments }) => dispatch => {
  return messagesApi.send(text, dialogId, attachments);
},
setIsLoading: bool => ({
  type: "MESSAGES:SET_IS_LOADING",
  payload: bool
}),
removeMessageById: id => dispatch => {
  if (window.confirm("Ви точно хочете видалити повідомлення?")) {
    messagesApi
      .removeById(id)
      .then(({ data }) => {
        dispatch({
          type: "MESSAGES:REMOVE_MESSAGE",
          payload: id
        });
      })
      .catch(() => {
        dispatch(Actions.setIsLoading(false));
      });
  }
},
fetchMessages: dialogId => dispatch => {
```

```
    dispatch(Actions.setIsLoading(true));
    messagesApi
      .getAllByDialogId(dialogId)
      .then(({ data }) => {
        dispatch(Actions.setMessages(data));
      })
      .catch(() => {
        dispatch(Actions.setIsLoading(false));
      });
  }
};

export default Actions;
```

Файл user.js----------

```
import { openNotification } from 'utils/helpers';
import { userApi } from 'utils/api';

const Actions = {
  setUserData: data => ({
    type: 'USER:SET_DATA',
    payload: data,
  }),
  setIsAuth: bool => ({
    type: 'USER:SET_IS_AUTH',
    payload: bool,
  }),
  fetchUserData: () => dispatch => {
    userApi
```

```
    .getMe()
    .then(({ data }) => {
      dispatch(Actions.setUserData(data));
    })
    .catch(err => {
      if (err.response.status === 403) {
        dispatch(Actions.setIsAuth(false));
        delete window.localStorage.token;
      }
    });
},
fetchUserLogin: postData => dispatch => {
  return userApi
    .signIn(postData)
    .then(({ data }) => {
      const { token } = data;
      openNotification({
        title: 'Супер!',
        text: 'Авторизация пройшла успішно!',
        type: 'success',
      });
      window.axios.defaults.headers.common['token'] = token;
      window.localStorage['token'] = token;
      dispatch(Actions.fetchUserData());
      dispatch(Actions.setIsAuth(true));
      return data;
    })
    .catch(({ response }) => {
      openNotification({
        title: 'Помилка авторизації',
```

```
          text: 'Невірний логін чи пароль',
          type: 'error',
        });
      });
  },
  fetchUserRegister: postData => () => {
    return userApi.signUp(postData);
  },
};

export default Actions;

};

export default createRoutes;
```

ДОДАТОК В

Диск з матеріалами.