МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри _____ Є.О. Давиденко

підпис

«___»_____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Вебзастосунок онлайн-комунікації користувачів

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.01 – 408.21910810

Студент _____ К. В. Ібрагімов

підпис

«__»_____ 2023 р.

Керівник PhD**,** ст. викладач кафедри ІПЗ _____ І. О. Кандиба

підпис

«__»_____ 2023 р.

Консультант канд. техн. наук, доцент

«__» _____А. О. Алексєєва

підпис

«__»_____ 2023 р.

Миколаїв – 2023

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

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри Давиденко Є. О.

«_____» _____ 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 408 факультету комп'ютерних наук

Ібрагімову Кирилу В'ячеславовичу
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

«Вебзастосунок онлайн комунікації користувачів»

Затверджена наказом по ЧНУ від «17» березня 2023 р. № 60

2. Строк представлення кваліфікаційної роботи «____» _____ 20___ р.

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

Розроблений та протестований вебзастосунок онлайн-комунікації користувачів

4. Перелік питань, що підлягають розробці

- проаналізувати існуючі соціальні мережі;
- дізнатись особливості проєктування та створення вебзастосунків;
- спроєктувати авторизацію, системи обміну повідомленнями, створення постів;
- розробити спроєктований вебзастосунок, провести тести застосунку.

5. Перелік графічних матеріалів:

Презентація.

6. Завдання до спеціальної частини

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
А. О. Алексєєва	Кафедра екології	Охорона праці

Керівник роботи PhD ст. викладач Кандиба Ігор Олександрович

(посада, прізвище, ім'я, по батькові)

Завдання прийнято до виконання

Ібрагімов Кирило В'ячеславович

(прізвище, ім'я, по батькові студент)

Дата видачі завдання «_____»_____2023 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: Вебзастосунок онлайн-комунікації користувачів

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	30.11.2022	01.12.2022	Виконано
2.	Визначення функціональних можливостей соціальної мережі	02.12.2022	21.12.2022	Виконано
3.	Розробка дизайну та функціональності	06.01.2023	10.01.2023	Виконано
4.	Підключення та тестування Google авторизації	11.01.2023	22.01.2023	Виконано
5.	Розробка програмного забезпечення	23.01.2023	08.02.2023	Виконано
6.	Створення бази даних для зберігання інформації на базі MongoDB	9.02.2023	22.02.2023	Виконано
7.	Тестування, апробація та усунення помилок	23.02.2023	01.03.2023	Виконано
8.	Розробка спеціальної частини з охорони праці	2.03.2023	28.03.2023	Виконано
9.	Оформлення КРБ та презентації	1.04.2023	20.04.2023	Виконано
10.	Відгук керівника КРБ	21.04.2023	28.04.2023	Виконано
11.	Попередній захист	8.05.2023	9.05.2023	Виконано
12.	Завершення оформлення КРБ та презентації	10.05.2023	4.06.2023	Виконано
13.	Захист кваліфікаційної роботи	26.06.2023	26.06.2023	Виконано

Розробив студент _____Ібрагімов Кирило В’ячеславович _____

(підпис) «__» _____ 2023 р.

Керівник роботи _PhD, ст. викладач Кандиба Ігор Олександрович_____ _____

(підпис) «__» _____ 2023 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Вебзастосунок онлайн-комунікації користувачів»

Студент 408 гр.: Ібрагімов Кирило В'ячеславович

Керівник: PhD, ст. викладач Кандиба Ігор Олександрович

Під час війни та з сучасним рівнем розвитку технологій люди стали менше розмовляти наживо. Спостерігається тенденція зменшення чи навіть неможливості спілкування в живу, що спричиняє необхідність використання соціальних мереж та месенджерів для спілкування. Отже галузь онлайн спілкування є дуже актуальною темою.

Об'єктом кваліфікаційної роботи є процес онлайн комунікації між людьми.

Предметом кваліфікаційної роботи є технології та алгоритми розробки вебзастосунку онлайн комунікації між людьми.

Метою кваліфікаційної роботи є полегшення комунікації між людьми зі всього світу шляхом розробки сучасного вебзастосунку.

Для досягнення мети потрібно виконати такі завдання, як:

- проаналізувати існуючі соціальні мережі;
- дізнатись особливості проєктування та створення вебзастосунків;
- спроєктувати авторизацію, системи обміну повідомленнями, створення постів;
- розробити спроєктований вебзастосунок;
- провести тести застосунку.

У першому розділі КРБ проведено аналіз сучасних автоматизованих систем та ресурсів, орієнтованих на онлайн комунікацію користувачів, оцінено сильні та слабкі сторони кожної системи, проведено порівняльний аналіз.

У другому розділі проводиться проєктування, конструювання та моделювання програмного забезпечення для вебзастосунку онлайн

комунікації користувачів. Це було виконано, включаючи створення архітектурних діаграм та діаграм варіантів використання та інших відповідних схем.

Третій розділ показує основні кроки проєктування застосунку, проєктування було розроблено діаграму класів для визначення структури системи і створено частину проєкту на основі визначеної структури класів. Також було розроблено діаграму станів, що дає змогу краще зрозуміти функції застосунку, було створено діаграму діяльності яка відображає деяку взаємодію та діяльності між компонентами системи.

У четвертому розділі розглянуто вибір технологій для розробки вебзастосунків. Обрано React Redux для розробки програмного забезпечення та MongoDB як базу даних. Використано WebStorm для написання коду і MongoDB Compass для створення бази даних. Архітектура MVC була використана для розділення застосунку на модель, вид і контролер. Описано користувацький інтерфейс з основними сторінками та функціями.

Результатом КРБ є функціональна соціальна мережа, розроблена в процесі комплексного аналізу, розробки та проєктування.

КРБ викладена на 95 сторінки, вона містить 4 розділи, 22 ілюстрацій, 10 таблиць, 17 джерел в переліку посилань.

Ключові слова: *сфера соціальних мереж, вебзастосунок, React, авторизація, обмін повідомленнями, профіль користувача*

ABSTRACT

of the Bachelor`s Thesis

«Web application for online user communication»

Student of group 408: Ibrahimov Kyrylo Viacheslavovych

Supervisor: Phd, senior Lecturer Igor Oleksandrovych Kandyba

During the war and with the current level of technology development, people are talking less in person. There is a tendency to reduce or even make it impossible to communicate in person, which necessitates the use of social networks and messengers for communication. Therefore, the field of online communication is a very relevant topic.

The object of the qualification work is the process of online communication between people.

The subject of the qualification work is technologies and algorithms for developing a web application for online communication between people.

The aim of the qualification work is to facilitate communication between people from all over the world by developing a modern web application.

To achieve this goal, you need to complete the following tasks

- analyze existing social networks;
- learn the peculiarities of designing and creating web applications;
- design authorization, messaging systems, and post creation;
- develop the designed web application;
- conduct application tests.

In the first chapter of the thesis, an analysis of modern automated systems and resources focused on online communication between users is carried out, the strengths and weaknesses of each system are assessed, and a comparative analysis is carried out.

The second section, the design, construction and modeling of the software for the online user communication web application is carried out. This was

accomplished by creating architectural and use case diagrams and other relevant diagrams.

The third section shows the main steps of designing the application, designing a class diagram to define the structure of the system and creating a part of the project based on the defined class structure. Also, a state diagram was developed to better understand the functions of the application, an activity diagram was created to show some of the interactions and activities between the components of the system.

The fourth section discusses the choice of technologies for developing web applications. React Redux was chosen for software development and MongoDB as a database. We used WebStorm to write the code and MongoDB Compass to create the database. The MVC architecture was used to divide the application into a model, view, and controller. The user interface with the main pages and functions is described.

The CRS is set out on 95 pages long, it contains 4 sections, 22 illustrations, 10 tables, 17 sources in the list of links.

Keywords: social networking, web application, React, authorization, messaging, user profile.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – База даних

ПЗ – Програмне забезпечення

CSS – Cascading Style Sheets

HTML – Hyper Text Markup Language

JS – JavaScript

SCSS – Sassy Cascading Style Sheets (Syntactically Awesome Style Sheet)

TS – TypeScript

UML - Unified Modeling Language

UI – User interface

ВСТУП

Під час війни та з сучасним рівнем розвитку технологій люди стали менше розмовляти наживо. Спостерігається тенденція зменшення чи навіть неможливості спілкування в живу, що спричиняє необхідність використання соціальних мереж та месенджерів для спілкування. Отже галузь онлайн спілкування є дуже актуальною темою.

Також актуальність теми зумовлена тим, що вміння проєктувати соціальні мережі є популярним на ринку праці у зв'язку з більшим переходом на цифрові технології. Оскільки всі соціальні мережі мають аналогічний функціонал, то на базі створених в ході виконання роботи матеріалів можна проєктувати соціальні мережі іншої тематики та рівня складності.

Темою даної КРБ є вебзастосунок онлайн комунікації користувачів.

Об'єктом кваліфікаційної роботи є процес онлайн комунікації між людьми.

Предметом кваліфікаційної роботи є технології та алгоритми розробки вебз астосунку онлайн комунікації між людьми.

Метою кваліфікаційної роботи є полегшення комунікації між людьми зі всього світу шляхом розробки сучасного вебзастосунку. Для досягнення мети потрібно виконати такі завдання, як:

- проаналізувати існуючі соціальні мережі;
- дізнатись особливості проєктування та створення вебзастосунків;
- спроєктувати авторизацію, системи обміну повідомленнями, створення постів;
- розробити спроєктований вебзастосунок;
- провести тести застосунку.

1 АНАЛІЗ ПРИКЛАДНОЇ ОБЛАСТІ ТА ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Опис предметної галузі

Соціальна мережа – це вебплатформа, яка дозволяє користувачам обмінюватися інформацією, контактувати з іншими користувачами та будувати віртуальні спільноти. Соціальна мережа зазвичай забезпечує користувачів інструментами для створення профілю, додавання до друзів, публікації постів, повідомлень, фото матеріалів.

Аналізуючи предметну галузь соціальних мереж, можна зазначити, що це вона швидко розвивається та змінюється відповідно до потреб користувачів. Бізнес-аккаунти в соціальних мережах можуть бути використані для реклами та просування продуктів та послуг, а також для взаємодії зі споживачами та отримання зворотного зв’язку. Крім того, соціальні мережі можуть бути використані як засіб розвитку освіти та саморозвитку, зокрема шляхом обміну знаннями та досвідом між користувачами. Однією з ключових переваг соціальних мереж є можливість взаємодії з іншими користувачами з усього світу, що дає змогу розширити свої знання та досвід, знаходити нових друзів.

Однак, соціальні мережі також можуть мати деякі негативні наслідки, зокрема можуть виникати проблеми з приватністю та кібербулінгом. Крім того, соціальні мережі можуть бути джерелом поширення недостовірної інформації, що може мати шкідливий вплив на суспільство. Також, використання соціальних мереж може призвести до залежності від інтернету та відсутності живого спілкування, що може негативно позначитися на психічному здоров'ї людини.

1.2 Аналіз найбільш актуальних засобів онлайн-комунікації

Для аналізу сучасних засобів онлайн-комунікації розглянуто ресурси трьох лідерів ринку «Viber», «Twitter» та «Telegram».

VoIP застосунок для дзвінків і обміну повідомленнями Viber.

Назва: Viber (рис. 1.1)

Архітектура: Трьохрівнева архітектура: Вебзастосунок, постійне сховище, розподілений індекс.

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

Мета: Спілкування в інтернеті, залишання коментарів, надсилання повідомлень.

Недоліки:

1. потребує багато пам’яті телефону;
2. використовує багато інтернет трафіку;
3. обов’язкова наявність sim – карти.

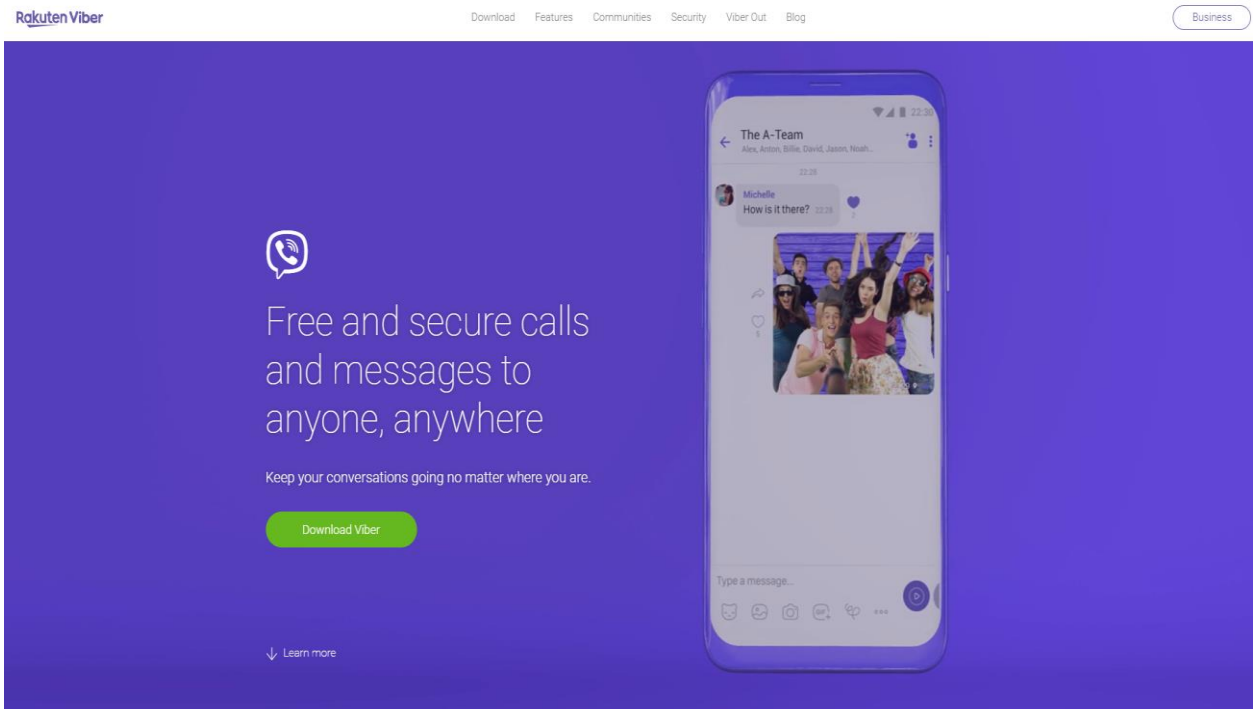

Рисунок 1.1 – Головна сторінка соціальної мережі Facebook

Соціальна мережа Twitter

Назва: Twitter (рис. 1.2)

Архітектура: Twitter Timeline

Мова реалізації: JavaScript, Ruby, Scala, Java.

Мета: Сайт для соціальних медіа для новин, розваг, спорту, політики, та багато іншого, головна мета наголошення інформації в реальному часі.

Переваги:

1) безкоштовний;
2) масивний;
3) безпечний;
4) швидкий;
5) не має реклами;
6) відносини зі знаменитостями та іншими громадськими діячами

Недоліки:

- дуже обмежена кількість символів у повідомленнях чи публікаціях;
- жодних засобів проти розповсюдження ненависті;
- погані засоби фільтрації.

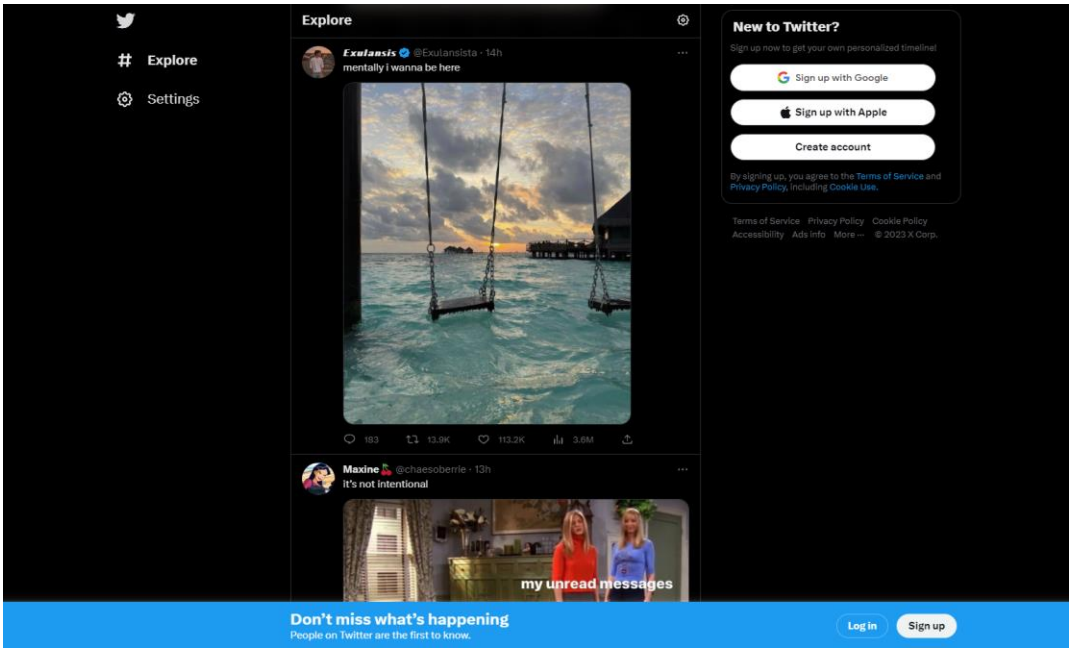

Рисунок 1.2 – Головна сторінка соціальної мережі Twitter

Соціальна мережа Telegram

Назва: Telegram (рис. 1.3)

Архітектура: Клієнт серверна

Мова реалізації: інтерфейс Qt, C++, Java

Мета: клієнт система миттєвого обміну повідомленнями

Переваги:

1) можна пересилати відео, фото, аудіофайли;
2) є канали у яких можна дізнатись багато інформації зі світу;
3) можна спілкуватись відео повідомленнями та голосовими повідомленнями;
4) створювати свої стікери;
5) використовується з підключенням до інтернету, але дуже малими витратами;
6) є кеш у котрому зберігаються завантажені файли і відтворюються коли немає інтернету;

Недоліки:

1) застосунок дуже конфіденційний що дає змогу використовувати його у злочинних цілях;
2) іноді не знайомі люди присилають спам-повідомлення;
3) іноді довго завантажуються великі файли;

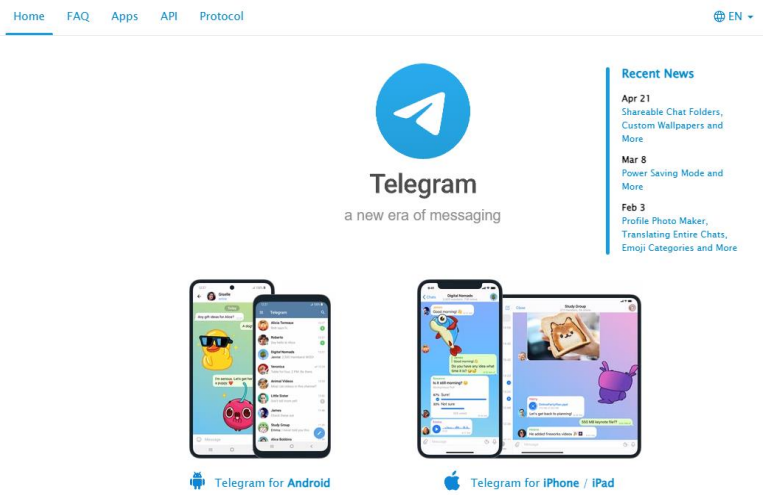

Рисунок 1.3 – Головна сторінка соціального вебзастосунку Telegram

Більш детальний опис про розробників застосунків, мов реалізації архітектури, їх порівняння описані у таблиці 1.1.

Таблиця 1.1 – Порівняльна таблиця аналогів

Назва	Розробник	Мова реалізації	Недоліки
Viber	Viber Media S.à r.l.	Java, C, Python, C++ і Objective-C	- Потребує багато пам'яті телефону - Використовує багато інтернет трафіку - Обов'язкова наявність Sim - карти
Telegram	Telegram FZ LLC Telegram Messenger Inc.	C++, Java	- Немає особистої сторінки - Потребує Sim – карту - Розповсюдження забороненого контенту - Деякий функціонал доступний лише за підпискою
Twitter	Twitter, Inc.	JavaScript, Ruby, Scala, Java	- Дуже обмежена кількість символів у повідомленнях чи публікаціях - Жодних засобів проти розповсюдження ненависті - Погані засоби фільтрації

Проаналізувавши сучасні соціальні мережі було спроєктовано та створено свій вебзастосунок з урахуванням всіх недоліків, який поєднує найкращі аспекти аналогів, і враховує їх недоліки.

1.3 Призначення та межі проєкту

Призначення застосунку, для якого розробляється програмне забезпечення:

Полегшення комунікації між людьми зі всього світу шляхом розробки сучасного вебзастосунку, простого у розумінні та захищеного, котрим можна користуватись у реальному часі.

Загальний опис характеристики користувачів:

У системі повинні бути передбачені наступні ролі користувачів з різними правами доступу до функцій системи:

- Гість (Unauthorized user) – роль з обмеженими правами доступу до функцій системи, присвоєна всім користувачам що не пройшли авторизацію;

- Адміністратор (Administrator) – роль з усіма правами доступу у системі, для забезпечення безпеки системи, роль присвоєна лише одному користувачеві;

- Користувач (Authorized user) – роль з повноцінними правами доступу до функцій системи, може взаємодіяти зі всіма користувачами на одному рівні.

Загальні обмеження

Обмеженням для функціонування застосунку є наявність інтернет-з’єднання користувача.

Опис структури системи

Програмне забезпечення, що розробляється, має наступні складові:

- Backend (серверна частина вебзастосунку, відповідає за доступ до даних, в даному випадку Next.js)

- Frontend (клієнтська частина вебзастосунку, відповідає за представлення, в даному випадку HTML, JavaScript [2], CSS [3], react-redux, TypeScript)

- База даних (MongoDB)

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Жорстких вимог до технічного забезпечення немає, користувач повинен мати комп’ютер чи ноутбук, завантажений браузер та підключення до мережі Інтернет.

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмного забезпечення складається з мов програмування (JavaScript, TypeScript), їх фреймворків (Next.js, react-redux), та використані сервіси Google (Cloud).

Системне програмне забезпечення

Для написання front-end частини обрано фреймворк React/Redux, для back-end частини – фреймворк Next.js та мову програмування JavaScript. Авторизація відбувається за допомогою Google Authorization [15]. В якості БД обрано MongoDB. [17]

Практичність:

- вебзастоунок має бути простим і зручним у використанні;
- дизайн вебзастосунку має бути адаптивним та зрозумілим для використання.

Надійність:

- інформація про користувача зашифрована та знаходиться базі даних;
- програмна обробка можливих помилок;
- стабільність роботи при підключенні до інтернету.

Доступність

Вебзастосунок повинен працювати на десктоп та мобільних пристроях в будь-якому браузері

Опис основних функцій системи

Програмне забезпечення соціальної мережі, яка розробляється в ході кваліфікаційної роботи, має наступний набір функціональних можливостей:

1. авторизація користувача;
2. редагування профілю користувача;
3. додавання у друзі інших користувачів;
4. видалення із друзів інших користувачів;
5. створення онлайн чатів з іншими користувачами;
6. пошук користувачів;
7. створення та видалення постів;
8. коментування своїх та інших постів;
9. проставлення та прибирання реакцій під постами;
10. завантаження картинок до постів і чатів;

ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

Інтерфейс користувача

Вебклієнт має задовольняти усім вимогам UX та UI, що дозволить користувачу затратити найменше часу на розуміння роботи системи. Шаблон сторінки повинен бути розділений на дві частини, де перша частина це інфографіка про існуючі сторінки та друзів користувача, друга частина це пошук користувачів та перегляд публікацій.

Апаратний інтерфейс

Апаратний інтерфейс – це пристрій користувача (ПК, ноутбук чи планшет), який він буде використовувати для взаємодії зі сторінками вебзастосунку.

Висновки до розділу 1

У результаті написання розділу 1 КРБ було виконано завдання для досягнення мети, яка полягала в аналізі предметної області та вимог до системи, що розробляється.

У розділі 1 було проведено аналіз предметної галузі. Також було проаналізовано наявні аналоги. На основі проведеного аналізу було з’ясовано, що соціальні мережі широко використовуються.

Кожен розглянутий аналог має свої переваги та недоліки. Також було поставлено завдання, у якому були описані основні функції системи, ролі користувачів у системі, описана структура системи визначено призначення та загальні обмеження, надійність та доступність системи.

2 МОДЕЛЮВАННЯ СИСТЕМИ ВЕБЗАСТОСУНКУ ОНЛАЙН-КОМУНІКАЦІЇ КОРИСТУВАЧІВ

2.1 Варіанти використання (Use cases)

В цьому розділі проведено ознайомлення зі сценаріями використання системи. Для цього розроблено діаграму варіантів використання та описано варіанти використання системи за допомогою сценарної техніки опису взаємодії Use Case, або ж її ще називають діаграмою прецедентів. Діаграма прецедентів [7] фіксує домовленість між користувачами системи щодо його поведінки. Вона описує поведінку застосунку при його відповідях на запит одного з учасників, що називається дійовою особою у різноманітних умовах.

Застосунок включає три дійові особи: гість, користувач, адміністратор. Кожен має свою роль та функції на сайті. Інформацію про акторів представлено у таблиці 2.1.

Таблиця 2.1 – Інформація про акторів

Назва	Опис
Гість	Гість може авторизуватись, передивлятись та шукати профілі авторизованих користувачів.
Авторизований користувач	Авторизовані користувачі можуть додати чи видалити із друзів інших користувачів, писати коментарі під своїми та іншими постами, проставляти під ними реакції, також можуть редагувати свою сторінку за своїм бажанням, можуть створювати та спілкуватись з іншими користувачами вебзастосунку.

Завершення таблиці 2.1

Адміністратор	Може управляти акаунтами, наповнює сайт інформацією.

Діаграму використання можна побачити на рис. 2.1. На ній зображено дійові особи та доступні їм функції вебзастосунку.

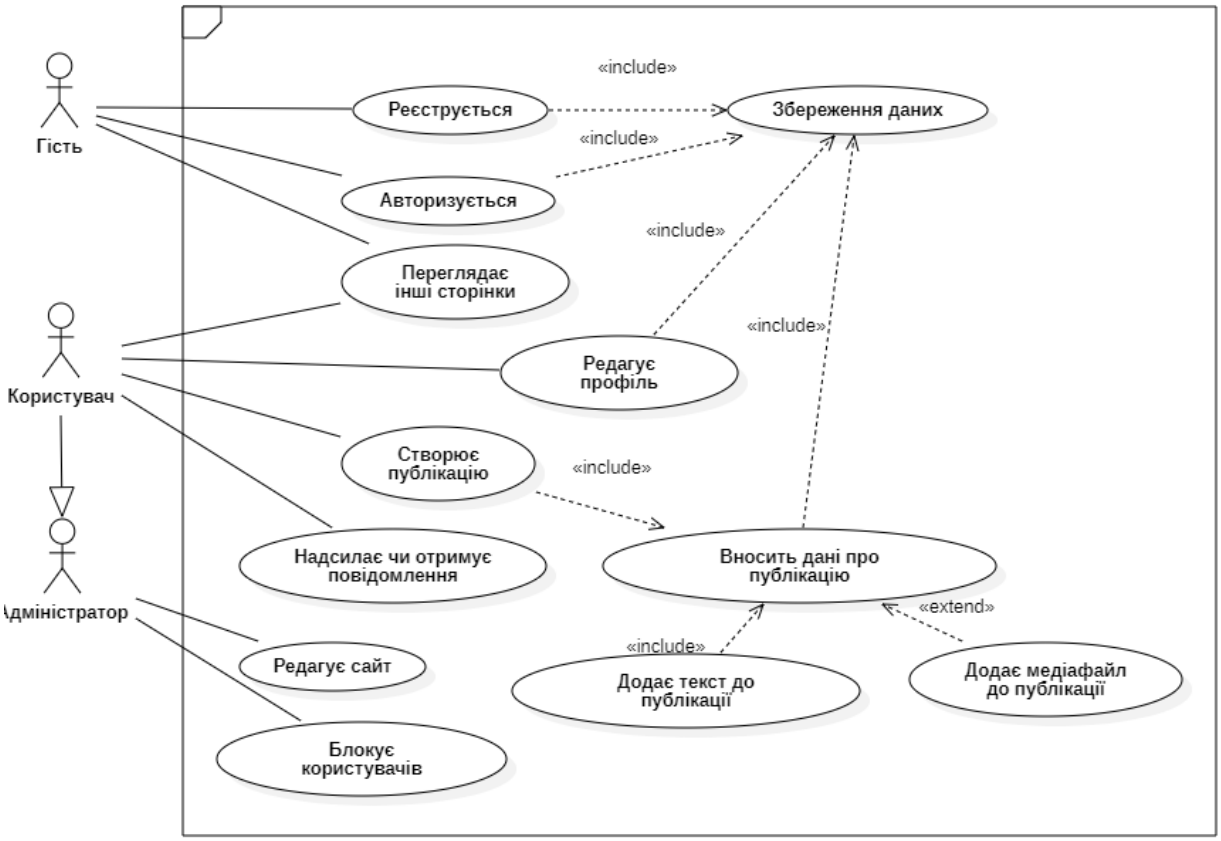

Рисунок 2.1 – Діаграма варіантів використання системи

На цій діаграмі варіантів використання зображено 3 ролі користувачів, гість може авторизуватись, передивлятись та шукати профілі авторизованих користувачів, авторизовані користувачі можуть додати чи видалити із друзів інших користувачів, писати коментарі під своїми та іншими постами, проставляти під ними реакції, також можуть редагувати свою сторінку за своїм бажанням.

2.2 Сценарії використання

Сценарії використання дозволяють сформувати та задокументувати вимоги для полегшення розробки та підтримки застосунку. Для демонстрації можливих форм написання сценаріїв, перші 3 сценарії представлені в:

1. короткій формі;
2. поверхневій формі;
3. повній формі.

Сценарій №1 (коротка форма): Коментування публікації іншого користувача.

Користувач заходить за посиланням на сайт вебзастосунку та переходить на сторінку авторизації. Користувач обирає авторизацію через гугл, обирає свій гугл аккаунт. Після успішної авторизації потрапляє на сторінку свого профіля, де користувач натискає на кнопку «Новини». Він потрапляє до усіх новин, публікацій всіх користувачів з інтернету, натискає кнопку коментування під будь-якою публікацією, відкривається поле вводу тексту та кнопка обирання медіа файлу. Користувач вводить текст не обираючи медіа файл та натискає кнопку «Відправити», натискає на пусте місце, і вікно коментування зачиняється.

Сценарій №2 (поверхнева форма): Додавання публікації.

Головний сценарій:

Користувач відкриває сторінку соціальної мережі та авторизується на сайті. Користувач обирає стрічку новин, переглядає публікації інших користувачів, та бажає викласти свою публікацію, обирає поле створення публікації вводить текст та обирає картинку чи відео для публікації, натискає на кнопку «Опублікувати». Система розміщує публікацію на сторінці користувача та відображає її в стрічці новин інших користувачів, які мають доступ до цього контенту.

Альтернативні сценарії:

1. користувач перейшов на сторінку новин, але без авторизації не має поля створення публікації;

2. користувач обрав поле створення та натиснув на «Опублікувати» не ввівши жодних даних про публікацію, користувач отримує повідомлення: «Додайте текст, або файл»;

3. опублікована публікація має вікове обмеження, вона буде схована для користувачів з віком менше 18;

4. при авторизації користувач не має гугл аккаунту, його перенесе на сторінку створення гугл аккаунту, потім він сам повинен повернутись на сайт вебзастосунку;

5. технічний збій роботи системи. Видається повідомлення «Немає зв’язку з сервером».

Таблиця 2.2 – Сценарій №3 (повний): Додавання друга в чат-платформі соціальної мережі.

Primary actor	Користувач
Scope	Чат-платформа для соціальних мереж
Level	Мета користувача
Preconditions	Користувач увійшов у свій обліковий запис і має доступ до платформи

Продовження таблиці 2.2

Stakeholders and interests	1. користувач: зацікавлений у додаванні друзів до своєї соціальної мережі та розбудові соціальних зв'язків; 2. чат-платформа для соціальних мереж: Зацікавлена у створенні зручного інтерфейсу для додавання друзів та покращенні користувацького досвіду; 3. друзі користувача: зацікавлені у прийнятті або відхиленні запитів на додавання в друзі та побудові власних соціальних зв'язків; 4. рекламодавці: Зацікавлені в охопленні ширшої аудиторії через соціальну мережу та підвищенні впізнаваності бренду.
Main Success Scenario: 1. користувач переходить до розділу «Додати друга» на платформі чату в соціальній мережі; 2. користувач вводить ім'я користувача, якого він хоче додати; 3. система шукає друга і відображає інформацію про його профіль; 4. користувач переглядає інформацію профілю, щоб переконатися, що він знайшов потрібного друга; 5. користувач натискає кнопку «Додати друга», щоб відправити запит на додавання; 6. система надсилає користувачеві повідомлення з проханням прийняти або відхилити запит на дружбу; 7. друг отримує повідомлення і приймає або відхиляє запит на дружбу;	

8. після того як друг приймає запит, він додається до списку друзів користувача, а користувач також додається до списку друзів друга; 9. система відповідним чином оновлює списки друзів користувача і друга; 10. тепер користувач і друг можуть бачити публікації один одного і спілкуватися один з одним в чаті соціальної мережі.

Продовження таблиці 2.2

Result	Користувач додав друга.
Extensions	
a*	Технічні проблеми: Якщо в системі виникають технічні проблеми, такі як простої сервера або проблеми з підключенням до мережі, користувач може не мати можливості додати друга. У цьому випадку система відобразить повідомлення про помилку, інформуючи користувача про проблему і пропонуючи йому повторити спробу пізніше.
1a	Якщо друг відхиляє запит, він не додається до списку друзів користувача, а користувач отримує повідомлення про те, що запит на дружбу було відхилено.
2a	Система не оновлює списки друзів, а користувач і друг залишаються не пов’язаними на платформі чату в соціальній мережі.

Продовження таблиці 2.2

3а	Неправильні дані користувача: Якщо користувач вводить невірне ім’я або ім’я користувача, яке не існує на платформі, система відобразить повідомлення про помилку, інформуючи користувача про те, що друга не вдалося знайти.
4а	Запит на дружбу вже надіслано: якщо користувач вже надіслав запит на дружбу тому ж користувачеві, система покаже повідомлення про те, що запит очікує на розгляд, і користувачеві потрібно почекати, поки друг прийме або відхилить запит.
4а	Запит на дружбу відхилено: Якщо друг відхиляє запит користувача на додавання в друзі, система відобразить повідомлення, що запит на додавання в друзі було відхилено, і вони не будуть підключені до платформи.

Завершення таблиці 2.2

Special Requirements	Продуктивність: Соціальна мережа повинна безперебійно працювати на різних пристроях та конфігураціях, з мінімальними затримками та часом завантаження. Локалізація: Соціальна мережа повинна бути локалізована різними мовами, щоб охопити ширшу аудиторію. Сумісність: Вебзастосунок має бути сумісним з різними операційними системами, пристроями та розмірами екранів.
Frequency of Occurrence	Система може працювати майже безперервно.
Miscellaneous (Open Issues)	Покращення користувацького інтерфейсу для покращення взаємодії з користувачем 1. посилити заходи безпеки для запобігання злому або несанкціонованому доступу; 2. розробити систему повідомлень про неприйнятний контент або поведінку; 3. впровадити функції для підтримки мультимедійного контенту, наприклад, фотографій і відео; 4. дослідити способи мінімізації затримок і підвищення загальної продуктивності системи.

Таблиця 2.3 – Сценарій №4: Редагування профілю користувача

Діючі особи	Авторизований користувач
Мета	Редагувати профіль користувача
Успішний сценарій: 1) користувач увійшов у меню редагування профілю; 2) користувач змінив ім’я, місто, дату народження, інформацію вказав у правильному форматі, та натискає на кнопку «Оновити»; 3) система обробляє запит та оновлює дані користувача у реальному часі.	
Результат	Збережено обрані дані користувача.

Таблиця 2.4 – Сценарій №5: Відправлення повідомлення

Діючі особи	Авторизовані користувачі
Мета	Відправлення повідомлення
Успішний сценарій: 1) користувач переходить на сторінку свого друга; 2) користувач натискає кнопку «Відправити повідомлення»; 3) пише повідомлення; 4) натискає «Надіслати повідомлення»; 5) система записує повідомлення до БД.	
Результат	Повідомлення відправлено.

Таблиця 2.5 – Сценарій №6: Проставлення реакцій під публікаціями

Діючі особи	Авторизовані користувачі
Мета	Залишення реакції користувача під публікацією
Успішний сценарій: 1) користувач переходить на сторінку свого друга чи на стрічку новин; 2) користувач переглядає публікації на сторінці; 3) користувач натискає кнопку вподобайки; 4) система зберігає зміни в БД.	
Extensions: - якщо вподобайка вже була проставлена то вона знімається.	
Результат: Реакцію під публікацією було збережено.	

2.3 Діаграми взаємодії (Interaction diagram)

Для відображення usecase графічним способом існують форми діаграм взаємодії: Sequence diagram (рис. 2.2) та Collaboration diagram, обидві форми діаграми зображають одну і ту ж інформацію, але різним виглядом. Було створено діаграми взаємодії для useCase:

Актори: Сайт соціальної мережі, користувачі

Компоненти системи:

- база даних профілів користувачів;
- система обміну повідомленнями;
- налаштування конфіденційності.

Етапи використання:

1. користувач заходить у свій акаунт на сайті соціальної мережі;
2. сайт отримує інформацію про профіль користувача з бази даних і відображає її;

3. користувач встановлює налаштування конфіденційності для своєї системи обміну повідомленнями;

4. користувач надсилає повідомлення другу за допомогою системи обміну повідомленнями;

5. друг отримує повідомлення і відповідає;

6. система повідомлень сповіщає користувача про нове повідомлення від його друга;

7. користувач отримує повідомлення від свого друга і відповідає на нього.

Деталі повідомлення:

Authorization: Ініціює запит на вхід від користувача в соціальну мережу для перевірки його облікових даних.

Get profile: Отримує інформацію про профіль користувача з бази даних і відображає її на сайті.

Set privacy settings: Дозволяє користувачеві встановити налаштування конфіденційності для своєї системи обміну повідомленнями, наприклад, хто може надсилати йому повідомлення і хто може бачити його онлайн-статус.

Send message: Дозволяє користувачеві відправити повідомлення другу за допомогою системи обміну повідомленнями.

Receive message: Сповіщає користувача про отримання нового повідомлення від друга.

Respond message: Дозволяє користувачеві відповісти на повідомлення від свого друга за допомогою системи обміну повідомленнями.

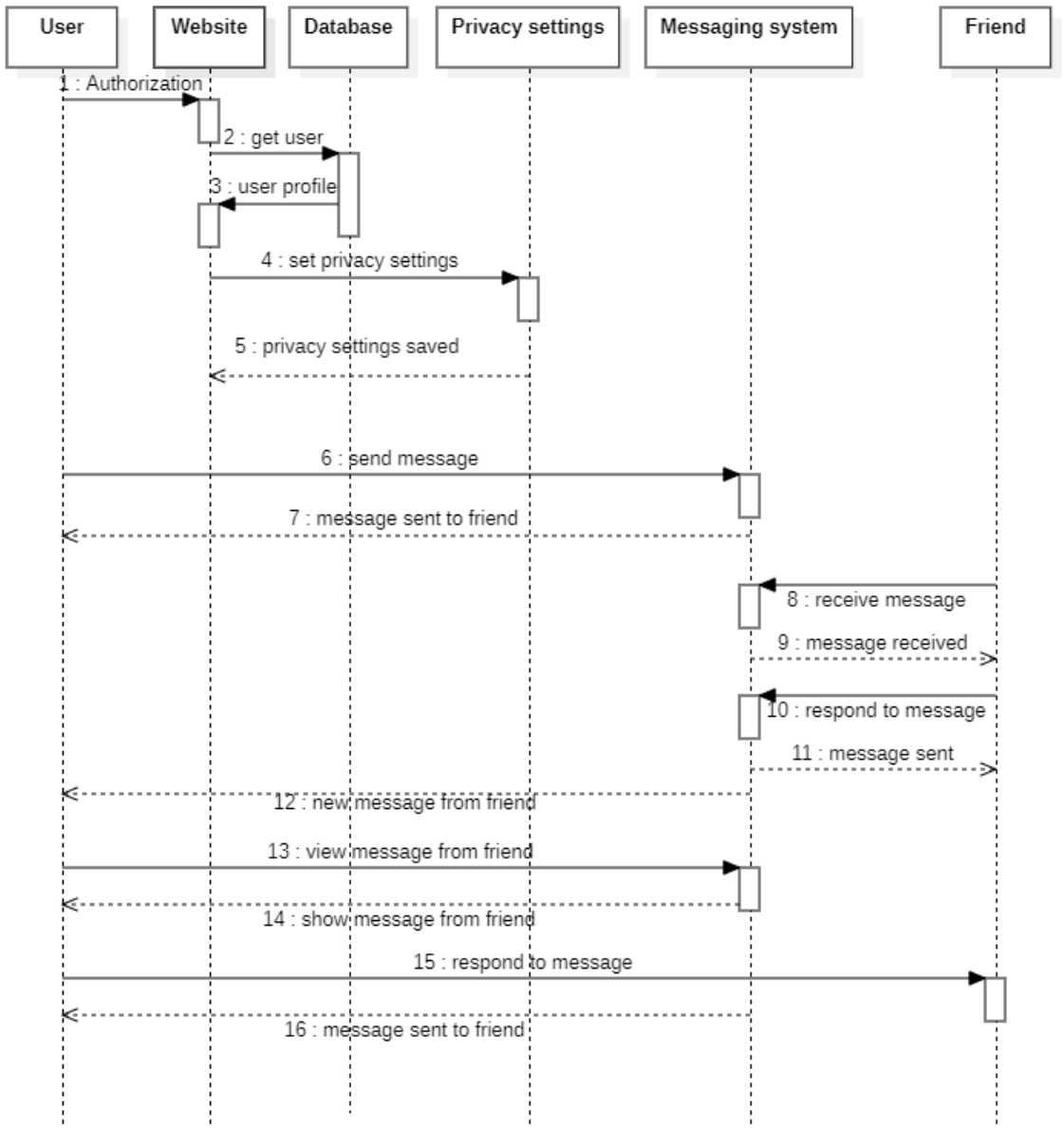

Рисунок 2.2 – Sequence Diagram для Use Case

Головна мета діаграми Collaboration (рис. 2.3) полягає у візуалізації, які об’єкти взаємодіють один з одним та як вони обмінюються повідомленнями у межах системи.

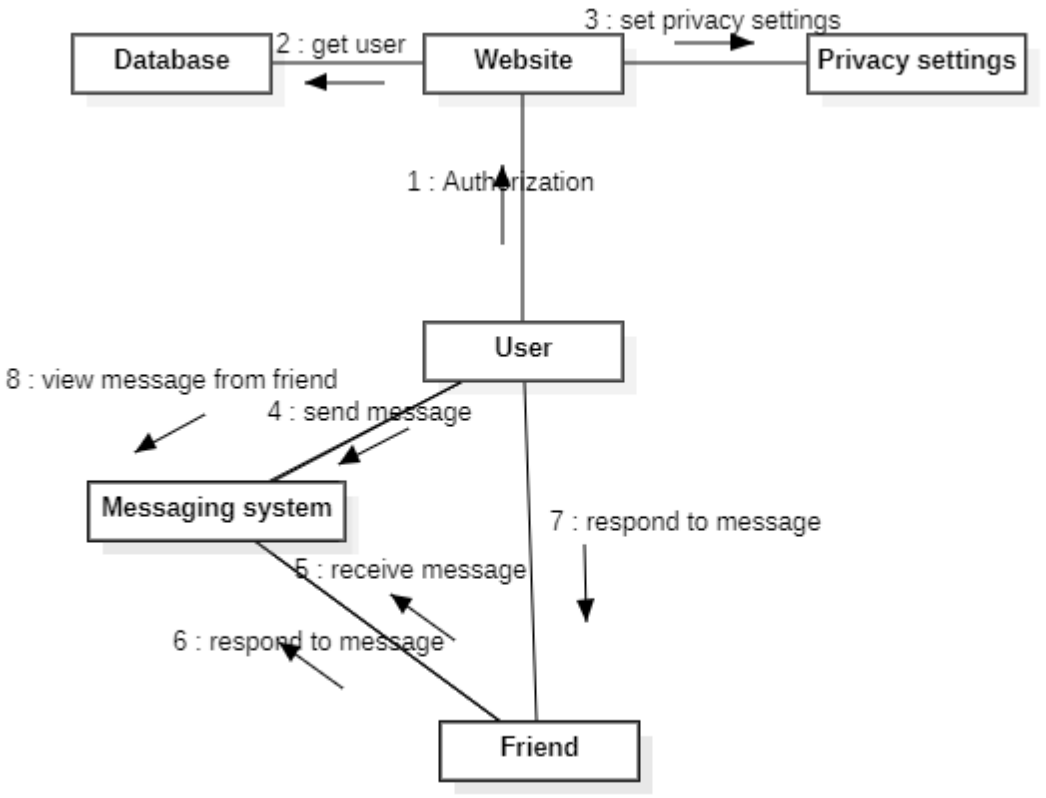

Рисунок 2.3 – Collaboration diagram для Use Case

Ця діаграма [7] може бути корисна для розуміння взаємодії між різними об’єктами, компонентами або модулями системи та може допомогти при проєктуванні та розробці програмного забезпечення. Також вона може допомогти виявити помилки в процесі взаємодії між об’єктами, що може призвести до більш ефективної роботи всієї системи.

Висновки до розділу 2

В даному розділі було проведено ознайомлення зі сценаріями використання системи, а також розроблено діаграму варіантів використання. Використано сценарну техніку опису взаємодії Use Case для опису варіантів використання системи. Діаграма прецедентів фіксує домовленість між користувачами системи щодо їх поведінки.

Сценарії використання допомагають сформувати та задокументувати вимоги для полегшення розробки та підтримки системи. Були представлені три сценарії у короткій, поверхневій та повній формі. Перший сценарій описує коментування публікації іншого користувача, другий - додавання публікації, а третій - проставлення реакцій під публікаціями.

Також були наведені діаграми взаємодії: Sequence diagram та Collaboration diagram. Ці діаграми візуалізують взаємодію об'єктів та процеси, що відбуваються у системі.

Загальною метою цього розділу було описання варіантів використання системи та створення вимог для полегшення розробки та підтримки системи.

3 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ОНЛАЙН-КОМУНІКАЦІЇ КОРИСТУВАЧІВ

3.1 Діаграма класів (Class diagram)

Діаграма класів показує структуру класів застосунка, зв’язки між ними та атрибути кожного окремого класу. Вона є частиною уніфікованої мови моделювання (UML).

У свою чергу UML – це мова графічного опису для моделювання об’єктів у розробці програмного забезпечення, бізнес-процесів, системної інженерії та візуалізації організаційної структури. Вона містить:

- класи;
- атрибути класів;
- операції (методи);
- взаємозв’язки між об’єктами.

На рис. 3.1 зображена діаграма класів фронтенд частини до вебзастосунку онлайн комунікації користувачів.

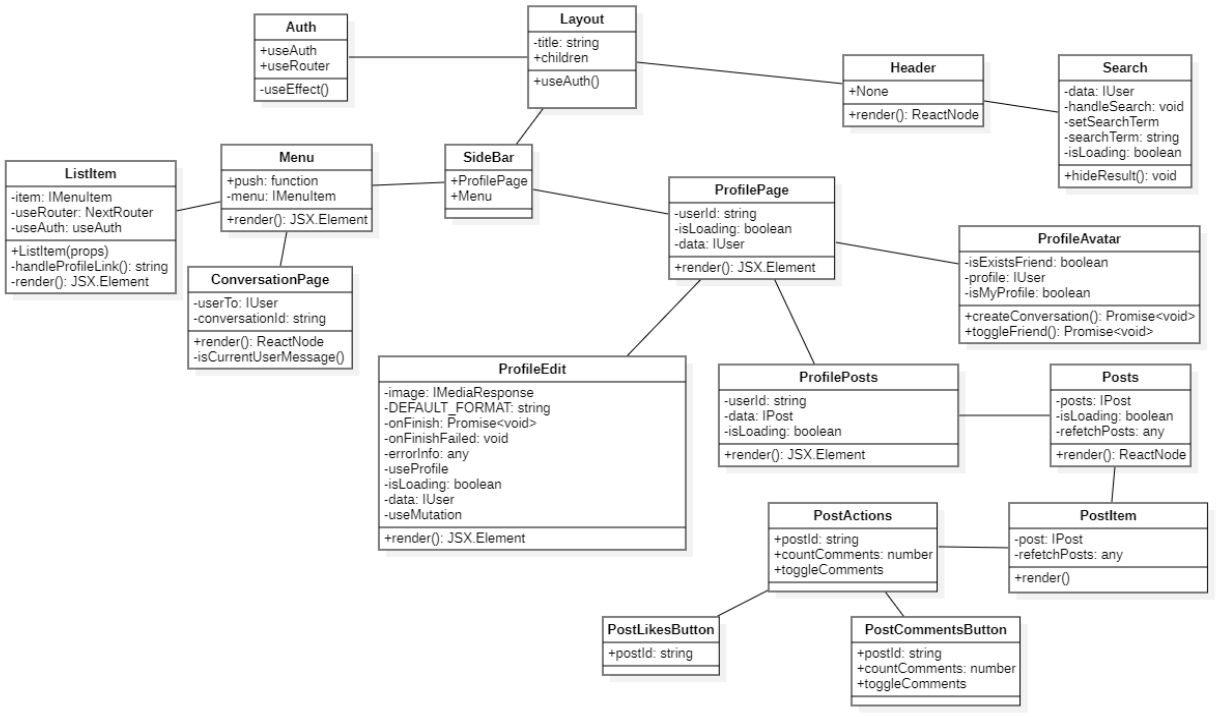

Рисунок 3.1 – Діаграма класів

3.2 Діаграма станів (State diagram)

Діаграма [7] станів визначає зміну станів об’єкту у часі, це діаграма моделювання поведінки у UML. На рис. 3.2 зображено діаграму станів для вебзастосунку.

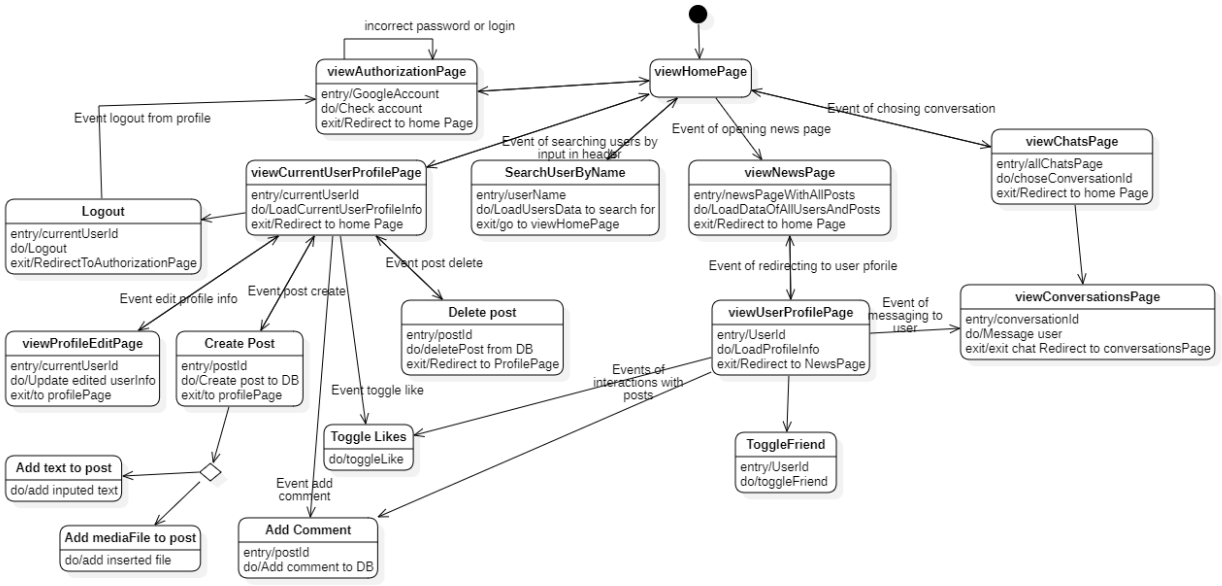

Рисунок 3.2 – Діаграма станів для вебзастосунку онлайн комунікації користувачів

На діаграмі зображено переходи та стани між сторінками вебзастосунку. Так наприклад зі сторінки HomePage користувач переходить до сторінки авторизації та може увійти до системи через гугл аккаунт, якщо він не має гугл аккаунту, гугл запропонує йому створити аккаунт.

У разі якщо авторизація вдала користувач може перейти до свого профілю в котрому може: створити пост, видалити пост, редагувати профіль, чи вийти з акаунту, в процесі створення посту користувач може додавати текст чи медіа файл, до майбутнього посту. Якщо користувач перейде до сторінки новин, він зможе переглянути існуючи пости усіх користувачів, та клікнути на фотографію профіля користувача, перейти до його профілю, в котрому можна: додати користувача у друзі, написати йому повідомлення, лайкнути чи додати коментар до постів користувача. Також з головної сторінки можна перейти до сторінки чатів, де можна обрати чат з користувачем та перейти до нього.

3.3 Діаграма діяльності

У цій діаграмі діяльності (рис. 3.3) використовуються два об’єкти: користувач та сервер. Користувачі взаємодіють із сервером, надсилаючи запити на логін/реєстрацію, пошук друзів, відправлення повідомлень та вихід з аккаунту. Сервер перевіряє запити користувачів та виконує відповідні дії, такі як перевірка інформації, пошук друзів, відправлення повідомлень та підтвердження виходу з аккаунту.

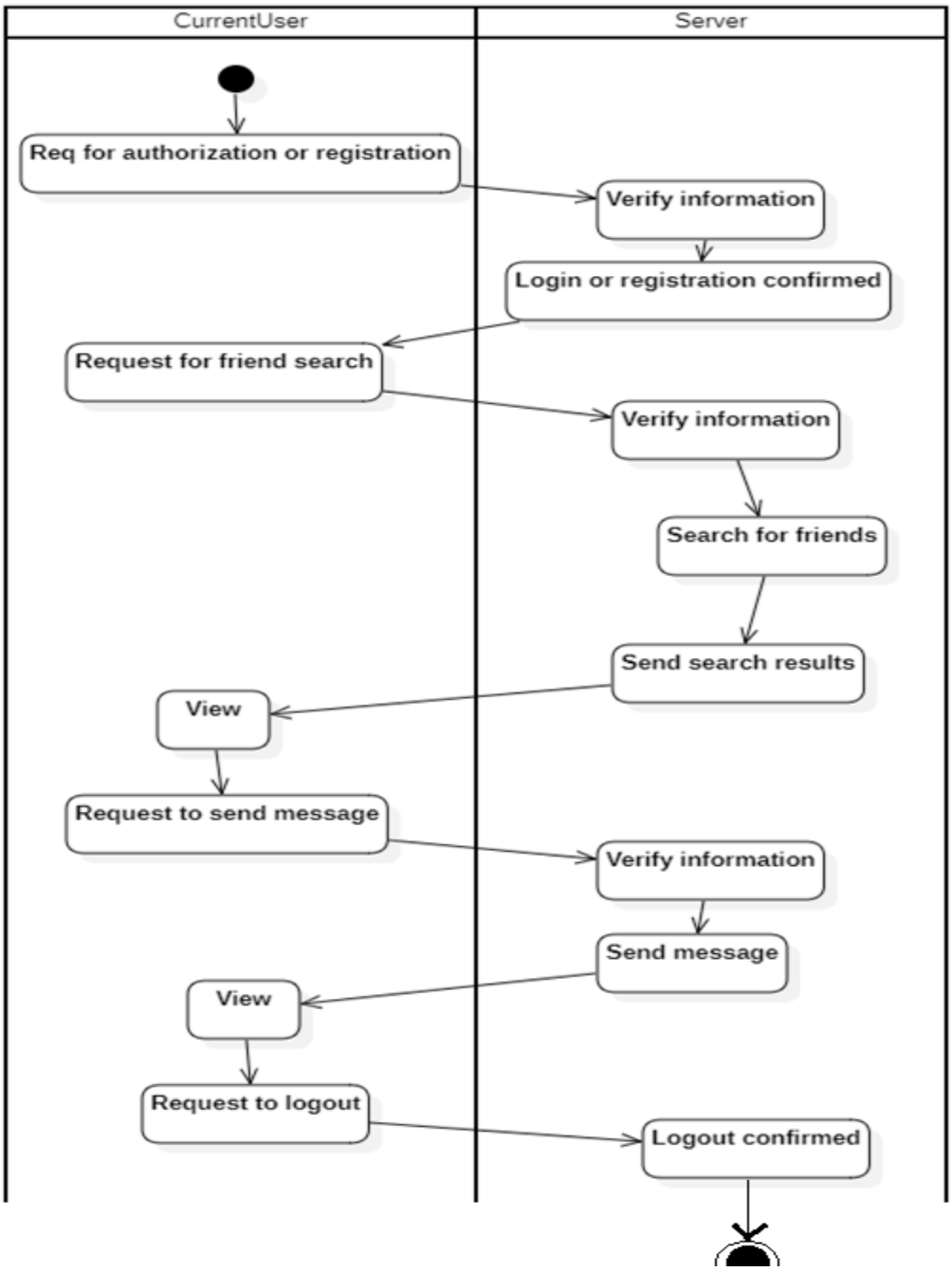

Рисунок 3.3 – Діаграма діяльності

У другій діаграмі діяльності використовуються три об’єкти: користувач А, сервер та користувач В. Користувачі можуть авторизуватись/реєструватись, шукати друзів, надсилати запити на дружбу та надсилати повідомлення один одному. Сервер перевіряє запити користувачів та виконує відповідні дії, такі як перевірка інформації, пошук друзів, надсилання повідомлень та відправлення повідомлень про події. Також сервер обробляє запит приймання користувачів у друзі. (рис. 3.4)

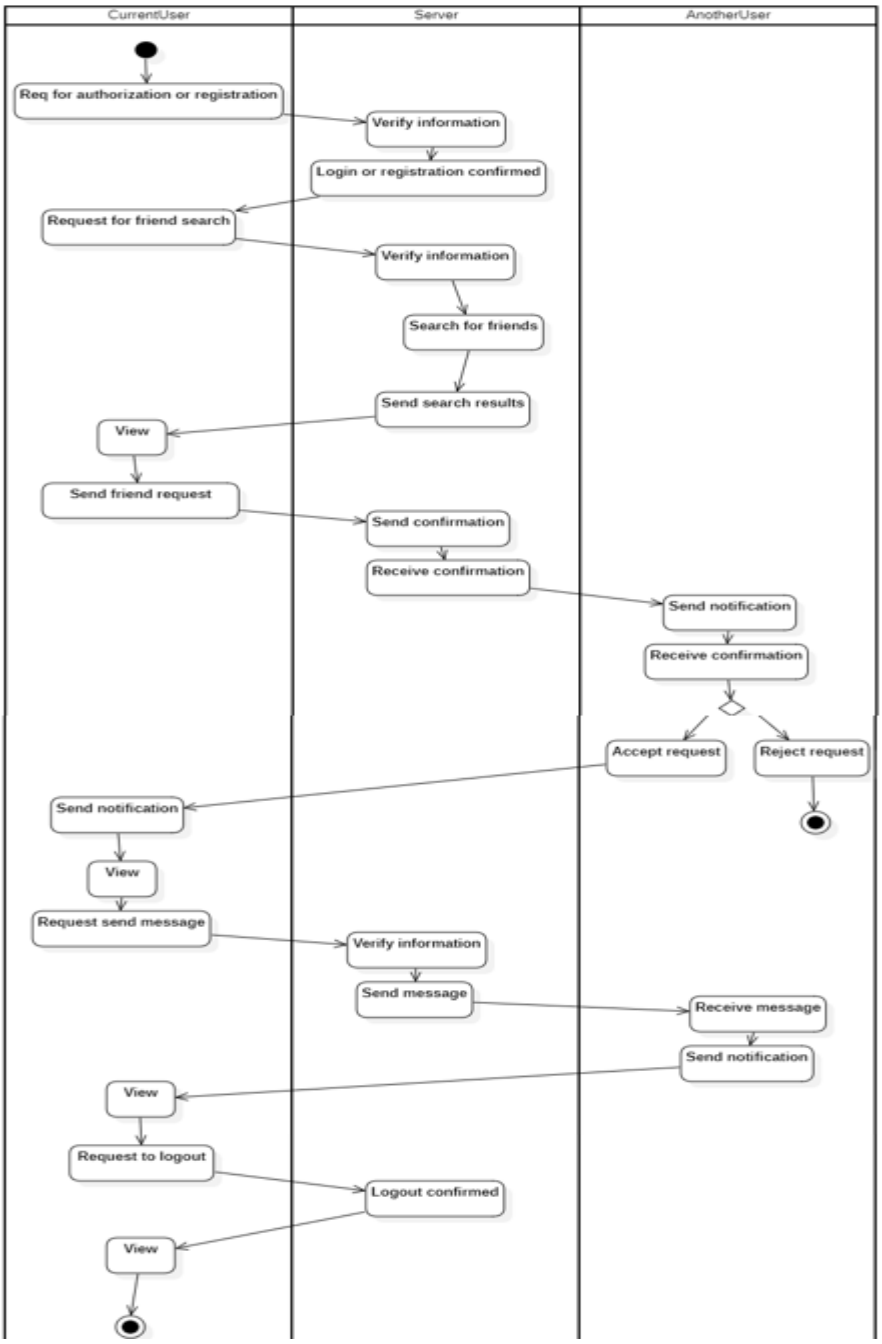

Рисунок 3.4 – Діаграма діяльності з трьома об’єктами

Висновки до розділу 3

У результаті написання розділу 3 КРБ було розроблено діаграму класів для визначення структури системи і створено частину проєкту на основі визначеної структури класів. Також було розроблено діаграму станів, що дає змогу краще зрозуміти функції застосунку, було створено діаграму діяльності яка відображає деяку взаємодію та діяльності між компонентами системи.

Для проєктування діаграм було використано мову моделювання UML. Для створення діаграм було використано таке програмне забезпечення, як StarUML. Використані UML-діаграми в даному розділі чітко демонструють структуру та вимоги до застосунку, а також описують архітектуру програмного комплексу.

4. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Обґрунтування вибору технологій для реалізації системи. Вибір програмних засобів для розробки

Розробка вебзастосунків вимагає ретельного вибору технологій і підходів. Важливо зрозуміти, як вони були відібрані для проєкту.

При виборі технології важливо враховувати наступне:

- складність програмного забезпечення;
- наявність ресурсів;
- наявність готових компонентів;
- наявність технічної документації;
- характеристики якості пз;
- витрати на технологію;
- ліцензійну політику;
- вимоги безпеки.

У якості технології розробки програмного забезпечення було обрано React Redux, у якості бази даних було обрано MongoDB. Для цього було обрано відповідно програмне забезпечення – Web Storm (для написання програмного коду) та MongoDB Compass (для створення бази даних).

React - це бібліотека JavaScript для створення користувацьких інтерфейсів, яка дозволяє розробникам створювати компоненти інтерфейсу, які можна повторно використовувати.

Redux - це передбачуваний контейнер стану для JavaScript-застосунків, який допомагає керувати станом за більш організовано.

TypeScript - це надмножина JavaScript, яка забезпечує статичну перевірку типів та інші функції для покращення продуктивності розробника та надійності додатку.

Коли ці технології використовуються разом, розробники можуть створювати потужні, підтримувані додатки з чітким розподілом обов'язків між

компонентами користувацького інтерфейсу, керуванням станом та бізнес-логікою.

Архітектура MVC (Model-View-Controller) – розділяє застосунок на три основні групи компонентів; модель, вид та контролер.

На рис. 4.1 зображена архітектура MVC. [10]

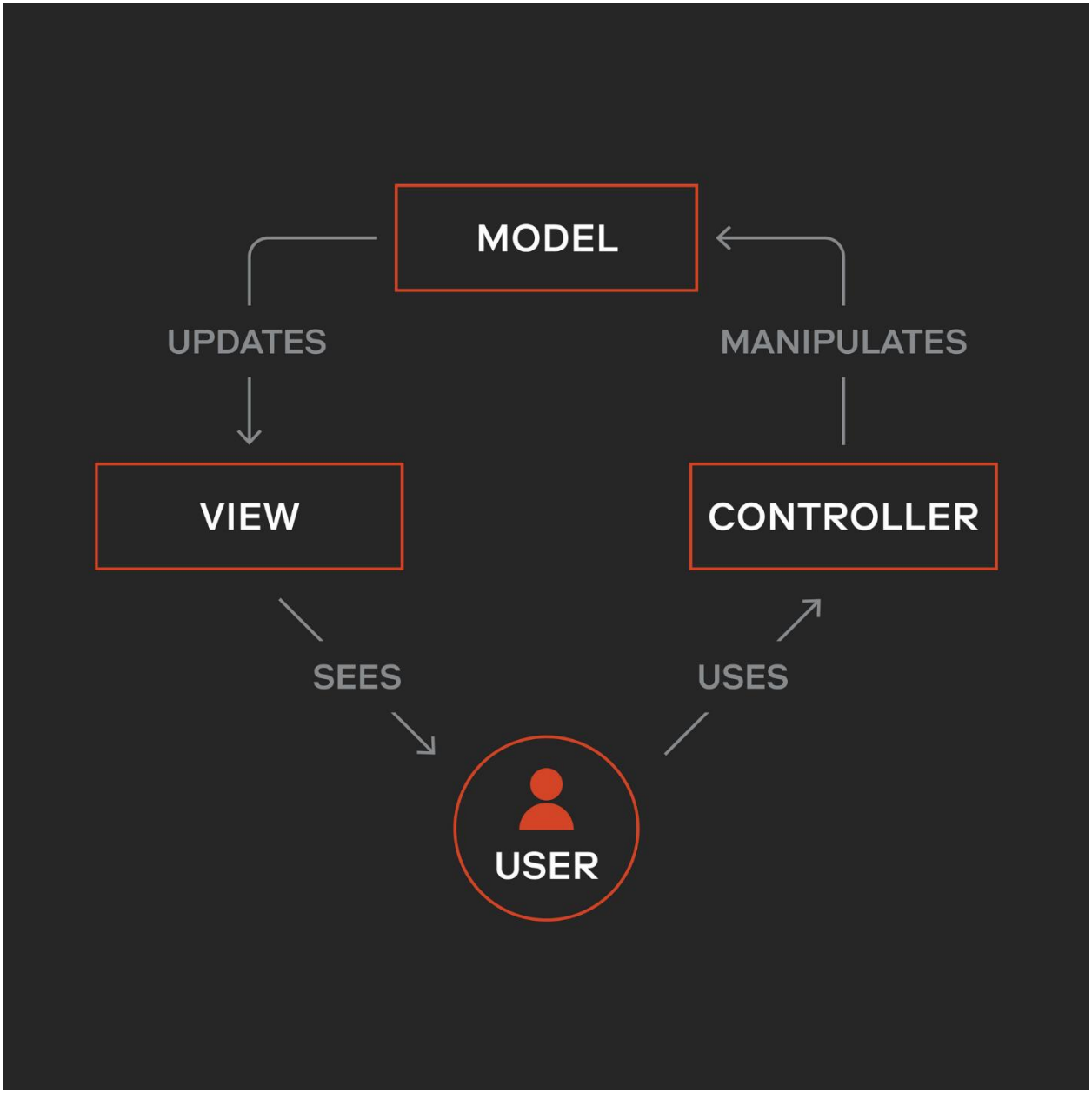

Рисунок 4.1 – Архітектура MVC

MVC - це архітектурний шаблон, який розділяє застосунок на три основні компоненти:

Модель (Model) - це компонент, який відповідає за обробку даних та логіку бізнес-процесів. Вона представляє структуру даних та методи для їх обробки. Модель не залежить від інтерфейсу користувача або відображення даних.

Вид (View) - це компонент, який відображає дані моделі та взаємодіє з користувачем. Він відповідає за представлення даних моделі у зручному для сприйняття користувача форматі. Вид також може приймати вхідні дані від користувача та передавати їх контролеру.

Контролер (Controller) - це компонент, який відповідає за обробку запитів користувача та управління потоком даних між моделлю та видом, а також виконує необхідні дії згідно з діями користувача. Контролер також відповідає за оновлення виду або моделі при зміні даних.

Основна перевага моделі MVC полягає в розділенні відповідальності між компонентами, що сприяє полегшенню розробки, тестуванню та підтримці програмного забезпечення. Вона також дозволяє більшу гнучкість та повторне використання коду завдяки незалежності компонентів.

MVC широко використовується у багатьох різних платформах та мовах програмування, включаючи веброзробку, комп’ютерні та мобільні застосунки.

Для розробки вебзастосунку використовується JavaScript, її бібліотека для розробки користувацьких інтерфейсів React, бібліотека управління станом Redux, для JavaScript застосунків, також використовується Next.js – це фреймворк JavaScript для розробки вебзастосунків з підтримкою рендерінгу на стороні сервера та на стороні клієнта. Також для розробки використано нереляційну базу даних MongoDB, яка зберігає дані у вигляді документів у форматі BSON. Вона широко використовується у веброзробці та інших проєктах, де потрібно зберігати та обробляти великі обсяги даних. Більш детальний опис технологій наведено у таблиці 4.1.

Таблиця 4.1 – Порівняння використаних технологій [11]

Технологія	Опис	Швидкість	Вартість	Розмір	Складність
MongoDB	Нереляційна база даних, що зберігає дані у вигляді документів у форматі BSON. Вона широко використовується для зберігання та обробки великих обсягів даних.	Висока	Безкоштовна (основна версія), комерційна (розширений функціонал)	Залежить від обсягу даних	Середня
JavaScript	Мова програмування, що використовується для розробки вебзастосунків та динамічного взаємодії з вебсторінками.	Висока	Безкоштовна	Залежить від розміру коду	Низька
React	Бібліотека JavaScript для розробки користувацьких інтерфейсів. Вона дозволяє розбити вебзастосунок на компоненти та ефективно управляти їх станом. React є популярним інструментом у веброзробці.	Висока	Безкоштовна	Залежить від розміру проєкту	Середня
Redux	Бібліотека управління станом для JavaScript додатків. Вона використовується з React та іншими фреймворками для зберігання та управління станом додатку. Redux спрощує керування складними станами додатку.	Висока	Безкоштовна	Залежить від розміру State	Середня
Next.js	Фреймворк JavaScript для розробки вебзастосунків з підтримкою SSR та CSR. Він надає можливість створювати реактивні та швидкодіючі вебзастосунки з покращеною маршрутизацією та іншими корисними функціями.	Висока	Безкоштовна	Залежить від розміру проєкту	Середня

4.2 Опис програмної реалізації та макети проєкту

Підготовка до розробки програмного забезпечення

Перед тим, як почати розробку вебзастосунку, потрібно встановити середовище розробки. WebStorm є інтегрованим середовищем розробки (IDE) для вебзастосунків, розробленим компанією JetBrains. Основна мета WebStorm - полегшити та прискорити розробку вебзастосунків, забезпечуючи розширені функціональні можливості та зручне середовище розробки.

- Основні особливості WebStorm включають:

Редактор коду: WebStorm надає потужний редактор коду з функціями автодоповнення, підсвічування синтаксису, перевірки помилок та інших корисних інструментів для покращення продуктивності розробника.

Підтримка мови програмування: WebStorm підтримує широкий спектр мов програмування, зокрема JavaScript, HTML, CSS, TypeScript і багато інших. Він надає можливості підсвічування синтаксису, навігації по коду та інструменти рефакторингу для кожної мови.

Інтеграція із засобами розробки: WebStorm інтегрується з популярними інструментами розробки, такими як системи керування версіями Git і Mercurial, фреймворки і бібліотеки JavaScript, такі як React і Angular, інструменти тестування та засоби розгортання.

Налаштування та відладка: WebStorm надає можливості налаштування редактора коду, налаштування форматування, кодування та перевірки стилю. Також доступна потужна система відладки для виявлення та виправлення помилок в коді.

Плагіни та розширення: WebStorm підтримує розширення та плагіни, що дозволяють розширити функціональність IDE.

WebStorm є комерційним продуктом, але також доступна безкоштовна пробна версія для оцінки його можливостей.

Макет користувацького інтерфейсу

Будь-який проєкт завжди складається з двох рівнозначних частин - користувацького інтерфейсу та програмно-апаратної частини. Ці дві складові грають важливу роль в успішному виконанні проєкту, забезпечуючи зручну та ефективну взаємодію з користувачами та надійне функціонування системи.

Користувацький інтерфейс є зовнішнім шаром проєкту, через який користувачі взаємодіють з системою. Його графічний дизайн, макети, елементи управління та інші компоненти роблять інформацію та функціонал доступними для користувача. Якість і зручність користувацького інтерфейсу впливають на задоволення користувачів, їх продуктивність та загальний досвід використання системи. При розробці інтерфейсу необхідно враховувати потреби та вимоги користувачів, їх звички та преференції. Дизайн повинен бути зрозумілим, привабливим та інтуїтивно зрозумілим, а комунікація з користувачем - ефективною та зручною, забезпечуючи необхідний функціонал та доступ до інформації.

Програмно-апаратна частина включає в себе програмне забезпечення та апаратне забезпечення, необхідні для функціонування проєкту. Це можуть бути комп'ютери, сервери, сенсори, пристрої зберігання даних, мережеве обладнання та інші технічні компоненти.

Макет інформаційної системи - це статичне або інтерактивне візуальне відображення користувацького інтерфейсу, що демонструє розташування елементів, їх зовнішній вигляд та можливі функціональні можливості системи. Створення макету є важливим етапом, оскільки він дозволяє визначити загальну структуру та компоненти інтерфейсу перед переходом до фази розробки.

Створюючи макет вебзастосунку, ключове місце займає профіль користувача соціальної мережі. (рис. 4.2)

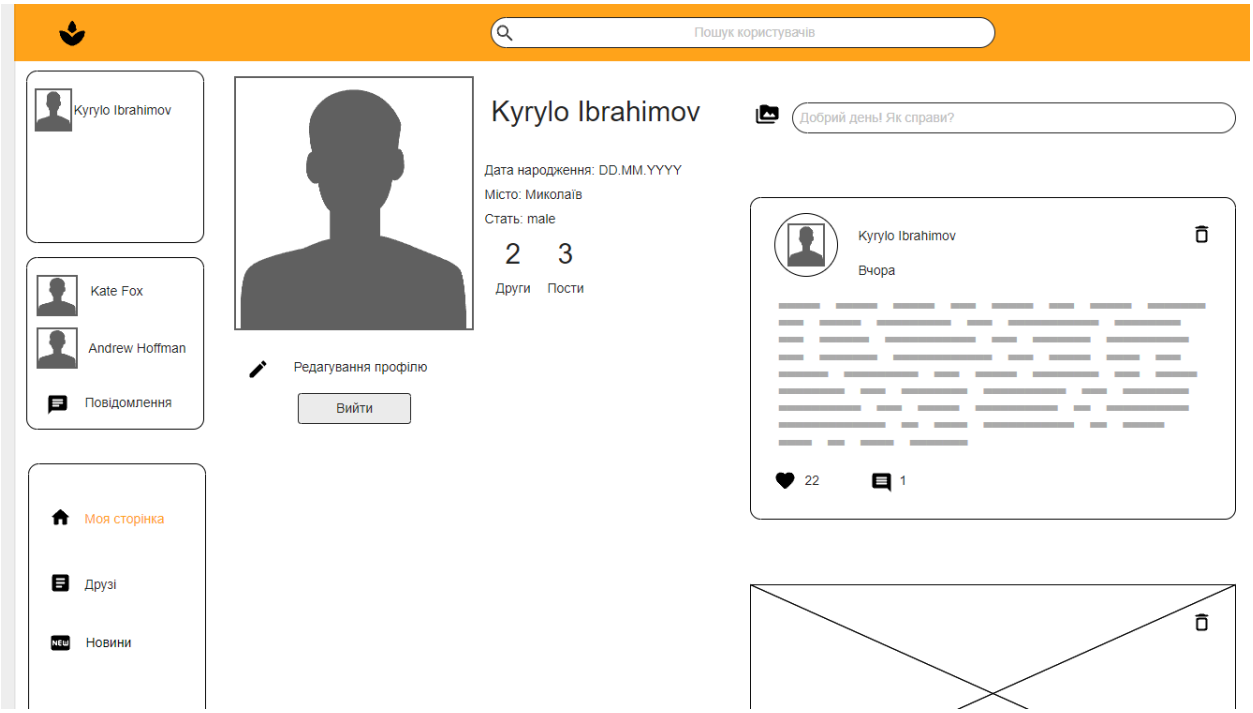

Рисунок 4.2 – Макет профілю користувача

У макеті профілю користувача включені наступні елементи:

- Основна інформація про користувача: ім'я, фотографія, контактні дані тощо.
- Дані про активності та статистика
- Налаштування профілю: можливість змінювати налаштування акаунта, такі як приватність, повідомлення тощо.
- Додаткова інформація: наприклад, професійні дані, освіта, інтереси або особисті хобі.

Стрічка новин є дуже важливою частиною соціальної мережі, це основна сторінка, на котрій користувачі переглядають публікації один одного. (рис. 4. 3)

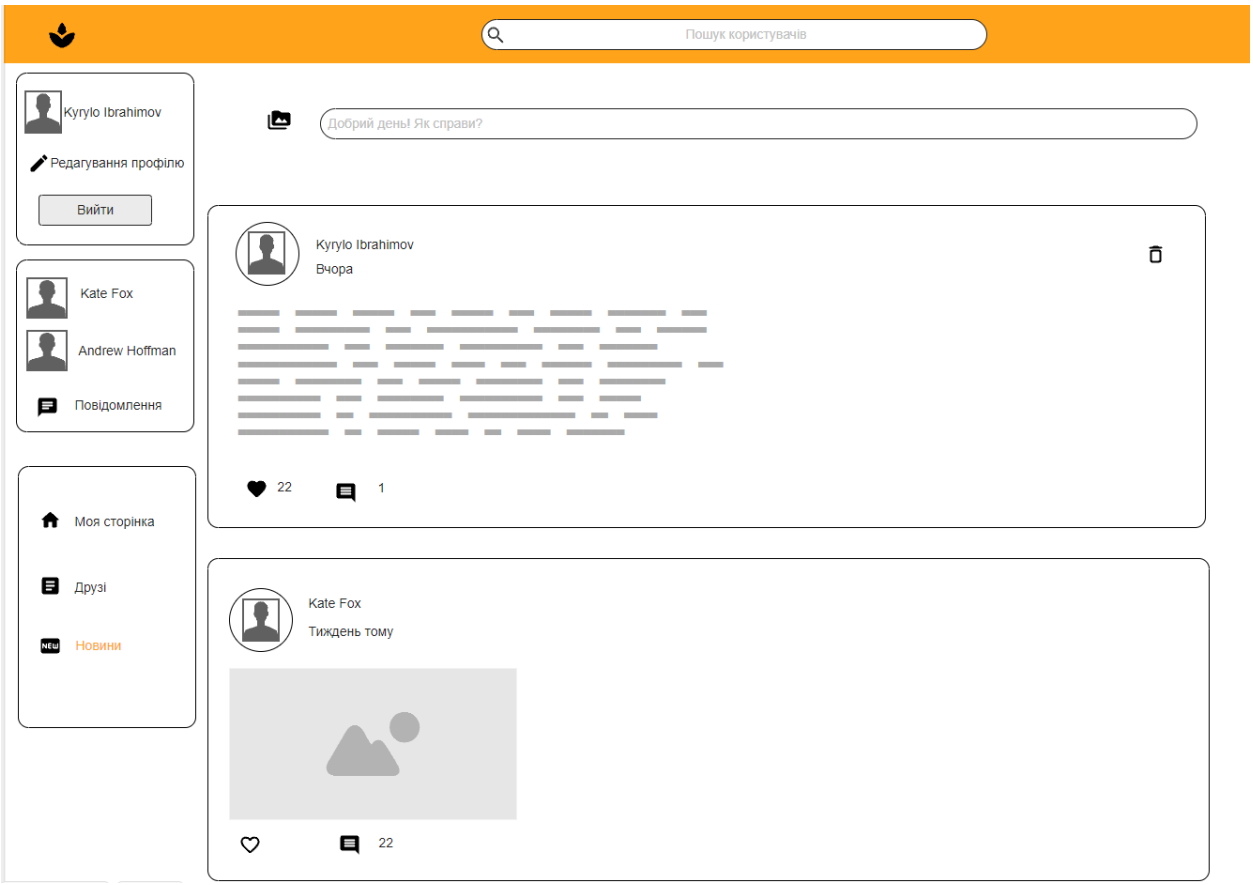

Рисунок 4.3 – Макет стрічки новин соціальної мережі

Стрічка новин – це сторінка, на яку потрапляє користувач після авторизації, також її може побачити незареєстрований користувач, на ній зображена вся інформація про публікації всіх користувачів соціальної мережі. Користувачі можуть видаляти лише свої публікації, але коментувати та ставити реакції можна під публікацією будь – якого іншого користувача.

Звісно у соціальній мережі повинен бути дійсно якісний і зрозумілий спосіб надсилання повідомлень між користувачами, перед створенням чату було зроблено макет (рис. 4.4)

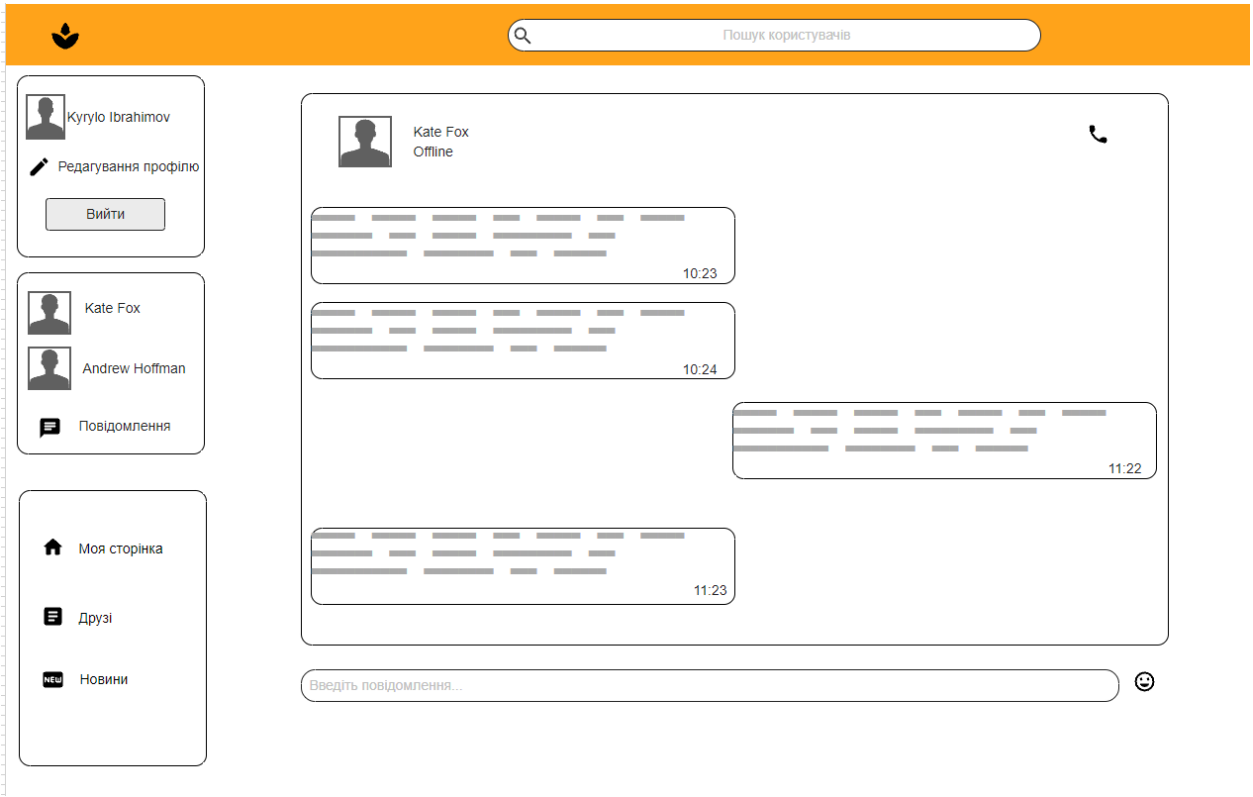

Рисунок 4.4 – Макет інтерфейсу чату

На цьому макеті можна побачити, що з меню чату можна потрапити до будь-якого меню зі всієї соціальної мережі, всі кнопки на зручних місцях, у вікні чату ми можемо побачити статус, чи в мережі друг користувача, бо надсилати повідомлення можна тільки друзям, можемо надсилати повідомлення, використовувати смайлики у діалозі.

4.3 Налаштування середовища роботи та опис структури проєкту

Перед початком програмної реалізації застосунку, потрібно налаштувати середовище розробки. Деякі деталі, такі як встановлення ПЗ для розробки та встановлення залежностей для застосунку у даному розділі упущено, оскільки дані налаштування є досить базовими і не потребують додаткового опису.

Спочатку було встановлено WebStorm та згенеровано Next js проєкт. Проєкт складається із наступних директорій: App, pages, node_modules, cfg. Директорія app розподіляється на такі піддиректорії:

api - містить в собі файли з прив’язками до бекенд серверної частини проєкту, обробку помилок, та файли отримання Cookies при авторизації користувача.

assets – директорія містить в собі файли глобальних стилів всього проєкту, логотипи та потрібні картинки для сайту.

components – директорія, яка містить в собі всі потрібні компоненти, повну обгортку сторінки (layout), header, sidebar, всі необхідні стилі для кожної компоненти, функції завантаження фотографій до публікацій, компоненти взаємодії з публікаціями.

hooks – директорія зі створеними кастомними хуками для взаємодії з елементами проєкту, такими хуками як: useAuth, useProfile, useProfileById, useOutside та ін.

providers – директорія з файлом авторизації котрий відповідає за вхід, отримання даних, та вихід користувача із системи

services – директорія, яка приймає дані з бекенд частини проєкту, отримує їх через axios та створює відповідні запити до отриманної інформації, такої як: авторизація, коментування, проставлення реакцій, відправлення повідомлень, створення постів, додавання у друзі та ін.

types – директорія в якій створені всі інтерфейси до кожної взаємодії

Наступна директорія в проєкті це **node_modules**, вона містить в собі усі залежності проєкту, які були завантажені за допомогою Node Package Manager (npm). Вона містить в собі піддиректорії з файлами та модулями для роботи відповідного пакету.

Наступною директорією в проєкті є **pages**, вона містить в собі tsx файли із кодом, без стилів, для малювання сторінки за посиланням у браузері, до неї відносяться такі файли як: conversations.tsx, friends.tsx, index.tsx, edit.tsx., та інші файли сторінок.

Додатково в проєкті є директорія **cfg** яка містить в собі конфігурації, налаштування пакетів, yarn, eslint та env, в котрій зберігаються адреси серверної частини проєкту, адреси бази даних, та ключі для входу через гугл авторизацію.

Опис backend структури проєкту

Backend серверна частина проєкту містить в собі такі директорії:

Головна директорія **node_modules** містить в собі усі залежності проєкту, які були завантажені за допомогою Node Package Manager (npm). Вона містить в собі піддиректорії з файлами та модулями для роботи відповідного пакету.

В серверній частині проєкту також є директорія **src**, в якій є піддиректорії пов'язані з головними функціями соціальної мережі, такими як: авторизація, коментування, запам’ятовування проставлених лайків під публікаціями, надсилання повідомлень, завантажування файлів, робота з даними користувача, тощо.

Директорія, відповідальна за авторизацію **auth**, поділяється на dto (Data-transfer object), об’єкт, який переносить дані між процесами для зменшення кількості викликів. Також в директорії авторизації є директорія, в якій описуються стратегії авторизації через гугл, її функціонал. В директорії авторизації також є файли auth.controller, auth.decorator, auth.interface, auth.module, auth.service.

У файлі auth.controller описується клас AuthController, який містить метод googleAuth(). Цей метод обробляє POST-запит на шляху 'auth/login/google' та використовує сервіс AuthService для автентифікації користувача за допомогою Google-акаунту. Для перевірки валідності даних з запиту використовується клас ValidationPipe.

У файлі auth.decorator описана функція Auth, яка повертає результат виклику функції UseGuards з параметром AuthGuard('jwt'). Ця функція використовується для захисту маршрутів від несанкціонованого доступу за допомогою JWT-токенів.

Далі наведено код контролеру авторизації AuthController.

```
@Controller('auth')
export class AuthController {
	constructor(private readonly authService: AuthService) {}

	@UsePipes(new ValidationPipe())
	@HttpCode(200)
	@Post('login/google')
	async googleAuth(@Body() dto: GoogleCodeDto) {
		return this.authService.googleLogin(dto)
	}
}
```

У файлі auth.service описано три асинхронні функції: getGoogleToken(), getGoogleProfile() та googleLogin(). Вони використовуються для автентифікації користувача за допомогою Google-акаунту. Метод googleLogin() отримує код авторизації від клієнта та викликає методи getGoogleToken() та getGoogleProfile(), які взаємодіють з API Google для отримання токена доступу та профілю користувача відповідно. Після успішного отримання профілю користувача, дані про нього використовуються для валідації користувача на сервері.

Код сервісу авторизації наведено в додатку **А**.

Наступна директорія, відповідальна за коментвання **comment**, так само розподіляється на comment.service, comment.controller, comment.dto, comment.model, comment.module.

У файлі comment.controller описано клас CommentController, який відповідає за обробку запитів, що стосуються коментарів. Контролер містить дві функції: getCommentByPostId() та createComment().

Функція getCommentByPostId() відповідає на GET-запити на шляху 'comment/by-post/:postId' та повертає коментарі, які стосуються поста з ідентифікатором postId. Параметр postId передається через шлях та проходить валідацію з використанням IdValidationPipe.

Функція createComment() відповідає на POST-запити на шляху 'comment' та створює новий коментар.

Далі наведено код контролера коментування CommentController.

```
@Get('by-post/:postId')
async getCommentByPostId(
	@Param('postId', IdValidationPipe) postId: Types.ObjectId
) {
	return this.commentService.byPostId(postId)
}

@HttpCode(200)
@Post()
@Auth()
async createComment(
	@CurrentUser('_id') _id: Types.ObjectId,
	@Body() dto: CommentDto
) {
	return this.commentService.create(_id, dto)
```

```
}
```

Також не менш важливим є файл comment.service, у цьому файлі описано клас CommentService, який містить дві асинхронні функції: byPostId() та create().

Функція byPostId() використовується для отримання коментарів, які стосуються поста з ідентифікатором postId. Функція виконує запит до бази даних MongoDB за допомогою моделі CommentModel, яка була інжектнута через конструктор з використанням декоратора @InjectModel. Функція повертає масив коментарів, відсортованих за датою створення та з попереднім поповненням поля user, яке містить об'єкт користувача, що створив коментар.

Функція create() використовується для створення нового коментаря. Функція отримує ідентифікатор користувача, який створює коментар, та об'єкт CommentDto, що містить дані про коментар. Функція створює новий об'єкт коментаря за допомогою моделі CommentModel та зберігає його в базу даних.

Далі наведено код сервісу коментування CommentService.

```
        @InjectModel(CommentModel)
        private readonly CommentModel: ModelType<CommentModel>
) {}

async byPostId(postId: Types.ObjectId) {
        return this.CommentModel.find({ post: postId }, '__v')
                .sort({ createdAt: 'desc' })
                .populate('user', 'name avatarPath isVerified')
                .exec()
}

async create(userId: Types.ObjectId, dto: CommentDto) {
        return this.CommentModel.create({
                post: dto.postId,
```

```
        user: userId,
        message: dto.message
    })
```

Наступна директорія, відповідальна за конфігурації проєкту **cfg**, розподіляється на jwt.config, mongo.config. Ці файли об'єднує те, що вони містять функції, які повертають налаштування для різних модулів в NestJS. Файл, який містить функцію getJWTConfig, повертає налаштування для модуля JWT, який використовується для аутентифікації та авторизації користувачів. Файл, який містить функцію getMongoConfig, повертає налаштування для модуля TypegooseModule, який використовується для підключення до MongoDB бази даних та роботи з нею. Обидва файли отримують параметр configService типу ConfigService, що дозволяє їм отримувати значення конфігураційних параметрів з файлу конфігурації.

Далі наведено код файлу mongo.config.

```
export const getMongoConfig = async (
    configService: ConfigService
): Promise<TypegooseModuleOptions> => ({
    uri: configService.get('MONGO_URI')
})
```

Наступна директорія, відповідальна за завантаження файлів **file**, розподіляється на media.service, media.controller, media.interface, media.module.

У файлі media.controller створюється контролер MediaController, який має маршрут "/media". В контролері є конструктор, який отримує залежність mediaService типу MediaService. Контролер містить метод uploadMediaFile, який є POST-маршрутом з маршрутом "/media" і приймає параметри mediaFile та folder. Для обробки файлу використовується бібліотека Multer та інтерсептор FileInterceptor. Метод також має декілька декораторів: HttpCode, який встановлює код статусу відповіді на 200, Auth, який виконує перевірку автентифікації користувача, та UseInterceptors, який дозволяє використовувати інтерсептор для обробки запиту.

Далі наведено код файлу додавання картинок Media Controller.

```
export class MediaController {
	constructor(private readonly mediaService: MediaService) {}
	@HttpCode(200)
	@Post()
	@Auth()
	@UseInterceptors(FileInterceptor('media'))
	async uploadMediaFile(
		@UploadedFile() mediaFile: Express.Multer.File,
		@Query('folder') folder?: string
	) {
		return this.mediaService.saveMedia(mediaFile, folder)
	}
```

У файлі media.service створюється сервіс MediaService, який є Injectable в NestJS. Сервіс містить метод saveMedia, який є асинхронним і приймає параметри mediaFile та folder, який має значення за замовчуванням "default". Метод зберігає файл на сервері за допомогою функції writeFile з буферу файлу та створює об'єкт типу IMediaResponse, який містить властивості url та name, що містять відповідно шлях до файлу та його оригінальне ім'я. Метод повертає цей об'єкт, який містить інформацію про збережений файл.

Далі наведено код медіа сервісу Media Service.

```
export class MediaService {
	async saveMedia(
		mediaFile: Express.Multer.File,
		folder = 'default'
	): Promise<IMediaResponse> {
		const uploadFolder = `${path}/uploads/${folder}`
		await ensureDir(uploadFolder)

		await writeFile(
			`${uploadFolder}/${mediaFile.originalname}`,
			mediaFile.buffer
		)

		return {
			url: `/uploads/${folder}/${mediaFile.originalname}`,
			name: mediaFile.originalname
		}
```

Наступна директорія відповідальна за користувача **user**, в ній описано методи додавання у друзі, оновлення профілю користувача, знаходження інших користувачів, та отримання інформації про користувача.

У файлі user.controller створюється контролер UserController, який має маршрут "/user". Контролер містить декілька методів для взаємодії з користувачами.

Метод getProfile є GET-маршрутом з маршрутом "/profile" та декоратором Auth, який виконує перевірку автентифікації користувача. Метод використовує декоратор CurrentUser для отримання ідентифікатора поточного користувача та передає його до методу getUser сервісу userService, щоб отримати профіль користувача.

Метод getUser є GET-маршрутом з маршрутом "/by-id/:id" та приймає параметр id, який проходить валідацію за допомогою IdValidationPipe. Метод передає цей ідентифікатор до методу getUser сервісу userService, щоб отримати профіль користувача.

Метод findUser є GET-маршрутом з маршрутом "/find/:searchTerm" та приймає параметр searchTerm, який використовується для пошуку користувачів з використанням методу findUser сервісу userService.

Метод updateProfile є PUT-маршрутом з маршрутом "/profile" та декоратором Auth. Метод використовує декоратор CurrentUser для отримання ідентифікатора поточного користувача та отриманий об'єкт UserDto [12] для оновлення профілю користувача за допомогою методу updateProfile сервісу userService.

Метод updateUser є PUT-маршрутом з маршрутом "/:id" та декоратором Auth. Метод приймає параметр id, який проходить валідацію за допомогою IdValidationPipe, та отриманий об'єкт UserDto для оновлення профілю користувача за допомогою методу updateProfile сервісу userService.

Метод toggleFriend є PATCH-маршрутом з маршрутом "/:friendId" та декоратором Auth. Метод використовує декоратор CurrentUser для отримання

ідентифікатора поточного користувача та отриманий ідентифікатор друга для додавання або видалення друга зі списку друзів користувача за допомогою методу toggleFriend сервісу userService.

Код user.controller повністю наведено у додатку Б.

Також важливою частиною роботи з користувачем виконує файл user.service.

У файлі user.service описано реалізацію методів для взаємодії з користувачами.

Метод getUser приймає ідентифікатор користувача та повертає профіль користувача з використанням методу aggregate, який дозволяє виконувати складні запити до бази даних MongoDB. Метод також використовує метод populate для заповнення поля friends об'єктом користувача.

Метод byId приймає ідентифікатор користувача та повертає профіль користувача з бази даних MongoDB.

Метод updateProfile приймає ідентифікатор користувача та об'єкт UserDto для оновлення профілю користувача в базі даних MongoDB.

Метод toggleFriend приймає ідентифікатор поточного користувача та ідентифікатор друга для додавання або видалення друга зі списку друзів користувача в базі даних MongoDB.

Метод findUser приймає рядок для пошуку та повертає список користувачів, що задовольняють критерій пошуку, з бази даних MongoDB.

Код user.service повністю наведено у додатку В.

Також в проєкті є файл .env, це файл конфігурації середовища, який містить змінні середовища для розробки додатку.

Змінна NODE_ENV визначає поточне середовище розробки, у цьому випадку - development.

Змінна MONGO_URI містить URL-адресу бази даних MongoDB, яку використовує застосунок.

Змінні GOOGLE_CLIENT_ID та GOOGLE_SECRET містять ідентифікатор та секретний ключ клієнта Google, які використовуються для автентифікації користувачів через сервіс Google.

Змінна JWT_SECRET містить секретний ключ для генерації та перевірки JWT-токенів, які використовуються для автентифікації та авторизації користувачів.

Далі наведено код файлу конфігурації середовища файл .env.

```
NODE_ENV=development
MONGO_URI=mongodb://127.0.0.1:27017/social-chat
GOOGLE_CLIENT_ID=…..apps.googleusercontent.com
GOOGLE_SECRET=….**-P0TTqsQ2gUUzclMA-
JWT_SECRET=3wef'2ED
```

Найголовніший файл в проєкті є main.ts цей файл, який містить функцію bootstrap, яка створює та запускає застосунок.

Функція використовує фреймворк NestJS та створює екземпляр класу AppModule, який містить налаштування та провайдери для всього додатку.

Далі, функція встановлює глобальний префікс для всіх маршрутів додатку за допомогою методу setGlobalPrefix.

Далі, функція встановлює глобальну обробку помилок валідації за допомогою пайпу ValidationPipe та параметру forbidUnknownValues.

Далі, функція запускає застосунок на порту 4200 за допомогою методу listen.

Далі, функція встановлює middleware для роботи з сесіями та автентифікацією користувачів за допомогою PassportJS.

4.4 Програма для тестування запитів та опис користувацького інтерфейсу

Усі запити було протестовано за допомогою програми Insomnia [13] (рис. 4.1) - це програма для тестування API, яка дозволяє легко та швидко надсилати запити до вебсервера та отримувати відповіді в різних форматах (JSON, XML, HTML тощо).

Програма має інтуїтивно зрозумілий інтерфейс, де ви можете створювати, зберігати та організовувати ваші запити до API. Ви можете налаштовувати різні параметри запитів, такі як заголовки, параметри запиту, тіло запиту та інші.

У загальному, Insomnia є потужним інструментом для розробників, який дозволяє ефективно працювати з API та зберігати ваші запити для майбутнього використання.

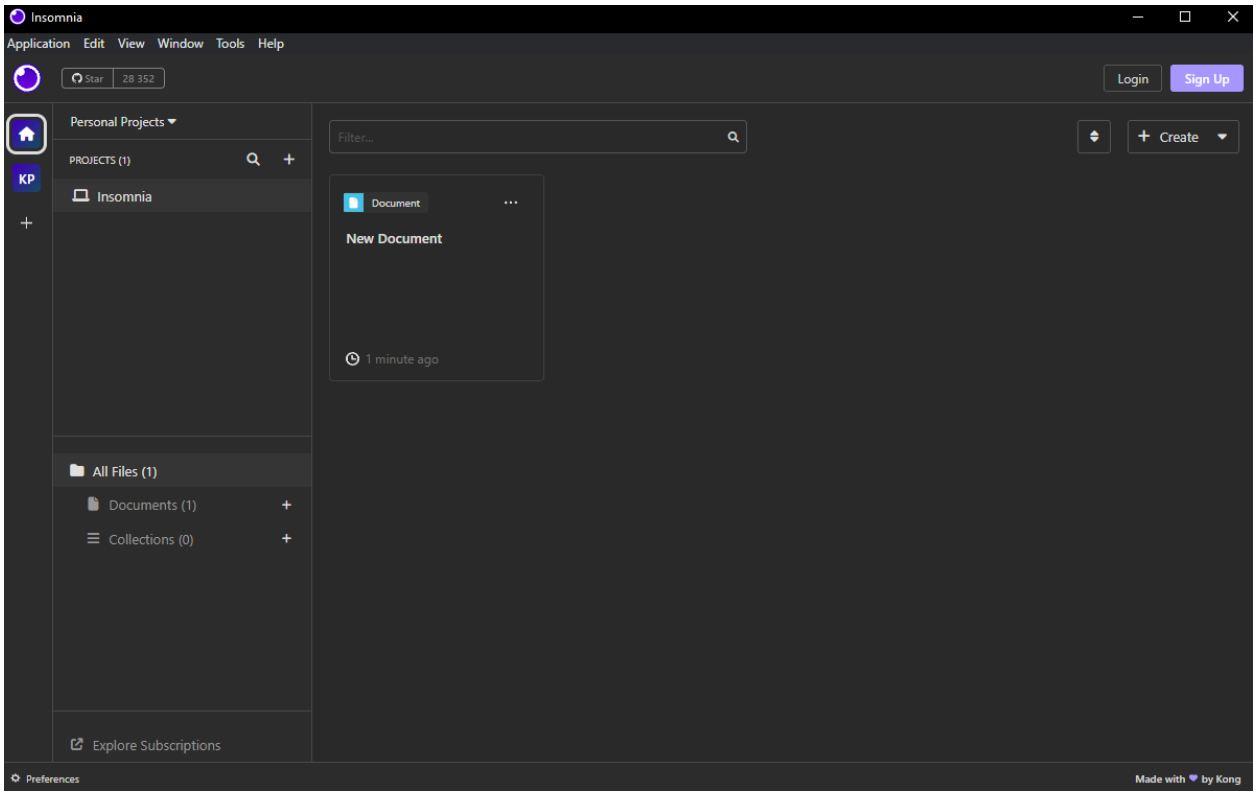

Рисунок 4.1 – Інтерфейс програми Insomnia

Користувацький інтерфейс є однією з найважливіших частин вебзастосунку. Інтерфейс повинен бути інтуїтивно зрозумілим для користувача, лаконічним та, що не мало важливо привабливим. Далі описується інтерфейс основних сторінок вебзастосунку.

Головна сторінка (стрічка новин)

Неавторизований користувач може тільки передивлятись інші публікації, шукати користувачів, при спробі поставити реакцію під публікацією, чи прокоментувати, чи додати свою публікацію неавторизованому користувачу сайт запропонує авторизуватись через гугл. (рис. 4. 2)

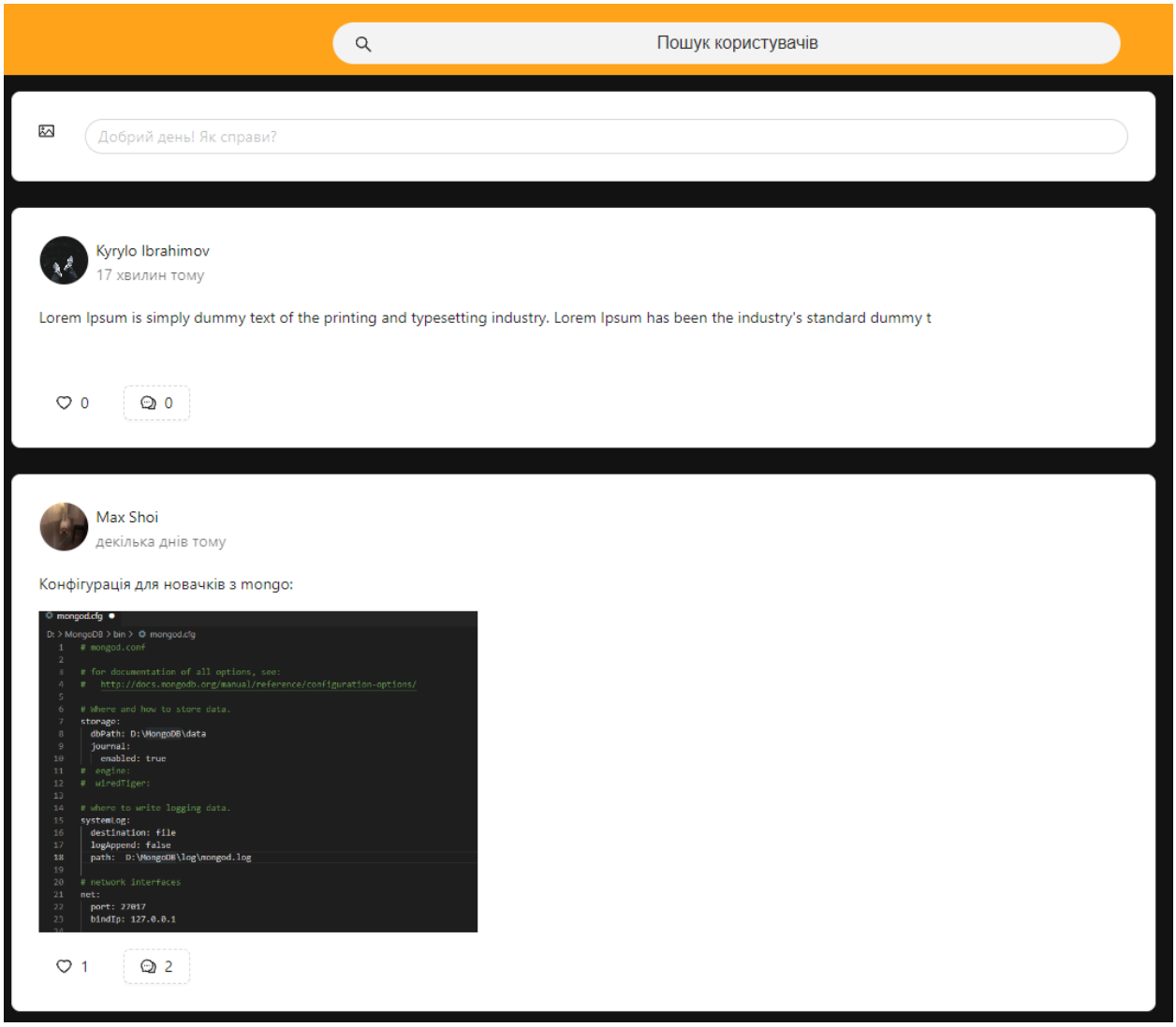

Рисунок 4.2 – Інтерфейс неавторизованого користувача

Після пройденої авторизації за допомогою гугл акаунта, користувач отримає відповідне повідомлення про успіх, та потрапить на головну стрічку новин, на якій можна переглянути свої публікації та публікації інших користувачів. Додатково користувач може видалити лише свої публікації. Також справа під фотографією користувача, яка підтягується з гугл аккаунту як і всі дані (про стать, місто тощо) користувач може перейти до редагування свого профілю, чи вийти з акаунта соціальної мережі. (рис. 4.3)

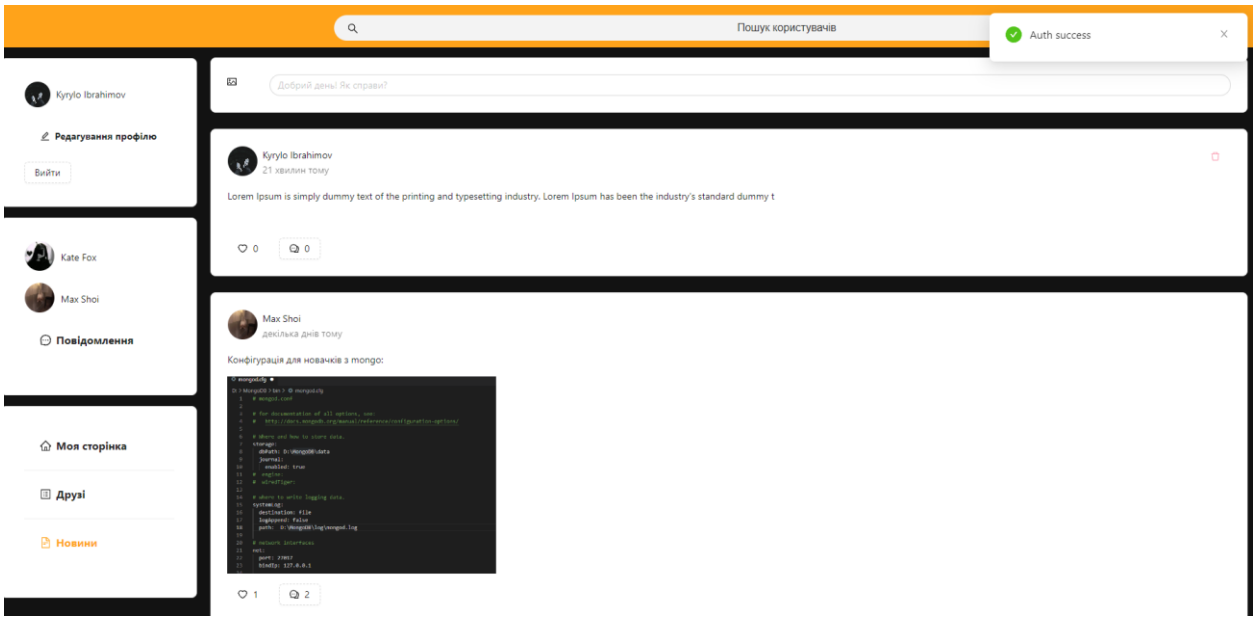

Рисунок 4.3 - Інтерфейс стрічки новин

Додатково під особистими налаштуваннями користувач може побачити 5 своїх друзів, з котрими він нещодавно вів бесіду, під друзями є посилання на усі повідомлення, та посилання на сторінку користувача, сторінку пошуку друзів, та стрічку новин, діюча сторінка виділена помаранчевим кольором.

Сторінка редагування профілю, та особиста сторінка користувача

Після переходу на сторінку редагування користувач бачить перед собою таку форму (рис. 4.4)

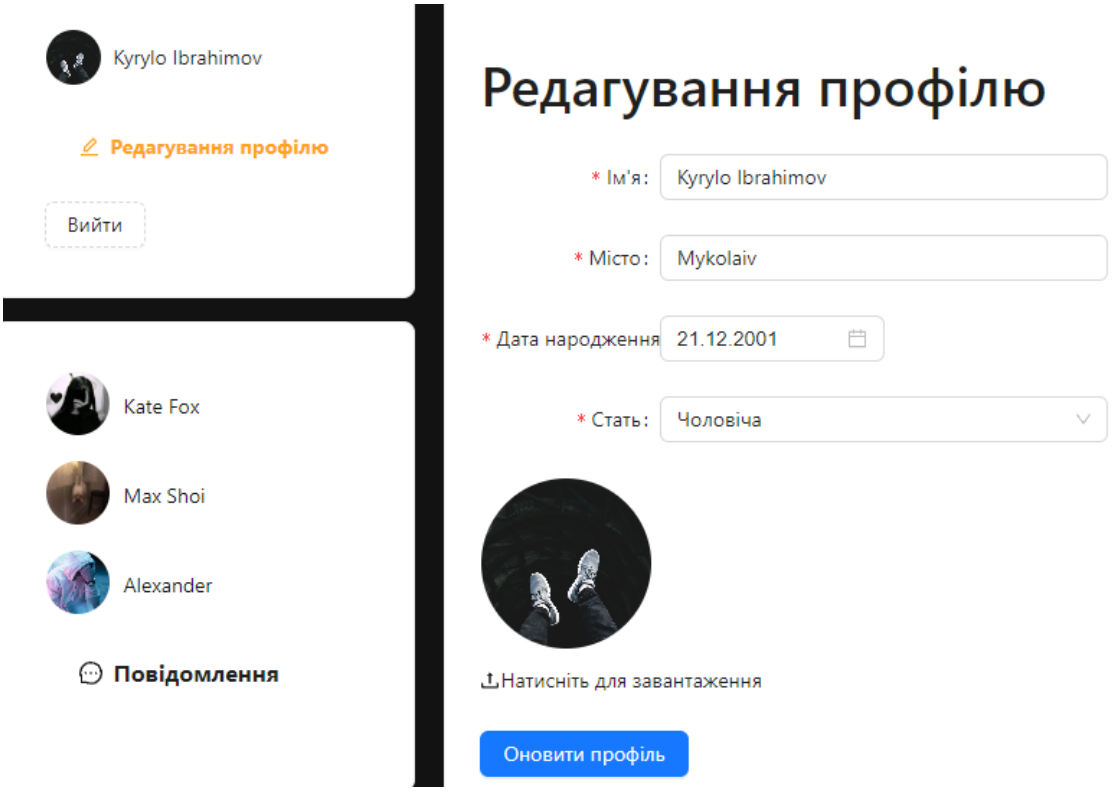

Рисунок 4.4 – Форма редагування профілю

У цій формі користувач може швидко та зручно замінити необхідні про себе, про свій профіль дані, провести такі зміни, як: ім’я, місто, дата народження та стать, також можна натиснути на відповідну кнопку та завантажити будь-яку картинку для виставлення її як головна на своєму профілі.

Особиста сторінка користувача містить всю інформацію, яку можна відредагувати, з неї можна переглянути всі свої публікації та створити нову. (рис. 4.5)

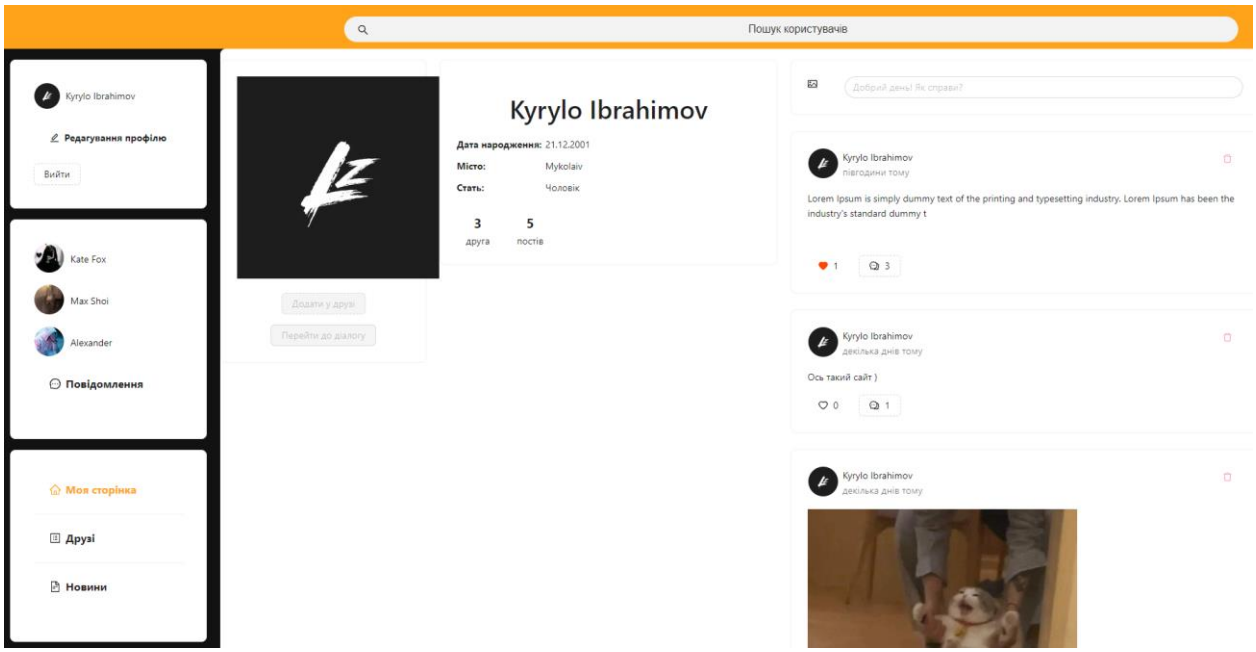

Рисунок 4.5 – Особиста сторінка користувача

Особисті сторінки інших користувачів дещо інші, через них можна написати в особисті повідомлення іншому користувачу, додати чи видалити іншого користувача до списку друзів. (рис. 4.6)

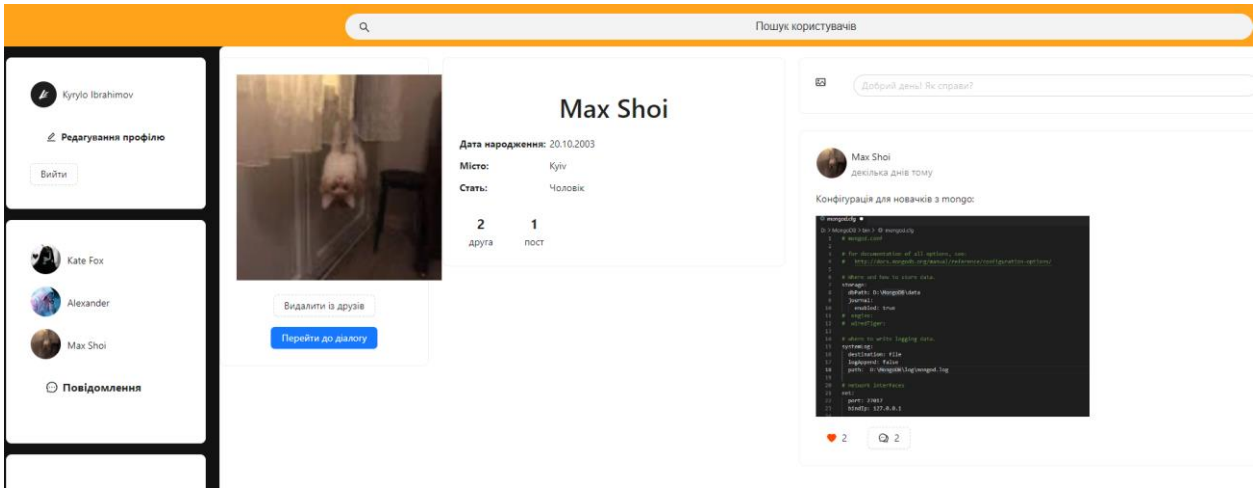

Рисунок 4.6 – Особиста сторінка іншого користувача

Функція пошуку інших користувачів є дуже важливою, тому вона і знаходиться на видимому місці весь час, вона за допомогою імені користувача знаходить усіх відповідних користувачів. (рис. 4.7)

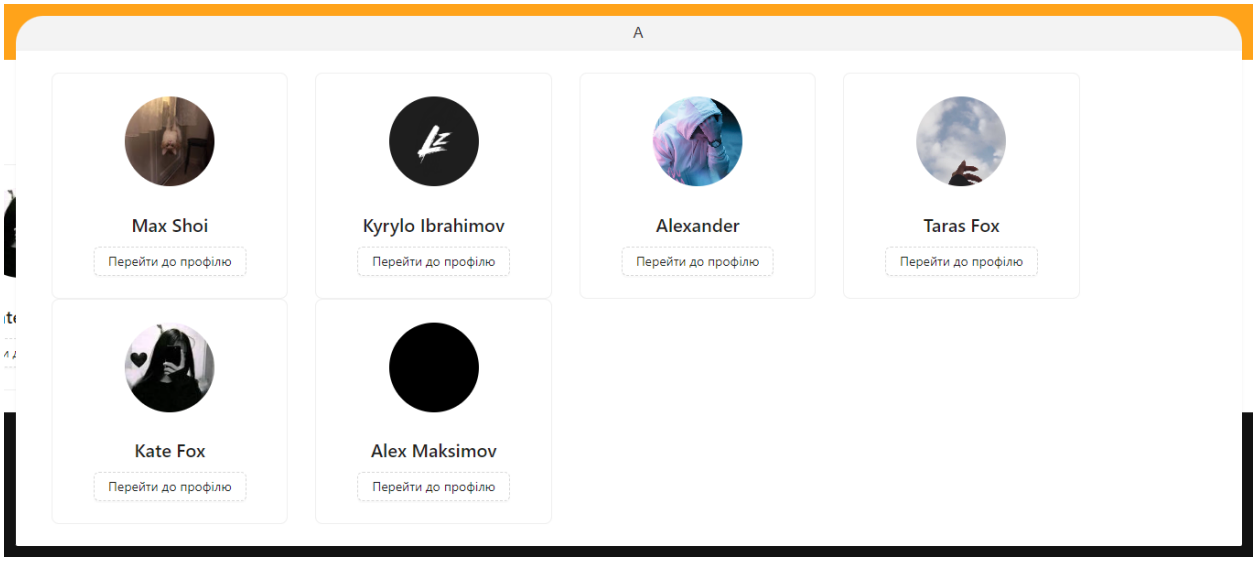

Рисунок 4.7 – Робота поля пошуку

Усі додані користувачі до списку друзів можуть бути знайдені у відповідальній за друзів сторінці соціальної мережі. (рис. 4.8)

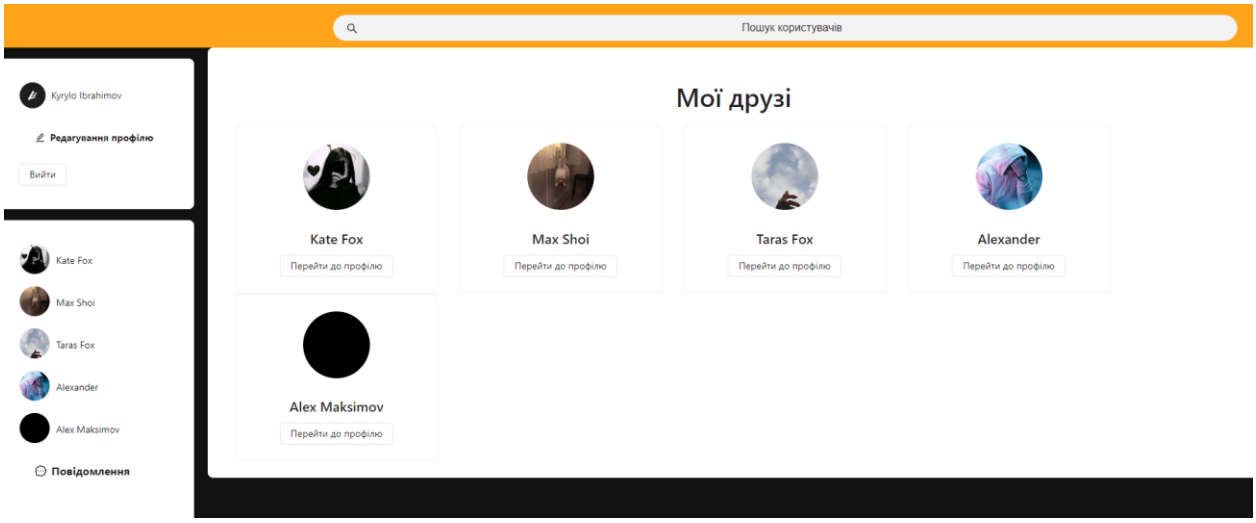

Рисунок 4.8 – Сторінка «Мої друзі»

Висновки до розділу 4

Протягом четвертого розділу було розглянуто важливі аспекти вибору технологій для розробки вебзастосунків. В ході виконання четвертого розділу був описаний етап програмної реалізації системи.

Врахування складності програмного забезпечення, наявності ресурсів, готових компонентів, технічної документації, характеристик якості ПЗ, витрат на технології, ліцензійної політики та вимог безпеки є ключовими факторами при прийнятті рішення щодо вибору технологій.

У даному проєкті для розробки вебзастосунків було обрано React Redux як технологію розробки програмного забезпечення і MongoDB як базу даних. React є потужною бібліотекою JavaScript, яка дозволяє створювати повторно використовувані компоненти інтерфейсу. Redux є передбачуваним контейнером стану, що допомагає керувати станом додатку організовано.

Також використано WebStorm як інтегроване середовище розробки для написання програмного коду і MongoDB Compass для створення бази даних. WebStorm надає ряд корисних функцій для покращення продуктивності розробника, таких як автодоповнення, підсвічування синтаксису і відлагодження коду.

Архітектура MVC була використана для розділення застосунку на три основні компоненти: модель, вид і контролер. Це дозволяє розподілити відповідальності між компонентами і забезпечує полегшення розробки, тестування і підтримки програмного забезпечення.

Протягом даного розділу було детально описано користувацький інтерфейс вебзастосунку. Було зазначено, що інтерфейс повинен бути інтуїтивно зрозумілим, лаконічним та привабливим для користувача. Були описані основні сторінки вебзастосунку, зокрема головна сторінка зі стрічкою новин, сторінка редагування профілю, особиста сторінка користувача, особисті сторінки інших користувачів, функція пошуку користувачів.

ВИСНОВКИ

В рамках проєкту було виконано аналіз предметної області, вимог до системи та існуючих аналогів. У розділі 2 були розроблені сценарії використання системи, діаграми класів, діаграма станів та діаграма діяльності. Це дозволило краще зрозуміти функції та взаємодію компонентів системи.

Розділ 3 був присвячений вибору технологій для розробки вебзастосунків. Було обрано React Redux як технологію розробки програмного забезпечення і MongoDB як базу даних. Архітектура MVC була використана для розділення системи на компоненти.

У розділі 4 було проведено програмну реалізацію системи, включаючи налаштування середовища розробки, розробку серверної та клієнтської частин, а також опис користувацького інтерфейсу. Результатом був реалізований вебзастосунок онлайн комунікації користувачів, який відповідає вимогам та моделям, розробленим у попередніх розділах.

У загальному, проєкт був успішно реалізований, враховуючи аналіз предметної області, вимоги до системи та вибір технологій. Розроблений вебзастосунок надає зручні та інтуїтивно зрозумілі можливості для користувачів у сфері онлайн комунікації.

В процесі виконання роботи були успішно виконані всі поставлені завдання, зокрема аналіз існуючих соціальних мереж, вивчення особливостей проєктування та створення вебзастосунків, проектування авторизації, систем обміну повідомленнями та створення постів, розробка вебзастосунку та проведення його тестування.

В результаті роботи було створено функціональний та зручний вебзастосунок, який забезпечує користувачам можливість обміну повідомленнями, створення постів та ефективну комунікацію. Розроблене програмне забезпечення відповідає вимогам, було належним чином та було належно протестоване.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. HTML basics - Learn web development | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 20.03.2023).

2. JavaScript - Клієнтською мова програмування, що робить сторінки сайту інтерактивними. URL: https://astwellsoft.com/uk/blog/tehnology/javascript.html (дата звернення: 28.03.2023)

3. CSS Tutorial. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/css/default.asp(last accessed: 04.02.2023).

4. CSS: Cascading Style Sheets | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (last accessed: 24.04.2023)

5. Icons. Free Open Source Icons. URL: https://www.toools.design/free-open-source-icon-libraries (last accessed: 24.04.2023)

6. Ant Design: вебсайт. URL: https://ant.design/

7. StarUML: вебсайт. URL: https://staruml.io/ (дата звернення 01.05.2023)

8. Система управління базами даних URL: https://hostiq.ua/wiki/database/ (дата звернення: 02.05.2023)

9. UML Diagram Types Guide. Learn About All types of UML Diagrams with Examples: вебсайт. URL: https://creately.com/tour/ (last accessed: 02.04.2023);

10. How React and Redux brought back MVC. Website URL: https://rangle.io/blog/how-react-and-redux-brought-back-mvc-and-everyone-loved-it (last accessed: 18.05.2023)

11. Best JavaScript Frameworks to Use in 2023. URL: https://hackr.io/blog/best-javascript-frameworks (last accessed: 17.05.2023)

12. Навіщо використовують DTO. Приклади в Java-застосунках. URL: https://dou.ua/forums/topic/34411/ (дата звернення: 22.05.2023)

13. The Collaborative API Development Platform – Insomnia. URL: https://insomnia.rest/ (last accessed: 12.03.2023)

14. Laurie P., Laurie B. Apache: the definitive guide (3rd edition). O'Reilly Media, Inc.: 2002. 536 p.

15. Google Cloud documentation. URL: https://cloud.google.com/docs

16. The WebSocket API (WebSockets). URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (last accessed 25.05.2023)

17. MongoDB: The Developer Data Platform | MongoDB. URL: https://www.mongodb.com/

ДОДАТОК А

Код відповідального за авторизацію користувачів

```
@Injectable()
export class AuthService {
	constructor(
		@InjectModel(UserModel) private readonly UserModel: ModelType<UserModel>,
		private readonly jwtService: JwtService,
		private readonly httpService: HttpService,
		private readonly configService: ConfigService
	) {}

	async validateUser(details: IResGoogleUser) {
		let user = await this.UserModel.findOne({ email: details.email })
		if (!user) user = await this.UserModel.create(details)
		return {
			user,
			accessToken: await this.issueAccessToken(String(user._id))
		}
	}

	async issueAccessToken(userId: string) {
		const data = { _id: userId }

		return await this.jwtService.signAsync(data, {
			expiresIn: '31d'
		})
```

```
    }

    async getGoogleToken(code: string) {
        return firstValueFrom(
            this.httpService
                .post<{ access_token: string }>(
                    'https://www.googleapis.com/oauth2/v4/token',
                    {
                        code,
                        client_id:
this.configService.get('GOOGLE_CLIENT_ID'),
                        client_secret:
this.configService.get('GOOGLE_SECRET'),
                        redirect_uri:    'http://localhost:3000/google-
auth',
                        grant_type: 'authorization_code'
                    }
                )
                .pipe(map((res) => res.data))
        )
    }
    async getGoogleProfile(accessToken: string) {
        console.log(accessToken)
        return firstValueFrom(
            this.httpService

    .get<IGoogleProfile>('https://www.googleapis.com/oauth2/v3/userinfo', {
                    headers: {
                        Authorization: `Bearer ${accessToken}`
```

```
                    }
                })
                .pipe(map((res) => res.data))
        )
    }

    async googleLogin({ code }: GoogleCodeDto) {
        if (!code) {
            throw new BadRequestException('Google code not found')
        }
        try {
            const { access_token } = await this.getGoogleToken(code)
            const profile = await this.getGoogleProfile(access_token)
            console.log(profile)
            return this.validateUser({
                email: profile.email,
                name: profile.name,
                avatarPath: profile.picture
            })
        } catch (e) {
            throw new HttpException(e.response.data, e.response.status)
        }
    }
```

ДОДАТОК Б

Код відповідальний за реалізацію методів взаємодії між користувачами, та редагування профілю

```
@Controller('user')
export class UserController {
	constructor(private readonly userService: UserService) {}

	@Get('profile')
	@Auth()
	async getProfile(@CurrentUser('_id') _id: Types.ObjectId) {
		return this.userService.getUser(_id)
	}

	@Get('by-id/:id')
	async getUser(@Param('id', IdValidationPipe) id: string) {
		return this.userService.getUser(new Types.ObjectId(id))
	}

	@Get('find/:searchTerm')
	async findUser(@Param('searchTerm') searchTerm: string) {
		return this.userService.findUser(searchTerm)
	}

	@HttpCode(200)
	@Put('profile')
	@Auth()
	async updateProfile(
```

```
        @CurrentUser('_id') _id: Types.ObjectId,
        @Body() dto: UserDto
) {
        return this.userService.updateProfile(_id, dto)
}

@HttpCode(200)
@Put(':id')
@Auth()
async updateUser(
        @Param('id', IdValidationPipe) id: Types.ObjectId,
        @Body() dto: UserDto
) {
        return this.userService.updateProfile(id, dto)
}

@HttpCode(200)
@Patch(':friendId')
@Auth()
async toggleFriend(
        @CurrentUser('_id') currentUserId: Types.ObjectId,
        @Param('friendId', IdValidationPipe) friendId: Types.ObjectId
) {
        return this.userService.toggleFriend(currentUserId, friendId)
}
```

ДОДАТОК В

Код відповідальний за реалізацію методів для взаємодії з користувачами

```
@Injectable()
export class UserService {
	constructor(
		@InjectModel(UserModel) private readonly UserModel: ModelType<UserModel>
	) {}

	async getUser(_id: Types.ObjectId) {
		const user = await this.UserModel.aggregate()
			.match({ _id })
			.lookup({
				from: 'Post',
				foreignField: 'user',
				localField: '_id',
				as: 'posts'
			})
			.addFields({
				postsCount: {
					$size: '$posts'
				}
			})
			.project({ __v: 0, posts: 0 })
		await this.UserModel.populate(user, {
			path: 'friends',
			select: 'name avatarPath'
```

```
        })

        return user[0]
    }

    async byId(_id: Types.ObjectId) {
        const user = await this.UserModel.findById(_id, '-__v')
        if (!user) throw new UnauthorizedException('User not found')

        return user
    }
    async updateProfile(_id: Types.ObjectId, dto: UserDto) {
        const user = await this.byId(_id)

        user.name = dto.name
        user.city = dto.city
        user.birthDate = dto.birthDate
        user.gender = dto.gender
        user.avatarPath = dto.avatarPath

        return await user.save()
    }

    async toggleFriend(currentUserId: Types.ObjectId, friendId: Types.ObjectId)
{
        const currentUser = await this.byId(currentUserId)
        const friend = await this.byId(friendId)
```

```
        if (currentUser.friends.includes(friendId)) {
                currentUser.friends = currentUser.friends.filter(
                        (id) => String(id) !== String(friendId)
                )
                friend.friends = friend.friends.filter(
                        (id) => String(id) !== String(currentUserId)
                )
        } else {
                currentUser.friends = [...currentUser.friends, friendId]
                friend.friends = [...friend.friends, currentUserId]
        }
        await currentUser.save()
        await friend.save()

        return true
}

async findUser(searchTerm: string) {
        return this.UserModel.find({
                $or: [
                        {
                                name: new RegExp(searchTerm, 'i')
                        }
        })
                .select('-__v')
                .sort({ createdAt: 'desc' })
                .populate('name avatarPath isVerified')
```

ДОДАТОК Г

Код відповідальний за надсилання, видалення та отримування повідомлень

```
const SERVER_URL = 'http://localhost:80'

export const useChat = (conversationId:string) => {

   const [conversation, setConversation] = useState<IConversation>({} as
IConversation)

   const [socket, setSocket] = useState<Socket<DefaultEventsMap,
DefaultEventsMap> | null>(null)

   useEffect(() => {

      if(!conversationId) {


         const newSocket = io(SERVER_URL, {

            query: {conversationId},

         })

         setSocket(newSocket)

         return () => {

            newSocket.close()

         }

      }

   }, [conversationId, setSocket])

   useEffect(() => {

      if(!socket || !conversationId) return

      socket.emit('message:get', {conversationId})

      socket.on('conversation', conversation => {

         setConversation(conversation)

      })

      return () => {

         socket.disconnect()

      }
```

```
}, [conversationId, socket])


const sendMessage = (body: IMessageFields) => {
    socket?.emit('message:add', body)
}
const removeMessage = (body: IDeleteMessageFields) => {
    socket?.emit('message:remove', body)
}
return {conversation, sendMessage, removeMessage}
                                                    }
```