Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ВЕБ-СЕРВІСУ ДЛЯ КОМУНІКАЦІЙ ЗА ІНТЕРЕСАМИ

DEVELOPMENT OF A WEB SERVICE FOR INTEREST COMMUNICATIONS

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗ-42
Кисіль Олег Володимирович
_____
(підпис)

Керівник:
д.т.н., професор
Гордєєв Олександр Олександрович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«__8__» ____06____ 2024 р.
к.т.н., доцент
Гарант освітньої програми:
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А. Ліщенко
«2» 02 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Кисілю Олегу Володимировичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: *розробка веб-сервісу для комунікацій за інтересами*

Керівник роботи: *д.т.н., професор Гордєєв Олександр Олександрович*
затверджені наказом закладу вищої освіти від «*30» грудня 2023 р.* № *477/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«5» червня 2024 р.*
3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.

5. Перелік графічного матеріалу: *16 рисунків*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Гордєєв О. О.*		
Специфікації вимог до розробленої системи	*Гордєєв О. О.*		
Розробка об'єкта проєктування	*Гордєєв О. О.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту		47 %	
Академічна доброчесність	*Яцук А. А.*		

7. Дата видачі завдання *«2» лютого 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*07.02.2024 р.*	
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*27.02.2024 р.*	
3	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*12.03.2024 р.*	
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*17.04.2024 р.*	
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*18.05.2024 р.*	
6	*Тестування та налагодження об'єкта проектування*	*01.06.2024 р.*	
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*05.06.2024 р.*	

Здобувач вищої освіти _____ *Кисіль О. В*
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ *Гордєєв О. О*
(підпис) (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

Рецензія

на кваліфікаційну роботу бакалавра

Здобувач вищої освіти: Кисіль Олег Володимирович

Тема: «Розробка веб-сервісу для комунікацій за інтересами»

Коротка характеристика кваліфікаційної роботи: Кваліфікаційна робота бакалавра присвячена розробці веб-сервісу для спілкування. Сервіс дозволяє користувачам спілкуватись, шукати друзів, підписуватись на них, писати пости та листуватись в спільному чаті. У роботі використані сучасні технології такі як React, TypeScript, JavaScript, Redux та CSS

Самостійні розробки і пропозиції автора: У роботі детально описаний процес розробки сервісу для спілкування. Розроблений інтерфейс користувача, який забезпечує інтуїтивно зрозуміле керування сторінкою, а також реалізований функціонал для швидкого та зручного використання сайту. За допомогою цього

Практичне значення роботи: Сервіс дозволяє створювати та підтримувати соціальні зв'язки через різні функції, такі як обмін повідомленнями, написання постів, пошук та підписка на друзів. Ця робота може бути корисною як для кінцевих користувачів, так і для розробників, які шукають приклад успішної реалізації веб-додатків із використанням сучасних технологій.

Недоліки: Одним із недоліків роботи можна вважати бекенд частину, так як вона є не повноцінною і не може мати надто великий функціонал

Загальний висновок: Робота виконана на високому рівні, має логічну структуру, відповідає встановленим вимогам, може бути представлена до захисту на екзаменаційній комісії та заслуговує високої оцінки.

Рецензент: Ткачук Анатолій Анатолійович

Рецензент кваліфікаційної роботи _____ Ткачук
(підпис)

«7» 06 2024р.

Міністерство освіти і науки України
Луцький національний технічний університет

Факультет *комп'ютерних та інформаційних технологій*
(назва)

Кафедра *інженерії програмного забезпечення*
(назва)

ВІДГУК
КЕРІВНИКА НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач вищої освіти: *Кисіль Олег Володимирович*
(ПІБ)

група *ІПЗ-42*
(шифр)

Тема кваліфікаційної роботи:

Розробка веб-сервісу для комунікацій за інтересами

Development of a Web Service for Interest Communications
(повна назва згідно наказу)

Керівник: *д.т.н професор Гордєєв О.О.*
(науковий ступінь, вчене звання, посада, ПІБ)

Актуальність теми:

У сучасному світі соціальні мережі стали невід'ємною частиною повсякденного спілкування, забезпечуючи зручні платформи для обміну інформацією та об'єднання людей за інтересами. Проте, незважаючи на розвинені можливості численних соціальних платформ, залишається потреба у комунікації за конкретними інтересами. Існуючи рішення не покривають необхідні налаштування для підтримки у спілкуванні відповідних спільнот з використанням можливостей комп'ютерних мереж та веб-технологій. Це створює обґрунтовану необхідність у розробки таких рішень.

Об'єкт дослідження:

Соціальна мережа, створена за допомогою react і TypeScript Процеси автоматизації різноманітних комунікацій.

Характеристика теоретичного рівня роботи, наявності самостійних розробок і практичної значущості роботи:

Кваліфікаційна робота містить усі необхідна розділи, обсяг та на змістовне наповнення. Дипломна робота відповідає усім вимогам, які висуваються для таких робот бакалаврського рівня.

Зауваження та недоліки:

Результати проектування були представлені не в повному обсязі.

Загальний висновок:

Кваліфікаційна робота відповідає вимогам, студент був допущений до захисту. Рекомендована оцінка «добре».

Керівник _____ (*Гордєєв Олександр Олександрович*)
(підпис) (ПІБ)

« 7» червня 2024 р.

АНОТАЦІЯ

Кисіль О. В. Розробка веб-сервісу для комунікацій за інтересами. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2024.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків і пропозицій, списку використаних джерел та додатків.

У роботі досліджено аналіз предметної області соціальних мереж, їх функціональність та різновиди, вплив на суспільство, вибір засобів проектування. Описано специфікацію вимог до розроблюваної системи, включаючи критерії дизайну та використані технології (React, TypeScript, Redux, CSS, Node.js). Проведено детальний опис архітектури програмного додатку, структури проекту, програмної реалізації, тестування та налагодження системи.

Ключові слова: соцмережа, JavaScript, React, Node, Typescript, HTML, CSS.

ABSTRACT

Kisil O. V. Development of a web service for communications based on interests. Manuscript.

Bachelor's qualifying thesis of the OP "Software Engineering". Lutsk National Technical University. Lutsk, 2024.

The bachelor's qualification work consists of an introduction, three sections, conclusions and proposals, a list of used sources and appendices.

The work examines the analysis of the subject area of social networks, their functionality and varieties, their impact on society, and the choice of design tools. The specification of the requirements for the developed system is described, including the design criteria and the technologies used (React, TypeScript, Redux, CSS, Node.js). A detailed description of the architecture of the software application, the structure of the project, software implementation, testing and debugging of the system was carried out.

Keywords: social network, JavaScript, React, Node, Typescript, HTML, CSS.

ЗМІСТ

ВСТУП

У сучасному світі соціальні мережі стали невід'ємною частиною повсякденного життя, забезпечуючи зручні платформи для спілкування, обміну інформацією та об'єднання людей за інтересами. Проте, незважаючи на розвиток численних соціальних платформ, залишається проблема створення спеціалізованих веб-сервісів для комунікацій за конкретними інтересами. Багато з існуючих платформ не пропонують достатньо гнучкості та налаштувань для підтримки специфічних спільнот, що створює потребу в нових рішеннях.

Світові тенденції вирішення цієї проблеми включають розробку спеціалізованих веб-сервісів, які дозволяють користувачам створювати та налаштовувати власні спільноти за інтересами, забезпечуючи при цьому високу інтерактивність та зручність використання. Такі сервіси часто використовують сучасні технології, такі як React, TypeScript, Node.js та інші, для створення швидких, масштабованих та зручних у використанні додатків.

Актуальність даної роботи полягає в необхідності розробки нового веб-сервісу, який би задовольняв потреби користувачів у створенні та підтримці спільнот за інтересами, надаючи інтуїтивно зрозумілий інтерфейс та багатий функціонал. Це дозволить покращити якість комунікацій та сприятиме розвитку нових спільнот.

Метою роботи є розробка веб-сервісу для комунікацій за інтересами, що базується на сучасних веб-технологіях та надає користувачам широкий спектр можливостей для взаємодії та обміну інформацією. Галузь застосування даного сервісу охоплює будь-які спільноти за інтересами, від професійних груп до хобі-клубів та соціальних організацій.

Ця робота має тісний взаємозв'язок з іншими дослідженнями у сфері соціальних мереж та веб-розробки, оскільки вона базується на використанні найновіших технологій та підходів, що були успішно застосовані в інших проектах.

Завдання кваліфікаційної роботи включає в себе наступні пункти:

1) розробка функціоналу для створення та управління профілями користувачів, включаючи можливість додавання друзів, редагування особистої інформації та управління налаштуваннями приватності;

2) реалізація системи аутентифікації та авторизації користувачів з використанням сучасних методів безпеки, щоб забезпечити захист конфіденційної інформації;

3) створення системи обміну повідомленнями та коментарів між користувачами, що включає в себе можливість відправлення текстових повідомлень, фотографій та відео;

4) оптимізація додатку для забезпечення масштабованості та ефективності його роботи при великій кількості користувачів та обсязі даних.

Об’єктом кваліфікаційної роботи є соціальна мережа, створена за допомогою React і TypeScript. Проєкт включає сторінки профілю, користувачів, діалогів та логіну. Для роутингу використовується react router, для керування станом - redux та react-redux, для форм – Formik і redux-form, асинхронний код виконується за допомогою redux-thunk. Весь проект типізований за допомогою TypeScript.

Предметом кваліфікаційної роботи є вивчення та розробка функціоналу соціальної мережі, а також інтеграція різних технологій та інструментів для створення ефективного та масштабованого веб-додатку.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Соціальні мережі є одним з найпотужніших інструментів сучасної комунікації, надаючи можливість людям з різних куточків світу спілкуватися, обмінюватися інформацією та об’єднуватися за інтересами. Вони являють собою платформи, де користувачі можуть створювати особисті профілі, додавати друзів, ділитися текстовими повідомленнями, зображеннями, відео та іншими медіафайлами.

1.1.1 Види соціальних мереж

Існує кілька типів соціальних мереж, що відрізняються за функціональними можливостями та аудиторією [1]:

1) горизонтальні соціальні мережі – ці мережі орієнтовані на широке коло користувачів і пропонують різноманітні функції для спілкування та обміну інформацією (наприклад, Facebook, Twitter);

2) вертикальні соціальні мережі – спеціалізовані мережі, що орієнтовані на конкретні групи користувачів або інтереси (наприклад, LinkedIn для професійних контактів, Goodreads для книголюбів);

3) фотографічні та відеоплатформи – соціальні мережі, зосереджені на обміні зображеннями та відео (наприклад, Instagram (рис. 1.1), YouTube);

4) месенджери – платформи для миттєвого обміну повідомленнями, що також включають елементи соціальних мереж (наприклад, WhatsApp, Telegram).

Соціальні мережі продовжують еволюціонувати, пропонуючи нові функції та можливості, що відповідають зростаючим потребам користувачів у комунікації та взаємодії.

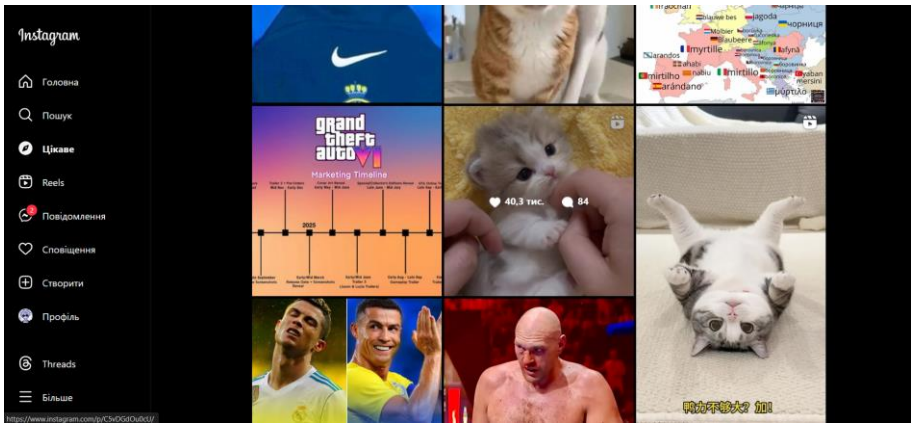

Рисунок 1.1 – Екранні форми фотографічної соціальної мережі

1.1.2 Функціональність соціальних мереж

Соціальні мережі пропонують широкий спектр функціональності, яка забезпечує їх популярність і ефективність у спілкуванні та обміні інформацією між користувачами. Основна функціональність соцмереж:

а) створення та налаштування профілів. Користувачі можуть створювати особисті профілі, де вказують свої дані: ім’я, прізвище, вік, місце проживання, освіту, роботу, інтереси тощо. Профілі зазвичай містять фотографії користувачів, їхні публікації та інформацію про активність на платформі. Налаштування конфіденційності дозволяють контролювати, хто може бачити ті чи інші частини профілю;

б) додавання друзів та підписників. Основною функцією більшості соціальних мереж є можливість додавати друзів або підписуватись на оновлення інших користувачів. Це створює мережу зв’язків, де користувачі можуть стежити за активністю своїх друзів, ділитися з ними своїми оновленнями та взаємодіяти через коментарі та повідомлення;

в) обмін повідомленнями. Соціальні мережі забезпечують кілька способів обміну повідомленнями: приватні повідомлення (direct messages), публічні повідомлення (пости на стінах або у стрічці новин), а також коментарі до публікацій інших користувачів. Деякі платформи також підтримують групові чати та відеоконференції;

г) створення груп та спільнот. Соціальні мережі дозволяють створювати групи та спільноти за інтересами. Учасники цих груп можуть обговорювати теми, ділитися контентом, організовувати події та співпрацювати над проектами. Групи можуть бути як відкритими для всіх користувачів, так і закритими, доступними лише за запрошенням або після схвалення адміністраторами;

д) поширення контенту. Користувачі можуть поширювати різноманітний контент, включаючи текстові пости, зображення, відео, посилання на інші веб-ресурси, документи та інші медіафайли. Цей контент може бути публічним або доступним лише певним групам користувачів залежно від налаштувань конфіденційності;

е) інтерактивні елементи. Соціальні мережі пропонують інтерактивні елементи, які сприяють взаємодії між користувачами. До них належать лайки, коментарі, репости (поширення публікацій інших користувачів), реакції на публікації (наприклад, сердечка, смайлики) та інші форми взаємодії;

ж) сповіщення. Соціальні мережі використовують сповіщення (рис. 1.2) для інформування користувачів про нові повідомлення, коментарі, вподобання їхніх публікацій, запрошення до груп та інші важливі події. Сповіщення можуть надходити як у вигляді внутрішніх повідомлень на платформі, так і через електронну пошту або мобільні додатки;

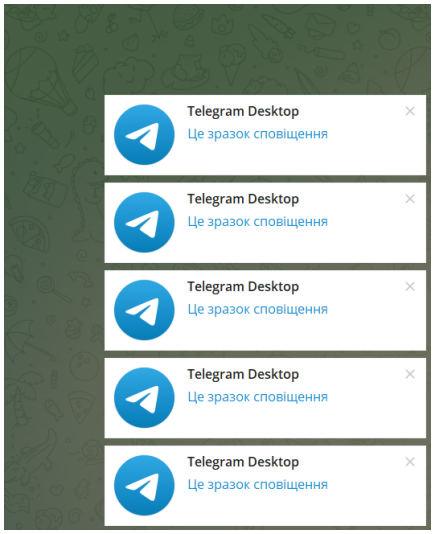

Рисунок 1.2 – Приклад сповіщення із соціальної мережі Telegram

з) інтеграція з іншими сервісами. Багато соціальних мереж інтегруються з іншими сервісами та додатками, такими як музичні стримінгові платформи, сервіси для обміну файлами, новинні сайти та інші. Це дозволяє користувачам легко ділитися контентом з інших джерел та отримувати зручний доступ до додаткових функцій;

и) аналітика та звітність. Соціальні мережі надають інструменти для аналізу активності користувачів, взаємодій з контентом та ефективності рекламних кампаній. Ці дані допомагають користувачам та компаніям краще розуміти свою аудиторію та оптимізувати стратегії взаємодії.

Загалом, соціальні мережі пропонують широкий спектр функціональних можливостей, які забезпечують їхню популярність та ефективність у сучасному цифровому світі.

1.1.3 Вплив соціальних мереж на суспільство

Соціальні мережі за короткий час перетворилися з нішових платформ на невід'ємну частину життя мільярдів людей по всьому світу. Їхній стрімкий ріст та глибоке проникнення у суспільство можна пояснити низкою факторів:

а) доступність та простота використання – завдяки поширенню смартфонів та доступності Інтернету, соціальні мережі стали доступними для людей з усіх верств суспільства. Інтерфейси користувача цих платформ, як правило, прості та інтуїтивно зрозумілі, що робить їх легкими для навігації та використання навіть людьми без досвіду роботи з комп'ютером;

б) зв'язок та спілкування – соціальні мережі дозволяють людям легко спілкуватися з друзями та родиною, незалежно від їхнього місцезнаходження. Вони також створюють можливості для знайомства з новими людьми, які мають схожі інтереси, та для формування онлайн-спільнот;

в) обмін інформацією та новинами – соціальні мережі стали важливим джерелом новин та інформації для багатьох людей. Вони дозволяють людям швидко та легко ділитися інформацією з іншими, а також отримувати доступ до різноманітних джерел новин;

г) розваги та самовираження – соціальні мережі пропонують широкий спектр розваг, таких як ігри, відео, музика та вірусний контент. Вони також надають платформу для самовираження та творчості, дозволяючи людям ділитися своїми думками, ідеями та досвідом з іншими;

д) бізнес та маркетинг – соціальні мережі стали потужним інструментом для підприємств усіх розмірів. Вони дозволяють компаніям рекламувати свої продукти та послуги, спілкуватися з клієнтами та створювати лояльність до бренду;

е) активізм та соціальні зміни – соціальні мережі використовуються для підвищення обізнаності про важливі соціальні та політичні проблеми, а також для мобілізації людей до дій. Вони можуть бути потужним інструментом для соціальних змін.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

У рамках кваліфікаційного проекту передбачається розробка соціальної мережі для спілкування за інтересами. Даний проект має на меті створення платформи, що надає користувачам можливість об’єднуватися у спільноти на основі їхніх інтересів, спілкуватися, обмінюватися думками та ідеями.

Завдання кваліфікаційної роботи включає в себе наступні пункти:

1) розробка функціоналу для створення та управління профілями користувачів, включаючи можливість додавання друзів, редагування особистої інформації та управління налаштуваннями приватності;

2) реалізація системи аутентифікації та авторизації користувачів з використанням сучасних методів безпеки, щоб забезпечити захист конфіденційної інформації;

3) створення системи обміну повідомленнями та коментарів між користувачами, що включає в себе можливість відправлення текстових повідомлень, фотографій та відео;

4) оптимізація додатку для забезпечення масштабованості та ефективності його роботи при великій кількості користувачів та обсязі даних.

В результаті виконання атестаційної роботи очікується отримання функціонального та безпечного веб-додатку, що забезпечує зручну та цікаву спільноту для користувачів з різними інтересами. З особливостей проекту буде те, що в ньому буде повноцінна back-end частина, при цьому вона знаходитиметься поза межами самого проекту, на віддаленому сервері.

Висновки до розділу 1

В розділі 1 було досліджено сучасний стан соціальних мереж, проаналізувавши їхні типи, функціональність та вплив на суспільство.

Було виявлено, що соціальні мережі пропонують широкий спектр функціональних можливостей, які роблять їх популярними та ефективними для спілкування та обміну інформацією. До цих функцій належать створення та налаштування профілів, додавання друзів, обмін повідомленнями, створення груп та спільнот, поширення контенту, інтерактивні елементи, сповіщення, інтеграція з іншими сервісами та аналітика.

Вплив соціальних мереж на суспільство є складним і багатогранним. З одного боку, вони полегшують зв’язок та спілкування, сприяють обміну інформацією та новинами, надають платформу для розваг та самовираження, а також слугують потужним інструментом для бізнесу, маркетингу та активізму. З іншого боку, вони можуть призводити до поширення дезінформації, кіберзалякування, залежності та проблем з конфіденційністю.

Постановка завдання на кваліфікаційну роботу бакалавра чітко окреслює цілі та завдання, які необхідно виконати для розробки соціальної мережі для спілкування за інтересами.

Очікується, що в результаті виконання роботи буде отримано функціональний та безпечний веб-додаток, який забезпечує зручну та цікаву спільноту для користувачів з різними інтересами.

Цей веб-додаток може мати значний позитивний вплив на суспільство, полегшуючи зв’язок між людьми, сприяючи обміну знаннями та ідеями, а також стимулюючи соціальну активність та співпрацю.

РОЗДІЛ 2
АНАЛІЗ ВИМОГ ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Визначення вимог до розроблюваного програмного забезпечення

Метою проекту є розробка соціальної мережі, яка буде мати такі функціональні можливості:

1) авторизація – система повинна підтримувати авторизацію за допомогою логіну та пароля, при закритті сторінки; при повторному відкритті авторизація повинна бути повторна;

2) профіль – користувачі повинні мати можливість створювати та редагувати свій профіль, додаючи фотографію, опис, інтереси тощо;

3) підписки – користувачі повинні мати можливість та підписуватися на інших користувачів;

4) публікації – користувачі повинні мати можливість публікувати дописи, зображення та відео. Публікації повинні бути доступні друзям та підписникам користувача;

5) взаємодія з публікаціями – користувачі повинні мати можливість лайкати інших користувачів;

6) пошук – користувачі повинні мати можливість шукати або фільтрувати інших користувачів;

7) повідомлення – користувачі повинні мати можливість надсилати приватні повідомлення один одному;

8) загальний чат всіх учасників – так як це мережа для певних груп людей і розрахована не на велику аудиторію в ній буде можливість спілкуватись з усіма користувачами відразу.

Цільовою аудиторією проекту є користувачі Інтернету певної групи, які шукають платформу для спілкування, обміну інформацією та розваг.

Нефункціональні вимоги:

1) продуктивність – система повинна бути здатною обробляти достатню кількість користувачів та запитів;

2) використовуваність – система повинна бути простою у використанні та зрозумілою для користувачів;

3) доступність – система повинна бути доступна користувачам з будь-яких пристроїв та браузерів.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Система буде складатися з трьох основних компонентів (рис. 2.1):

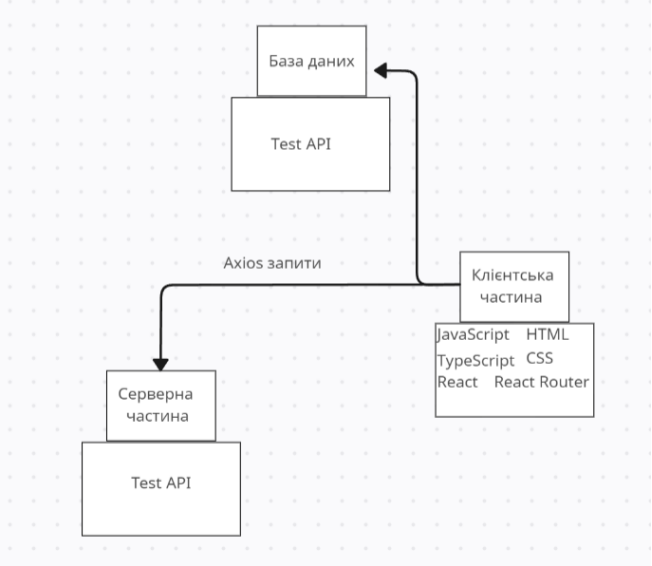

Рисунок 2.1 – Структура проекту

1) клієнтська частина – клієнтська частина буде відповідати за відображення інтерфейсу користувача та взаємодію з користувачем. Вона буде реалізована на TypeScript, React, JavaScript, HTML та CSS, Node [2].

Проаналізувавши макети інших соціальних мереж, було вирішено зробити макет на їх основі. Зверху сторінки буде Header з логотипом та кнопкою Login. Під хедером зліва буде знаходитись navbar, а справа головний блок. Navbar займатиме

приблизно 1/5 ширини сайту, а main 4/5 від всієї ширини. На рисунку 2.2 зображений макет сайту по якому будуть робитись всі компоненти.

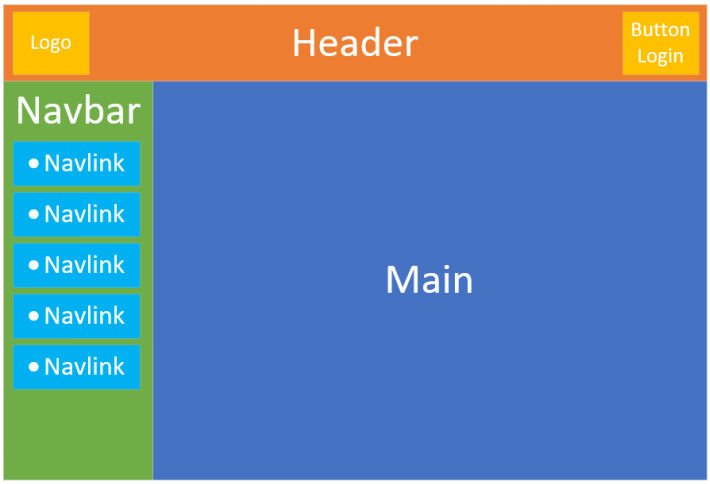

Рисунок 2.2 – Макет соціальної мережі

Структура компонентів соціальної мережі (рис. 2.3) досить проста. Вона складається з двох відділів – Home-page та Navbar, а ті діляться на свої компоненти. Компоненти, розроблені таким чином, можуть бути легко інтегровані в загальний проект. Це означає, що кожен новий компонент може бути швидко доданий до існуючої структури без необхідності значних змін в інших частинах проекту [3].

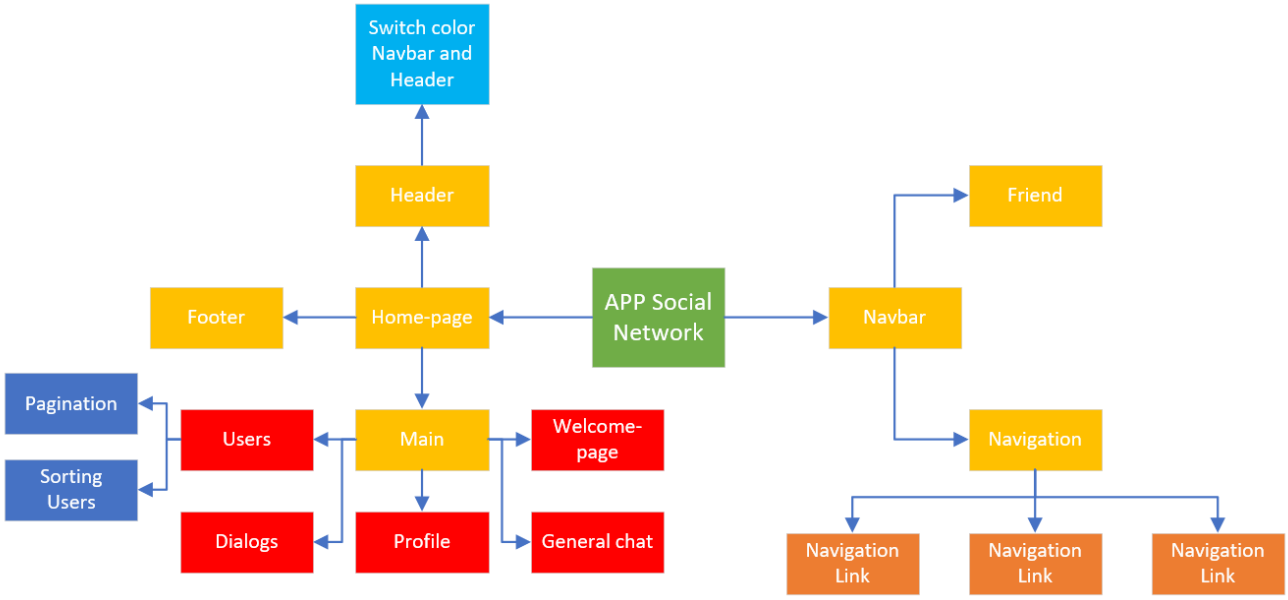

Рисунок 2.3 – Структура компонентів проекту

Наприклад, компонент Dialogs можна додати до будь-якого контейнера, який вимагає навігації, без необхідності переписувати існуючий код. Це значно зменшує час і зусилля, необхідні для підтримки проекту.

Модульний підхід до розробки компонентів дозволяє розробникам працювати паралельно над різними частинами проекту. Кожен компонент розробляється і тестується незалежно, що сприяє швидшому виявленню та виправленню помилок. Наприклад, розробники можуть працювати над компонентами Users-page і Navbar одночасно, не заважаючи один одному. Це сприяє підвищенню продуктивності команди та ефективному використанню ресурсів.

Один з найбільших плюсів такого підходу полягає у можливості повторного використання компонентів. Компоненти, такі як Navigate, можна використовувати в різних частинах проекту або навіть в інших проектах з мінімальними змінами. Це дозволяє розробникам створювати бібліотеки компонентів, які можуть бути використані повторно, що значно економить час і зусилля при розробці нових функціональностей або проектів [4].

Кожен компонент може бути окремо протестований, що забезпечує високу якість коду. Тестування компонентів відбувається швидше і ефективніше, оскільки я можу зосередитися на конкретному функціоналі, а не на великій кількості взаємозалежних елементів. Це також сприяє створенню більш надійного коду, що знижує кількість помилок у кінцевому продукті.

Проект, побудований з використанням таких принципів, легше підтримувати та розширювати. Нові функції можна додавати без значних змін у вже існуючому коді, що знижує ризик виникнення нових помилок. Крім того, оскільки кожен компонент має чітко визначену відповідальність, легше відстежувати і виправляти помилки;

2) серверна частина – серверна частина буде відповідати за підключення до тестової API, обробку запитів та логіку програми. Вона буде реалізована на Redux, а запити через Axios. Серверна частина також забезпечить авторизацію та автентифікацію користувачів, керуючи їхніми сесіями та маркерами доступу.

Обробка даних буде відбуватися на рівні Redux, що дозволить ефективно керувати станом програми та обробляти асинхронні запити. Крім того, будуть реалізовані функції для управління профілями користувачів, постами, коментарями та іншими елементами соціальної мережі. Це дозволить централізовано керувати логікою програми та забезпечити стабільну роботу додатку;

3) база даних – використання тестового сервера та тестової API передбачає, що зберігання даних буде відбуватися на стороні API, а не на локальній базі даних. Це спрощує розробку та тестування, адже не потребує налаштування та обслуговування власної бази даних. При цьому в проекті буде state, куди будуть зберігатись тимчасові файли.

Для створення ER діаграми бази даних соціальної мережі проекту, потрібно проаналізувати доступну документацію та структуру коду. Репозиторій надає уявлення про функціональність додатку, включаючи профілі, користувачів, діалоги та автентифікацію.

Всі основні компоненти бази даних, такі як User, Profile, Posts, Folowing та Messages будуть взаємодіяти через API, що забезпечить централізоване управління даними та їхнє зберігання. ER діаграма (рис. 2.4) відображає взаємозв’язок між цими таблицями та їхніми атрибутами.

Таблиця User буде основною таблицею, яка містить інформацію про всіх користувачів, включаючи їхні унікальні ідентифікатори, імена користувачів, електронні адреси та паролі. Таблиця Profiles буде містити додаткову інформацію про користувачів, таку як ім’я, біографія та соціальні посилання, з посиланням на таблицю користувачів через зовнішній ключ. Post міститимуть записи, створені користувачами, включаючи вміст посту та часову мітку створення.

Повідомлення між користувачами будуть зберігатись у таблиці Messages, що містить інформацію про відправника, одержувача, вміст повідомлення та часову мітку. Following відображає взаємозв’язки між користувачами, дозволяючи їм слідкувати за іншими користувачами.

Ця структура бази даних дозволяє легко розширювати функціональність додатку, додаючи нові таблиці та зв’язки між ними. Вся взаємодія з даними буде здійснюватися через API, що спрощує розгортання та масштабування системи.

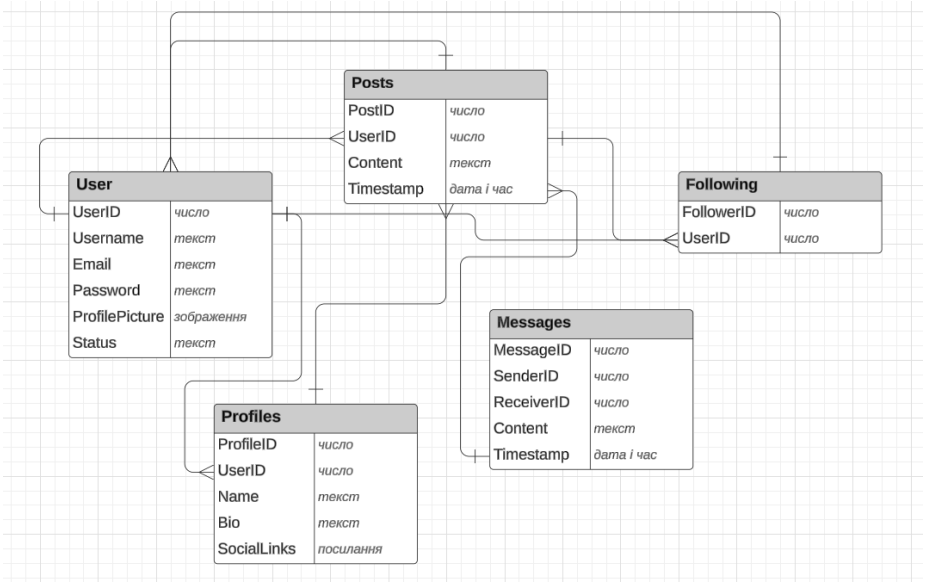

Рисунок 2.4 – ER-діаграма соціальної мережі

Чому вибрано саме ці технології, а не якісь інші:

1) TypeScript – це мова програмування, яка є надбудовою над JavaScript і додає до нього статичну типізацію. Це забезпечує підвищену продуктивність розробників, кращу підтримку IDE та спрощує виявлення помилок під час компіляції. Статична типізація допомагає виявляти помилки ще на етапі написання коду, що знижує кількість помилок у виробничому середовищі [5].

Чому TypeScript краще за інші (рис. 2.5):

– безпека типів – статична типізація дозволяє уникнути багатьох помилок, які важко знайти в JavaScript;

– поліпшена IDE підтримка – автодоповнення, рефакторинг та інші можливості, що значно підвищують продуктивність;

– кросплатформенність – можливість використання з будь-якими фреймворками та бібліотеками, що підтримують JavaScript.

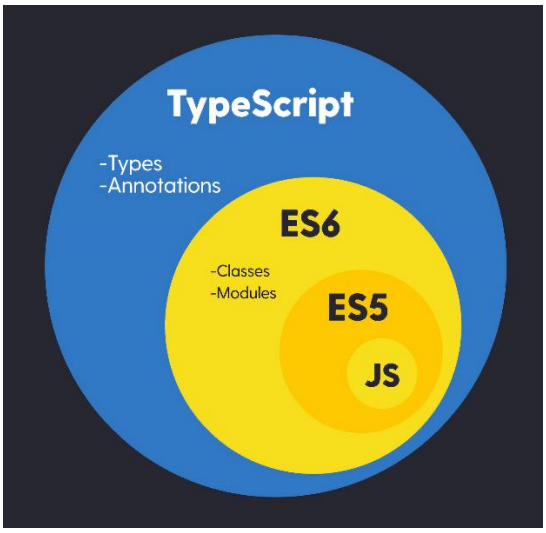

Рисунок 2.5 – Переваги Typescript з ES 5 та ES6

Для соціальної мережі, TypeScript забезпечує високу надійність коду, що особливо важливо для масштабованих проектів з дуже великою кількістю користувачів [6].

2) React – це бібліотека для побудови користувацьких інтерфейсів, яка була створена компанією Facebook. Це найпопулярніша бібліотека з трьох основних бібліотек і фраємворків для створення клієнтської частини сайту. Вона дозволяє створювати багаторазові компоненти, що спрощує розробку та підтримку великих проектів. Основні переваги React – це висока продуктивність завдяки використанню віртуального DOM, можливість створення компонентів на основі стану, а також широка підтримка спільноти [7].

Чому React краще за інші (рис. 2.6):

– компонентний підхід – спрощує розробку та підтримку, дозволяє повторне використання коду [8];

– віртуальний DOM – підвищує продуктивність завдяки ефективному оновленню користувацького інтерфейсу;

– широка підтримка спільноти – велика кількість готових рішень, бібліотек та інструментів, що спрощують розробку.

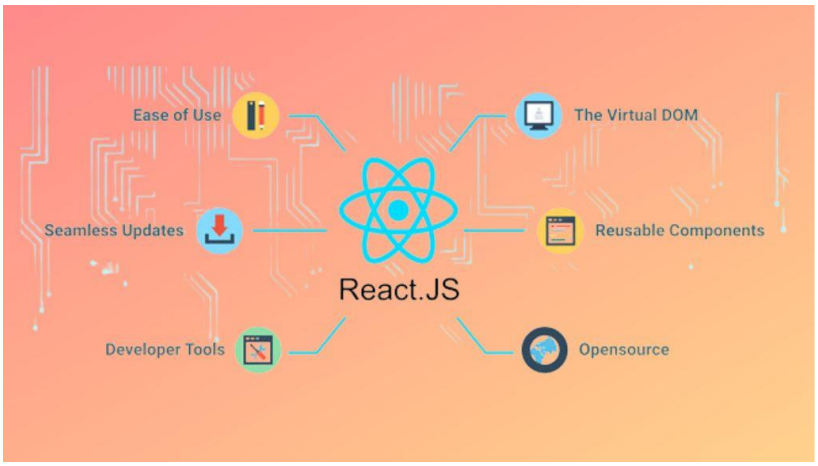

Рисунок 2.6 – Особливості і переваги бібліотеки React

React ідеально підходить для розробки соціальних мереж, оскільки дозволяє створювати динамічний та інтерактивний інтерфейс користувача, що є критично важливим для сучасних соціальних мереж [9].

3) Node.js – це серверна платформа для виконання JavaScript, що працює на движку V8 від Google. Node.js дозволяє створювати високопродуктивні та масштабовані серверні застосунки завдяки неблокуючій моделі вводу/виводу [10].

Чому Node.js краще за інші (рис. 2.7):

– висока продуктивність – неблокуюча модель вводу/виводу дозволяє обробляти велику кількість запитів одночасно;

– єдина мова для клієнта і сервера – спрощує розробку та підтримку, дозволяє використовувати один стек технологій;

– широка екосистема – велика кількість бібліотек та модулів, які можуть бути легко інтегровані у проект;

– швидкість розробки – можливість використання популярних фреймворків, таких як Express.js, що прискорює процес розробки;

– спільнота та підтримка – велика та активна спільнота розробників, що забезпечує підтримку і постійне оновлення технологій;

– кросплатформенність – Node.js можна використовувати на різних платформах, включаючи Windows, Linux і macOS, що робить його універсальним для розробки.

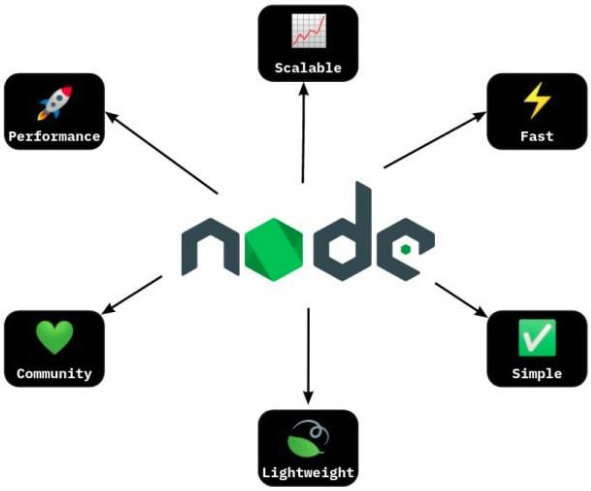

Рисунок 2.7 – Особливості і переваги платформи Node.js

Node.js є оптимальним вибором для серверної частини соціальної мережі завдяки своїй здатності обробляти великі обсяги запитів та забезпечувати високу продуктивність і масштабованість, що необхідно для підтримки активної взаємодії користувачів у соціальній мережі [11].

Висновок до розділу 2

У другому розділі було проведено детальний аналіз вимог до розроблюваного програмного забезпечення соціальної мережі та обґрунтовано вибір технологій для її реалізації. Визначені функціональні вимоги включають авторизацію користувачів, управління профілями, підписки, публікації, взаємодію з публікаціями, пошук користувачів, приватні повідомлення та загальний чат для всіх учасників. Це забезпечить користувачам можливість ефективного спілкування, обміну інформацією та розваг. Нефункціональні вимоги акцентують на продуктивності, використовуваності та доступності системи з будь-яких пристроїв і браузерів.

Для реалізації клієнтської частини обрано TypeScript, React, JavaScript, HTML та CSS, Node, що забезпечить створення динамічного та інтуїтивно зрозумілого інтерфейсу користувача. Серверна частина буде реалізована на Node.js

з використанням Redux для управління станом додатку та Axios для обробки запитів до API. Такий вибір технологій гарантує високу продуктивність, масштабованість та надійність сервісу. Зберігання даних буде здійснюватись на стороні тестового сервера та API, що спрощує розробку і тестування [12].

Використання TypeScript дозволяє виявляти помилки ще на етапі написання коду завдяки статичній типізації, що знижує кількість помилок у виробничому середовищі. React, з його компонентним підходом та віртуальним DOM, забезпечить високу продуктивність і можливість багаторазового використання коду. Node.js, завдяки своїй неблокуючій моделі вводу/виводу, дозволяє обробляти велику кількість запитів одночасно, що є критично важливим для соціальної мережі з активною взаємодією користувачів.

Отже, вибрані технології та підходи забезпечують реалізацію всіх функціональних і нефункціональних вимог до системи, що дозволить створити надійну, продуктивну та зручну у використанні соціальну мережу для цільової аудиторії.

РОЗДІЛ 3
РОЗРОБКА ПРОТОТИПУ СОЦІАЛЬНОЇ МЕРЕЖІ

3.1 Практична реалізація клієнтської частини

Розробка проекту соцмережі починається зі створення компонентів. Ключові етапи включають імпорт необхідних модулів, створення компонентів, їх стильове оформлення та інтеграцію в загальну структуру проекту [13].

Спочатку створимо Navbar, Navigate та додамо до них стилі для зручного користування сайтом:

1) компонент Navbar. Імпорт коспонент та модулів, в лістингу 3.1.

Лістинг 3.1 – Імпорт React, Navigate та інших модулів

```
import React, { FC } from "react";
import { Friends } from "@/components/navbar/friends";
import { Navigate } from "@/components/navbar/navigate";
import module from "@/components/navbar/navbar.module.css";
```

Кінець лістингу 3.1

Батьківська компонента Navbar, що у лістингу 3.2, створена для того, щоб об’єднати елементи навігації Navigate з Friends;

Лістинг 3.2 – компонента Navbar

```
export const Navbar: FC = () => {
  return (
    <div className={module.nav}>
      <Navigate />
      <Friends />
    </div>
  );
};
```

Кінець лістингу 3.2

2) компонент Navigate. Після підключення необхідних модулів потрібно створити компонент Navigate, що у лістингу 3.3, саме він відповідає за навігацію між сторінками. Для кожного елемента він добавляє класи. Також при натиску на

Navlink він перевіряє чи він став активним, якщо так, то він добавляє .active до елемента, і він підсвічується певним кольором;

Лістинг 3.3 – Написання NavLink для переходу між сторінками

```
export const Navigate = () => {
  return (
    <nav>
      <div className={module.navbar}>
        <NavLink className={({ isActive }) => (isActive ? module.active :
"")} to="/profile">Profile</NavLink>
      </div>
      <div className={module.navbar}>
        <NavLink className={({ isActive }) => (isActive ? module.active :
"")} to="/users">Users</NavLink>
      </div>
      <div className={module.navbar}>
        <NavLink className={({ isActive }) => (isActive ? module.active :
"")} to="/dialogs">Messages</NavLink>
      </div>
      <div className={module.navbar}>
        <NavLink className={({ isActive }) => (isActive ? module.active :
"")} to="/news">News</NavLink>
      </div>
      <div className={module.navbar}>
        <NavLink className={({ isActive }) => (isActive ? module.active :
"")} to="/music">Music</NavLink>
</div>
      <div className={module.navSettings}>
        <NavLink className={({ isActive }) => (isActive ? module.active :
"")} to="/setting">Settings</NavLink>
      </div>
    </nav>
  );
};
```

Кінець лістингу 3.3

3) файл стилів navbar.module.css, що у лістингу 3.4: для покращення компонентів, та надання сайту певного дизайну.

Лістинг 3.4 – Стилі під Navbar, NavLink Та Friends

```
.nav {
  grid-area: nav;
  background-color: #181A1B;
  padding: 20px;
  height: 100vh;
}
```

```
.friendsBlock {
  color: white;
}
.friends {
display: flex;
  gap: 10px;
flex-wrap: wrap;
  justify-content: center;
  align-items: center;
}

.friend img {

width: 40px;
}
.navbar {
  display: flex;
  flex-direction: column;
}
.navbar a {
  color: white;
  font-weight: 500;
}
.navbar a.active,
.navSettings a.active {
  color: #1890FF;
  font-size: larger;
}
.navSettings a {
  color: white;
  font-weight: 800;
}
```

Кінець лістингу 3.5

Компонентний підхід до створення компонентів забезпечує не тільки чітку структуру і підтримуваність коду, але також значно спрощує процес розробки та подальшого розширення проекту. Кожен компонент має своє чітке призначення, що дозволяє розробникам легко розуміти його функціональність і взаємодію з іншими частинами проекту [14].

Так виглядає Navbar з логотипом на сторінці (рис. 3.1).

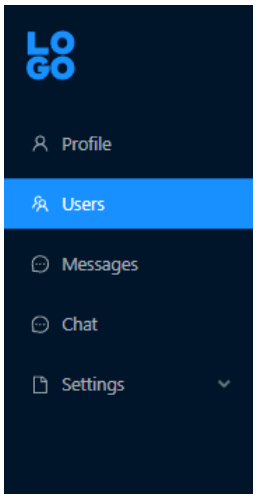

Рисунок 3.1 – Вигляд Navbar соцмережі

Інші компоненти проекту (Welcome, Dialog, Users, Chat, Profile) розроблювались по зразку цього навбару, тому опис розробки цих компонентів пропуститься. Після розробки всіх компонент соціальна мережа виглядає так (рис. 3.2).

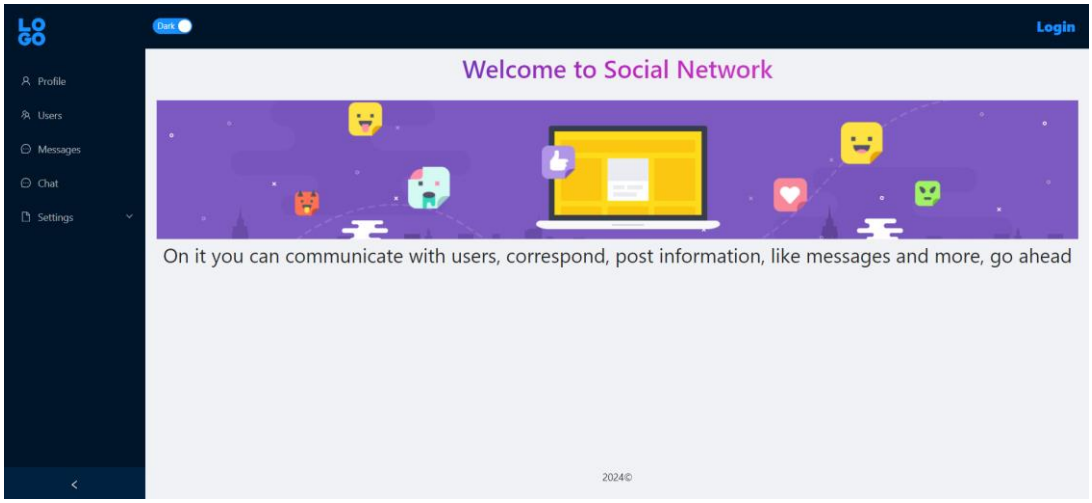

Рисунок 3.2 – Вигляд Navbar соцмережі

Навігаційна панель розміщена праворуч, тоді як header знаходиться ліворуч, а основний контент розташований під ним. Таке розміщення елементів забезпечує інтуїтивно зрозумілий інтерфейс для користувачів. Логічна структура навігаційної панелі праворуч допомагає легко орієнтуватися на сторінці. На header знаходиться

кнопка логінізації та кнопка зміни кольору теми інтерфейсу. Основний контент під header буде змінюватись, залежно від того що потрібно користувачу.

3.2 Розробка Store на Redux та Redux Thunk, Route та Preloader

Далі створюється Store. Він дозволяє централізовано керувати станом додатку, забезпечуючи ефективне управління даними та їх синхронізацію між різними компонентами. Завдяки ньому до кожної компоненти буде приходити ті стани і та інформація яка потрібна саме на ній. Тому ми уникнем зайвих завантажень сторінки та забезпечним швидшу та ефективнішу роботу соціальної мережі. На рисунку 3.3 показана схема взаємодії користувача, reducer та store.

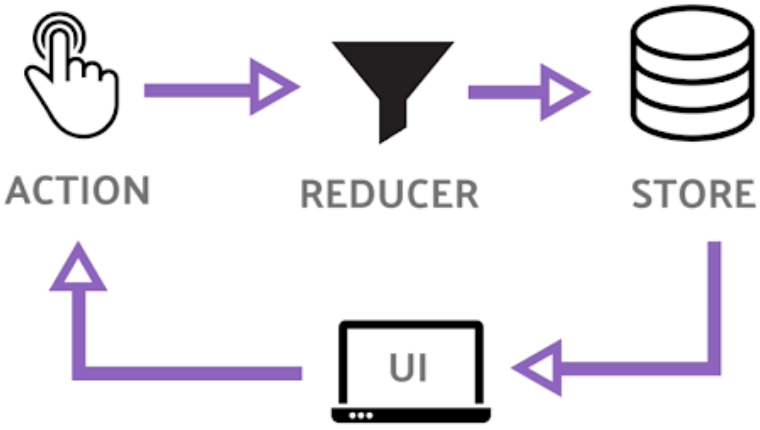

Рисунок 3.3 – Схема взаємодії користувача з сайтом

Перший крок у налаштуванні Redux Store – це імпорт необхідних модулів та ред'юсерів, що у лістингу 3.5.

Лістинг 3.5 – Імпорт необхідних ред'юсерів в Store

```
import { Action, applyMiddleware, combineReducers, compose,
legacy_createStore as createStore } from "redux";
import thunkMiddleware, { ThunkAction } from "redux-thunk";
import { useDispatch } from "react-redux";
import { reducer as formReducer } from "redux-form";
import authReducer from "./auth.reducer";
import dialogsReducer from "./dialogs.reducer";
import profileReducer from "./profile.reducer";
```

```
import usersReducer from "./users.reducer";
import appReducer from "./app.reducer";
import chatReducer from "./chat.reducer";
```

Кінець лістингу 3.5

Після цього імпортуються необхідні модулі з бібліотеки Redux та Redux Thunk, а також ред'юсери для обробки різних частин стану додатку, що у лістингу 3.6.

Всі ред'юсери комбіновуються в один кореневий ред'юсер, який буде використовуватися для створення Store. Це необхідне для розподілу відповідальності та підтримки чистоти та організованості коду у великому додатку.

Лістинг 3.6 – Комбінація всіх окремих ред'юсерів в один

```
const rootReducer = combineReducers({
  profilePage: profileReducer,
  dialogsPage: dialogsReducer,
  usersPage: usersReducer,
  auth: authReducer,
  form: formReducer,
  app: appReducer,
  chat: chatReducer,
});
```

Кінець лістингу 3.6

Для налаштування Store використовується функція createStore, що у лістингу 3.7 з додаванням middleware thunkMiddleware, що дозволяє працювати з асинхронними діями. Також налаштовується підтримка Redux DevTools для зручного дебагінгу.

Лістинг 3.7 – Імпорт необхідних ред'юсерів в Store

```
const composeEnhancers = window.__REDUX_DEVTOOLS_EXTENSION_COMPOSE__ ||
compose;
const store = createStore(rootReducer,
composeEnhancers(compose(applyMiddleware(thunkMiddleware))));
export type AppDispatch = typeof store.dispatch;
export const useAppDispatch = () => useDispatch<AppDispatch>();
window.__store__ = store;
export default store;
```

Кінець лістингу 3.7

Експортуються типи та диспетчер для використання у компоненті, а також додається Store у глобальний об’єкт window для зручного доступу під час розробки. Ред’юсер appReducer обробляє стан ініціалізації додатку, що у лістингу 3.8. Під час ініціалізації змінюється значення initialization на true. Створюються дії для зміни стану ініціалізації та асинхронну дію для отримання даних користувача з сервера перед встановленням стану ініціалізації.

Лістинг 3.8 – Обробка стану сайту

```
const appActions = {
  setInitial: () => ({ type: INITIALIZATION } as const),
};
export const initializeApp = (): any => (dispatch: any) => {
  const promise = dispatch(getAuthUser());
  promise.then(() => {
    dispatch(appActions.setInitial());
```

Кінець лістингу 3.8

Також необхідно розробити reducer dialog, що у лістингу 3.9. При отриманні дії з типом SEND_NEW_MESSAGE, редуктор перевіряє, чи є в дії поле message. Якщо поле message присутнє та не пусте, то створюється нове повідомлення з унікальним ідентифікатором, зображенням користувача (взяте з першого діалогу у стані) та текстом повідомлення, яке було передано в дії. Потім створюється копія стану (щоб уникнути мутації початкового стану) та до копії додається нове повідомлення. Оновлений стан з новим повідомленням повертається як результат роботи редуктора.

Лістинг 3.9 – розробка dialog reducer

```
const dialogsReducer = (state = initialState, action: ActionsTypes):
InitialStateType => {
  switch (action.type) {
    case SEND_NEW_MESSAGE: {
  if (action.message) {
        const newMessage = {
          id: v1(),
```

```
          img: state.dialogs[0].img,
          message: action.message,
        };
        const stateCopy = { ...state };
        stateCopy.messages = [...state.messages];
        stateCopy.messages.push(newMessage);
        return stateCopy;
      }
      return state;
    }
    default:
      return state;
  }
};
```

Кінець лістингу 3.9

Після цього експортуєм об’єкт actionsDialogs, який містить дію sendNewMessage. Дія sendNewMessage є функцією, яка приймає рядок message як аргумент. Вона повертає об’єкт дії, що містить тип дії SEND_NEW_MESSAGE та передане повідомлення. Тип дії SEND_NEW_MESSAGE використовується редуктором для визначення того, яку зміну слід застосувати до стану. Повернений об’єкт має типізацію TypeScript as const, що гарантує незмінність типу дії та значень. Експортуючи цю дію, ми дозволяємо іншим частинам додатку викликати її для створення нових повідомлень у діалогах. Це робить код більш організованим та полегшує управління діями у додатку. Таким чином, sendNewMessage забезпечує створення і відправлення нових повідомлень у стані додатку.

Далі робляться Route, що у лістингу 3.10. Цей код визначає функціональний компонент App у React, який налаштовує маршрутизацію додатку за допомогою react-router-dom. Компонент App використовує компонент Router для огортання всіх маршрутів додатку. Всередині Router використовується компонент Switch, який забезпечує рендеринг лише першого відповідного маршруту. Компоненти Route визначають різні маршрути для додатку. Маршрут з шляхом /home рендерить

компонент Home при доступі до кореневого URL. Маршрут з шляхом /profile рендерить компонент Profile. Маршрут з шляхом /dialogs рендерить компонент Dialogs. Використання exact у першому маршруті гарантує, що він рендериться відразу при завантаженні сторінки. Ця структура забезпечує чітку навігацію між різними сторінками додатку. Використання react-router-dom дозволяє створити SPA (Single Page Application) з плавним перемиканням між сторінками без перезавантаження. Компонент App є основою для маршрутизації в додатку, забезпечуючи зручність користування та організованість навігації.

Лістинг 3.10 – Структура App через Switch та Routes

```
const App = () => {
  return (
    <Router>
      <Switch>
        <Route exact path="/" component={Home} />
        <Route path="/profile" component={Profile} />
        <Route path="/dialogs" component={Dialogs} />
      </Switch>
    </Router>
  );
};
```

Кінець лістингу 3.10

Під час перезавантаження сторінки, або під час заміни Route на інший потрібно викликати на сторінку блок, який показує що сторінка завантажується. Компонент Preloader, що у лістингу 3.11, просто відображає текст «Loading...». Компонент App використовує хук useSelector з бібліотеки react-redux для доступу до стану initialization з глобального стану Redux.

Лістинг 3.11 – Preloader встановлюється в App

```
const Preloader = () => <div>Loading...</div>;
const App = () => {
```

```
const initialization = useSelector((state) => state.app.initialization);
if (!initialization) {
  return <Preloader />;
}
```

Кінець лістингу 3.11

Якщо initialization має значення false, компонент App (додаток А) повертає компонент Preloader, відображаючи «Loading...». Це означає, що додаток знаходиться в процесі ініціалізації і ще не готовий до роботи. Коли initialization змінюється на true, компонент App продовжує рендерити решту свого вмісту (який не показаний у цьому фрагменті коду). Це забезпечує плавний користувацький досвід, показуючи індикатор завантаження поки додаток ініціалізується. Таким чином, код управляє станом ініціалізації додатку та відображає відповідний контент в залежності від його значення.

3.3 Розробка підключень до тест API

Перед тим як інтегрувати Test API до сайту, я провів детальне вивчення його документації. Це був важливий крок, оскільки правильне розуміння можливостей і обмежень API дозволяє ефективно його використовувати та уникати помилок. Перш за все, я ознайомився із загальними відомостями про API, такими як призначення, основні можливості та структура. Це допомогло мені зрозуміти, які функції API надає і як вони можуть бути корисними для нашого додатку. Документація описує, як здійснюється аутентифікація користувачів. Я вивчив методи, які використовуються для логіну, реєстрації та отримання поточного статусу користувача. Особливо важливо було зрозуміти, як працює система з API-ключами та куками. Детально переглянув всі доступні ендпоінти API. Це включало методи для роботи з користувачами, повідомленнями, друзями, профілями тощо. Я занотував собі URL кожного ендпоінта, методи запитів (GET, POST, DELETE) і формат очікуваних даних. Дуже важливо було зрозуміти, які дані потрібно відправляти в запитах і в якому форматі приходять відповіді. Я переглянув

приклади запитів і відповідей, щоб знати, які поля є обов'язковими, які дані потрібно обробляти на клієнтській стороні.

Дане API має обмеження на 1000 (рис. 3.4) запитів на день для безкоштовних аккаунтах. Я буду використовувати безкоштовну версію, адже навіть 1000 запитів в день навряд чи використається.

	Безкоштовний акаунт	Преміум акуант	Експерт акаунт
Сумарна кількість BCIX запитів на день	1000	1000	5000
Сумарна кількість BCIX запитів в секунду	11	11	11
Кількість POST/PUT/DELETE/GraphQL запитів на годину	20	-	-

Рисунок 3.4 – Таблиця кількості запитів для різних аккаунтів

Для запитів на API було використано бібліотеку axios для виконання HTTP-запитів до API. У лістингу 3.12 можна побачити налаштування екземпляру axios для роботи з API соціальної мережі, а також визначаємо типи даних та коди результатів для обробки відповідей від сервера.

Лістинг 3.12 – Імпорт axios та налаштування запиту до сервера

```
import axios from "axios";
import { UserType } from "@/types/types";
export const instanse = axios.create({
  baseURL: "https://social-network.samuraijs.com/api/1.0/",
  withCredentials: true,
  headers: { "API-KEY": "8824ceb4-d76e-4c2d-84eb-bcba650d97c7" },
});
```

Кінець лістингу 3.12

Після налаштування запитів створюється екземпляр axios з базовим URL, включенням cookies у всі запити, та вказанням API-ключа у заголовках.

У лістингу 3.13 ми визначаємо перерахування ResultCodesEnum для обробки кодів результатів, які повертає сервер. Ці коди результатів допомагають зрозуміти, чи був запит успішним, чи виникли помилки, або чи потрібна CAPTCHA.

Лістинг 3.13 – Визначення що має повертати сервер

```
export enum ResultCodesEnum {
  Succses = 0,
  Error = 1,
  CaptchaIsRequired = 10,
}
```

Кінець лістингу 3.13

Для визначення типів даних для стандартизації та типізації відповідей API у TypeScript потрібно створити два типи даних, що у лістингу 3.14:

1) тип GetItemsType визначає об’єкт, який містить масив items з елементами типу UserType, загальну кількість totalCount елементів та можливу помилку error, яка може бути рядком або null;

2) тип APIResponseType є універсальним (generic) типом, який приймає два параметри типу: D (за замовчуванням порожній об’єкт) і RC (за замовчуванням ResultCodesEnum). Цей тип описує об’єкт відповіді API, що містить дані data типу D, код результату resultCode типу RC і масив повідомлень messages. Ці типи даних визначають структуру відповідей для різних запитів до API.

Лістинг 3.14 – Визначення типів даних в API

```
export type GetItemsType = {
  items: Array<UserType>;
  totalCount: number;
  error: string | null;
};
export type APIResponseType<D = {}, RC = ResultCodesEnum> = {
  data: D;
  resultCode: RC;
```

```
  messages: Array<string>;
};
```

Кінець лістингу 3.14

Після цього до кожної сторінки створюється модуль для API. На прикладі AuthAPI під компоненту Auth, ми імпортуємо модуль AuthAPI, який містить методи для автентифікації користувача. Потім ми використовуємо ці методи в компоненті Auth, щоб керувати автентифікацією користувача через наш сайт.

На лістингу 3.15 визначається об’єкт AuthAPI, який містить методи для автентифікації користувача.

Лістинг 3.15 –Метод для аунтифікації користувача

```
export const AuthAPI = {
  me() {
    return instanse
      .get<APIResponseType<MeResponseDataType>>(`auth/me`)
      .then((res) => res.data);
  },
  login(
    email: string,
    password: string,
    rememberMe = false,
    captcha: null | string = null
  ) {
    return instanse
      .post<APIResponseType<LoginResponseDataType>>(`auth/login`, {
        email,
        password,
        rememberMe,
        captcha,
      })
      .then((res) => res.data);
  },
  logout() {
    return instanse.delete(`auth/login`).then((res) => res.data);
  },
};
```

Кінець лістингу 3.15

Метод me() виконує GET-запит для отримання даних про поточного користувача. Метод login() виконує POST-запит для входу користувача в систему з

переданими обліковими даними. Метод logout() виконує DELETE-запит для виходу користувача з системи. Кожен метод повертає об’єкт типу Promise, який відображає результат виконання запиту до сервера.

Після даного коду з’являється можливість виконання запитів, але лише з сторінки аунтифікації (рис. 3.5). Для інших сторінок (Profile, Users, Messages, Chat) прописуються схожі методи які містять ті чи інші запити, які необхідні на тій чи іншій сторінці.

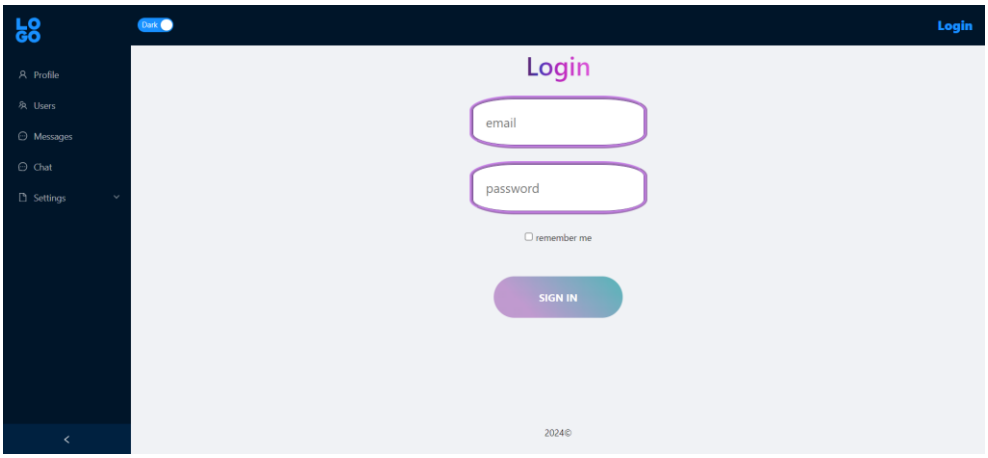

Рисунок 3.5 – Сторінка аунтифікації

3.4 Тестування веб-сайту

Тестування сайту розпочнеться з перевірки всіх коспонент та їх запитів на правильність. Спочатку перевіриться аунтифікація (рис. 3.6).

Загальні класи коду стану HTTP:

1) 1xx – сервер обдумує запит;

2) 2xx – запит було успішно виконано, і сервер надав браузеру очікувану відповідь;

3) 3xx – вас перенаправили в інше місце. Запит отримано, але є переадресація якась;

4) 4xx – сторінка не знайдена. Не вдалося отримати доступ до сайту або сторінки. Запит зроблено, але сторінка недійсна – це помилка на стороні веб-сайту в розмові та часто з’являється, коли сторінки на сайті не існує;

5) 5xx – збій. Клієнт зробив дійсний запит, але сервер не зміг виконати запит.

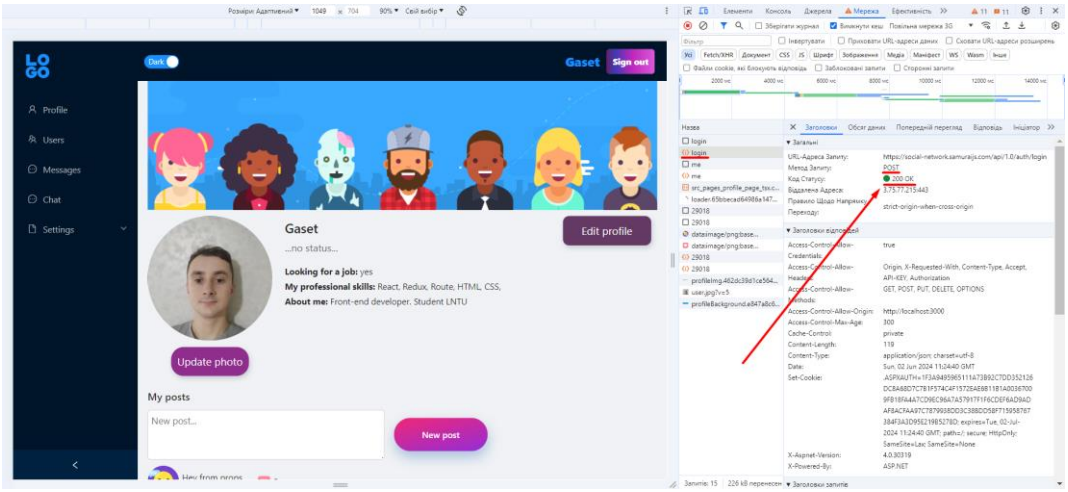

Рисунок 3.6 – Метод login та HTTP статус 200

На рисунку 3.7 видно, що post-запит login пройшов успішно та показує статус 200. Тепер можна оглянути сторінку, завантажити свою аватарку та інформацію про себе. Get метод me також працює успішно та приймає інформацію з сервера.

Після успішного виконання запитів на автентифікацію та отримання даних про користувача здійснюється перевірка функціонування компонентів Profile, Users, Messages та Chat.

На сторінці Users можна переглядати список усіх користувачів, які надходять з сервера. Користувач має можливість переглядати детальну інформацію про кожного з них, включаючи ім’я, аватарку та статус. Реалізована функціональність підписки та відписки від користувачів дозволяє створювати мережу контактів, взаємодіяти з іншими користувачами та слідкувати за їхньою активністю. Це сприяє більшій взаємодії та соціалізації в межах платформи (рис. 3.7). Також на сторінці Users забезпечується зручний інтерфейс для перегляду та взаємодії з іншими користувачами. Користувачі можуть легко підписуватись на оновлення інших користувачів, що дозволяє бути в курсі їхньої активності та новин. Ця функціональність є ключовою для створення динамічної і взаємодіючої спільноти, де кожен може знайти нових друзів та підтримувати зв'язок з існуючими.

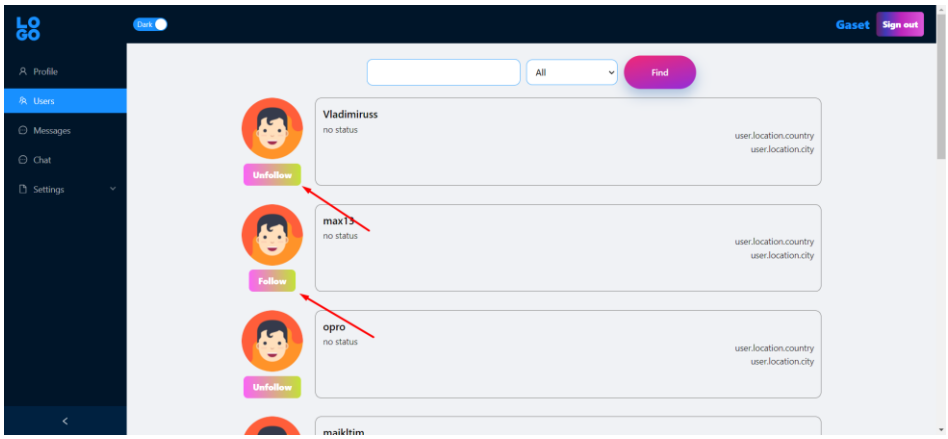

Рисунок 3.7 – Демонстрація підписки та відписки на сторінці Users

Переглядаючи рисунок 3.7, можна бачити, що кожен з методів отримання та відправки даних працює коректно, відображаючи статуси відповідей від сервера.

В Profile користувач може бачити своє ім’я, адресу електронної пошти та інші особисті дані, які були збережені на сервері.

На сторінці Messages користувач може надсилати та отримувати повідомлення в реальному часі, забезпечуючи живу комунікацію з іншими користувачами системи. Всі повідомлення зберігаються на сервері та синхронізуються між користувачами.

На сторінці Chat реалізовано функціонал групових чатів, де користувачі можуть обмінюватися повідомленнями у рамках створених груп. Всі повідомлення в межах чату надходять від сервера та відображаються в реальному часі.

Загалом, цей розширений аналіз демонструє повну інтеграцію клієнтської та серверної частин системи, також підтверджує надійність виконання API-запитів для підтримки функціональності веб-додатку.

Висновок до розділу 3

У третьому розділі було детально розглянуто процес розробки прототипу соціальної мережі. Спочатку було описано архітектуру програмного додатку, що включає серверну частину, базу даних та клієнтську частину. Це дозволило

зрозуміти, як кожна з компонент взаємодіє між собою та які технології використовуються для їх реалізації.

Особлива увага приділялась серверній частині, яка базується на Node.js, та базі даних, де використовується MongoDB. Така комбінація забезпечує масштабованість та ефективність роботи додатку. Клієнтська частина була реалізована за допомогою React, що дозволяє створювати динамічний та швидкий інтерфейс користувача.

Опис структури проекту включав розподіл на модулі та компоненти, що спрощує процес розробки та подальшого обслуговування системи. Було створено основні компоненти сайту, такі як головна сторінка, сторінка користувача, сторінка створення постів та коментарів.

Функціонал користувачів включає реєстрацію, авторизацію, редагування профілю, додавання друзів та управління налаштуваннями приватності. Реалізація функціоналу постів та коментарів забезпечує можливість створювати, редагувати та видаляти пости, а також залишати коментарі до них. Це сприяє активній взаємодії між користувачами.

Тестування та налагодження системи проводилось для забезпечення коректної роботи усіх компонентів. Було здійснено модульне та інтеграційне тестування, що дозволило виявити та виправити помилки на ранніх стадіях розробки. Також було проведено тестування користувацького інтерфейсу для забезпечення зручності та інтуїтивної зрозумілості системи.

Розробка прототипу соціальної мережі продемонструвала важливість ретельного планування та дотримання сучасних технологічних стандартів. Завдяки використанню React та Node.js вдалося створити ефективний та масштабований додаток, який відповідає вимогам сучасних користувачів.

Отримані результати можуть бути використані для подальшого розвитку та вдосконалення системи, зокрема, для додавання нових функцій та оптимізації існуючих. Прототип соціальної мережі вже на цьому етапі демонструє високий потенціал та можливості для розширення.

Таким чином, проведена робота підтвердила доцільність обраних технологій та підходів, а також їх ефективність для створення сучасних веб-додатків. Це відкриває нові перспективи для розвитку соціальних платформ, які здатні забезпечити користувачів зручними та функціональними інструментами для комунікації та взаємодії.

ВИСНОВКИ

Результати кваліфікаційної роботи бакалавра включають розробку функціонального прототипу соціальної мережі, що охоплює серверну частину, базу даних та клієнтську частину. Завдання виконано в повному обсязі, досягнуто якісних і кількісних показників, які включають інтеграцію функціональних можливостей користувача, публікацій, коментарів і обміну повідомленнями в реальному часі.

У роботі використовуються сучасні технології React, TypeScript та Node.js, що відповідає сучасним тенденціям у сфері розробки веб-додатків.

Практичне використання матеріалів дослідження можливе у сфері розробки спеціалізованих веб-сервісів для комунікацій за інтересами. Це може бути корисним для організацій, яким потрібно створити власні спільноти або платформи для обміну інформацією та досвідом.

Подальші шляхи вдосконалення розробки включають розробка власного повноцінного бекенду та розширення функціональних можливостей, таких як: обмін зображеннями, відео, добавлення стікерів та повна реєстрація через сторонні сервіси; підвищення безпеки даних (особливо в покращені токенів), оптимізацію продуктивності та інтеграцію з іншими сервісами.

Завдання кваліфікаційної роботи включало розробку ключових функцій для соціальної мережі, було успішно виконано усі пункти, а саме:

1) розробив функціонал для створення та управління профілями користувачів за допомогою React, Redux та Typescript, включаючи додавання друзів, редагування особистої інформації та налаштування приватності. Використовував react-router для маршрутизації, redux-thunk для асинхронних дій, та Formik для роботи з формами;

2) реалізував систему аутентифікації та авторизації, що забезпечує захист конфіденційної інформації користувачів. Застосовував session cookies для управління сесіями;

3) розробив систему обміну повідомленнями та коментарями, що дозволяє відправляти текстові повідомлення, фотографії та відео. Використовував Redux для реалізації реального часу та для управління станом повідомлень;

4) оптимізував додаток для забезпечення його масштабованості та ефективної роботи при великій кількості користувачів і обсязі даних. Використовував code-splitting, lazy loading та мемоізацію для зменшення часу завантаження та підвищення продуктивності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тренди соціальних мереж. URL: https://bannerboo.com/ua/blog/trendy-sotsialnykh-merezh (дата звернення: 21.05.2024);

2. Побудова організаційної структури онлайн. URL: https://creately.com (дата звернення: 13.04.2024);

3. Веб-технології для розробників. URL: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата звернення: 15.04.2024);

4. Документація React. URL: https://ru.legacy.reactjs.org/docs/getting-started.html (дата звернення: 17.04.2024);

5. Синтаксис Typescript. URL: https://www.typescriptlang.org/ (дата звернення: 13.05.2024);

6. Інформація про TypeScript. URL: https://foxminded.ua/ru/typescript/ (дата звернення: 10.05.2024);

7. React Router Documentation. URL: https://reactrouter.com/web/guides/quick-start (дата звернення: 26.04.2024);

8. Інформація про React. URL: https://web-developer.in.ua/assets/articles/react/react-top/react-top.html (дата звернення: 26.04.2024);

9. Redux Documentation. URL: https://redux.js.org/introduction/getting-started (дата звернення: 10.05.2024);

10. Node.js Documentation. URL: https://nodejs.org/en/docs/ (дата звернення: 23.04.2024);

11. Інформація про Node.js. URL: https://itvdn.com/ru/blog/article/bestpractices-nodejs (дата звернення: 23.04.2024);

12. Документація по API. URL: https://social-network.samuraijs.com (дата звернення: 22.04.2024);

13. Вікіпедія Соціальна мережа. URL: https://uk.wikipedia.org/wiki (дата звернення: 13.05.2024);

14. Інформація про React. URL: https://developer.mozilla.org/ru/docs/Learn/Tools_and_testing/Clientside_JavaScript_frameworks/React_getting_started (дата звернення: 21.05.2024).

ДОДАТКИ

Додаток А

Код файлів App.tsx та loyout.tsx

```
import React, { useEffect } from "react";
import { BrowserRouter } from "react-router-dom";
import { connect, Provider } from "react-redux";
import { compose } from "redux";

import { initializeApp } from "@/redux/app.reducer";
import { Preloader } from "@/components/common/preloader/preloader";
import store, { AppStateType } from "@/redux/store-redux";
import { LayoutApp } from "@/components/layout/layout";
import "antd/dist/antd.css";
import "./App.css";

type MapStatePropsTypes = ReturnType<typeof mapStateToProps>;
type MapDispatchPropsTypes = { initializeApp: () => void };

function App(props: MapStatePropsTypes & MapDispatchPropsTypes) {
  useEffect(() => props.initializeApp());
  if (!props.initialization) {
    return <Preloader />;
  }
  return (
    // <div className="app-wrapper">
    <LayoutApp />
    // </div>
  );
}

const mapStateToProps = (state: AppStateType) => ({
  initialization: state.app.initialization,
});

const AppMain = () => {
  return (
    //  <React.StrictMode>
    <BrowserRouter>
      <Provider store={store}>
        <AppContainer />
      </Provider>
    </BrowserRouter>
    // </React.StrictMode>
  );
};

const AppContainer = compose<React.ComponentType>(
  connect(mapStateToProps, { initializeApp })
```

```
)(App);

export default AppMain;
import React, { Suspense, useState } from "react";
import { NavLink } from "react-router-dom";
import { Route, Routes } from "react-router-dom";
import {
  FileOutlined,
  TeamOutlined,
  UserOutlined,
  MessageOutlined,
} from "@ant-design/icons";
import type { MenuTheme } from "antd";
import { Layout, Menu, Switch } from "antd";
import { Row } from "antd";

import { Preloader } from "@/components/common/preloader/preloader";
import { HomePage } from "@/components/home-page/home-page";
import { Logo } from "@/components/logo/logo";
import { SignUp } from "@/components/sign-up/sign-up";
import style from "@/components/layout/layout.module.css";

const { Header, Content, Footer, Sider } = Layout;

const DialogsPage = React.lazy(() => import("@/pages/dialogs.page"));
const ProfilePage = React.lazy(() => import("@/pages/profile.page"));
const UsersPage = React.lazy(() => import("@/pages/users.page"));
const ChatPage = React.lazy(() => import("@/pages/chat.page"));
const LoginPage = React.lazy(() => import("@/pages/login.page"));
const NotFoundPage = React.lazy(() => import("@/pages/not-found.page"));

type MenuItem = Required<any>["items"][number];

function getItem(
  label: React.ReactNode,
  key: React.Key,
  icon?: React.ReactNode,
  children?: MenuItem[]
): MenuItem {
  return {
    key,
    icon,
    children,
    label,
  } as MenuItem;
}

const items: MenuItem[] = [
  getItem(<NavLink to="/profile">Profile</NavLink>, "1", <UserOutlined />),
  getItem(<NavLink to="/users">Users</NavLink>, "2", <TeamOutlined />),
```

```
  getItem(<NavLink to="/dialogs">Messages</NavLink>, "3", <MessageOutlined
/>),
  getItem(<NavLink to="/chat">Chat</NavLink>, "4", <MessageOutlined />),
  getItem("Settings", "sub2", <FileOutlined />, [getItem("Team 2", "8")]),
];

export const LayoutApp: React.FC = () => {
  const [collapsed, setCollapsed] = useState(false);
  const [theme, setTheme] = useState<MenuTheme>("dark");
  const changeTheme = (value: boolean) => {
    setTheme(value ? "dark" : "light");
  };
  const headerStyle: React.CSSProperties = {
    textAlign: "center",
    color: "#fff",
    height: 64,
    paddingInline: 50,
    lineHeight: "64px",
    backgroundColor: theme === "dark" ? "#001529" : "#fff",
    padding: "0 0 0 10px",
  };

  return (
    <Layout style={{ minHeight: "100vh" }}>
      <Sider
        theme={theme}
        collapsible
        collapsed={collapsed}
        onCollapse={(value: any) => setCollapsed(value)}
      >
        <div className="logo">
          <Logo />
        </div>
        <Menu theme={theme} mode="inline" items={items} />
      </Sider>
      <Layout className="site-layout">
        <Header className="site-layout-background" style={headerStyle}>
          <Row className={style.wrapper} justify="space-between"
align="middle">
            <Switch
              className={style.switch}
              checked={theme === "dark"}
              onChange={changeTheme}
              checkedChildren="Dark"
              unCheckedChildren="Light"
            />
            <SignUp />
          </Row>
        </Header>
        <Content style={{ margin: "0 16px" }}>
```

```
          <Suspense fallback={<Preloader />}>
            <Routes>
              <Route path="/" element={<HomePage />} />
              <Route path="/profile" element={<ProfilePage />} />
              <Route path="/users/profile/:id" element={<ProfilePage />} />
              <Route path="/dialogs/" element={<DialogsPage />} />
              <Route path="/users" element={<UsersPage />} />
              <Route path="/chat" element={<ChatPage />} />
              <Route path="/login" element={<LoginPage />} />
              <Route path="*" element={<NotFoundPage />} />
            </Routes>
          </Suspense>
        </Content>
        <Footer style={{ textAlign: "center" }}>2024©</Footer>
      </Layout>
    </Layout>
  );
};
```