МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»

освітньо-професійної програми «Інформаційні технології проектування»

на тему: Вебдодаток соціальної мережі «ConnectHub»

Здобувача (ки) групи ІТ-01 Денисенка Федора Михайловича
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ _____Федір ДЕНИСЕНКО__
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник старший викладач кафедри ІТ, к.т.н. Ольга БОЙКО _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2024

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра інформаційних технологій

Спеціальність 122 «Комп'ютерні науки»

Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ

В. о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО

«__» _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Денисенку Федору Михайловичу

1 Тема роботи *Вебдодаток соціальної мережі «ConnectHub»*

керівник роботи *Бойко Ольга Василівна, к.т.н., старший викладач*

затверджені наказом по університету від «___»_____2024 р. _____

2 Строк подання студентом роботи « » _____ 2024 р.

3 Вхідні дані до роботи *текстові та графічні матеріали для наповнення вебдодатку*

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *зміст, вступ, аналіз предметної області вебдодатків соціальних*

мереж, моделювання та проєктування вебдодатку соціальної мережі «ConnectHub», практична реалізація вебдодатку соціальної мережі «ConnectHub», висновки, список використаних джерел, додаток А, додаток Б, додаток В, додаток Г.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) *актуальність роботи, проблематика області вебдодатків соціальних мереж, мета роботи, задачі роботи, об'єкт та предмет дослідження, порівняння продуктів-аналогів, функціональні вимоги, діаграма у нотації IDEF0, декомпозиція діаграми у нотації IDEF0, діаграма варіантів використання вебдодатку, схема бази даних, архітектура вебдодатку, використовувані інструменти розробки, демонстрація роботи вебдодатку, висновки, апробація.*

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання *01.03.2024*

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Ознайомлення з вимогами кваліфікаційної роботи	01.03.2024 – 05.03.2024	
2	Визначення мети роботи, об’єкту та предмету дослідження	06.03.2024 – 09.03.2024	
3	Аналіз предметної області, актуальності розробки продукту, проблематики предметної області	10.03.2024 – 14.03.2024	
4	Порівняння аналогів розроблюваного вебдодатку	15.03.2024 – 18.04.2024	
5	Постановка мети та задач проєкту	19.03.2024 – 20.03.2024	
6	Складання технічного завдання	21.03.2024 – 25.03.2024	
7	Планування робіт	26.03.2024 – 29.03.2024	
8	Проєктування вебдодатку соціальної мережі «ConnectHub»	01.04.2024 – 03.04.2024	
9	Проєктування бази даних вебдодатку	04.04.2024 – 05.04.2024	
10	Розробка вебдодатку соціальної мережі «ConnectHub»	06.04.2024 – 15.05.2024	
11	Тестування вебдодатку соціальної мережі «ConnectHub»	16.05.2024 – 17.05.2024	
12	Завантаження вебдодатку соціальної мережі «ConnectHub» на хостинг	18.05.2024 – 20.05.2024	
13	Оформлення пояснювальної записки	21.05.2024 – 24.05.2024	
14	Звітування про проведену роботу	25.05.2024 – 02.06.2024	

Студент _____ Федір ДЕНИСЕНКО
(підпис)

Керівник роботи _____ к.т.н., ст. викл. Ольга БОЙКО
(підпис)

АНОТАЦІЯ

Тема кваліфікаційної роботи бакалавра «Вебдодаток соціальної мережі «ConnectHub»».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 21 найменування, додатків А, Б, В, Г. Загальний обсяг роботи − 145 сторінок, у тому числі 66 сторінок основного тексту, 3 сторінки списку використаних джерел, 76 сторінок додатків.

Актуальність роботи полягає в забезпеченні людей стабільним взаємозв’язком в мережі Інтернет за допомогою смартфонів, якими володіє майже кожен.

Мета роботи: розробка вебдодатку соціальної мережі з інтуїтивно зрозумілим інтерфейсом, що забезпечить користувачів стабільним зв'язком у віртуальному середовищі. Він дозволить створювати унікальні сторінки особистого профілю, заводити нові знайомства, підтримувати контакт з друзями та рідними, обговорювати новини та важливі суспільні питання.

Об’єктом роботи є сфера соціальних мереж, а предметом – створення вебдодатку соціальної мережі за допомогою сучасних інструментів веброзробки.

Ключові слова: соціальна мережа, вебдодаток, вебсервер, база даних, front-end, back-end, full-stack, Nodejs, Express, MySql.

ЗМІСТ

ВСТУП

У наш час, коли сфера ІТ стрімко зростає, та користувачів мережі Інтернет стає дедалі більше, важко виключати той факт, що кожен хоче мати можливість підтримувати контакт та взаємодіяти з іншими навіть у віртуальному просторі.

Саме соціальні мережі бездоганно справляються з цією функцією. Надаючи користувачам інтуїтивно зрозумілий та повноцінний інтерфейс, такі додатки дозволяють швидко розібратися у своєму функціоналі та захоплюють увагу. Завдяки цікавим концепціям написання постів та коментування дописів інших, соціальні мережі дають можливість дізнатися останні новини від своїх колег, друзів та рідних, прослідкувати за їх життям і здобутками. Невід'ємною перевагою є наявність комплексних алгоритмів, що дають можливість заводити нові знайомства з однодумцями та допомагають знайти спільноту за інтересами [1].

Будь-який користувач може гнучко налаштувати свій профіль, аби змінити його доступність і приховати конфіденційну інформацію, додати дату народження, встановити фото облікового запису тощо. Наявність таких розширених параметрів регулювання акаунту дозволяє кожному створити свій унікальний та неповторний образ у мережі. Це вкрай важливо, адже завдяки своїй соціальній функції, такі додатки дуже популярні і серед рекламодавців, що може гарантувати стабільний заробіток активним та відомим користувачам [2].

Соціальні мережі забезпечують зручний та швидкий зв'язок з іншими, дозволяють поширювати останні новини, ділитися своїми думками та досягненнями, заводити нових друзів, розширювати власний світогляд і відкривати нові перспективи особистого зростання. Зявдяки неймовірній популярності мережі Інтернет, такі додатки вже давно стали чимось необхідним та природним [3].

Таким чином мета кваліфікаційної роботи полягає у розробці вебдодатку соціальної мережі з інтуїтивно зрозумілим інтерфейсом, що забезпечить

користувачів стабільним зв'язком у віртуальному середовищі. Він дозволить створювати унікальні сторінки особистого профілю, заводити нові знайомства, підтримувати контакт з друзями та рідними, обговорювати новини та важливі суспільні питання.

Для досягнення мети кваліфікаційної роботи варто виділити основні завдання:

– проаналізувати предметну область вебдодатків соціальних мереж;

– визначити актуальність розробки вебдодатку соціальної мережі «ConnectHub»;

– виявити проблеми сфери соціальних мереж;

– порівняти аналоги вебдодатку соціальної мережі «ConnectHub»;

– описати функціональні вимоги вебдодатку соціальної мережі «ConnectHub»;

– обрати технологій розробки вебдодатку соціальної мережі «ConnectHub».

– провести структурно-функціональне моделювання вебдодатку соціальної мережі «ConnectHub»;

– виконати програмну реалізацію вебдодатку соціальної мережі «ConnectHub», а саме: розробити front-end, back-end, провести тестування;

– запровадити реліз вебдодатку соціальної мережі «ConnectHub».

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ВЕБДОДАТКІВ СОЦІАЛЬНИХ МЕРЕЖ

1.1 Загальна характеристика предметної області вебдодатків соціальних мереж

На сьогоднішній день соціальні мережі є одним з найбільш вживаних ресурсів мережі Інтернет. Кількість їх користувачів уже сягає близько п’яти мільярдів осіб, і щорічно цей показник лише зростає (рис. 1.1) [4].

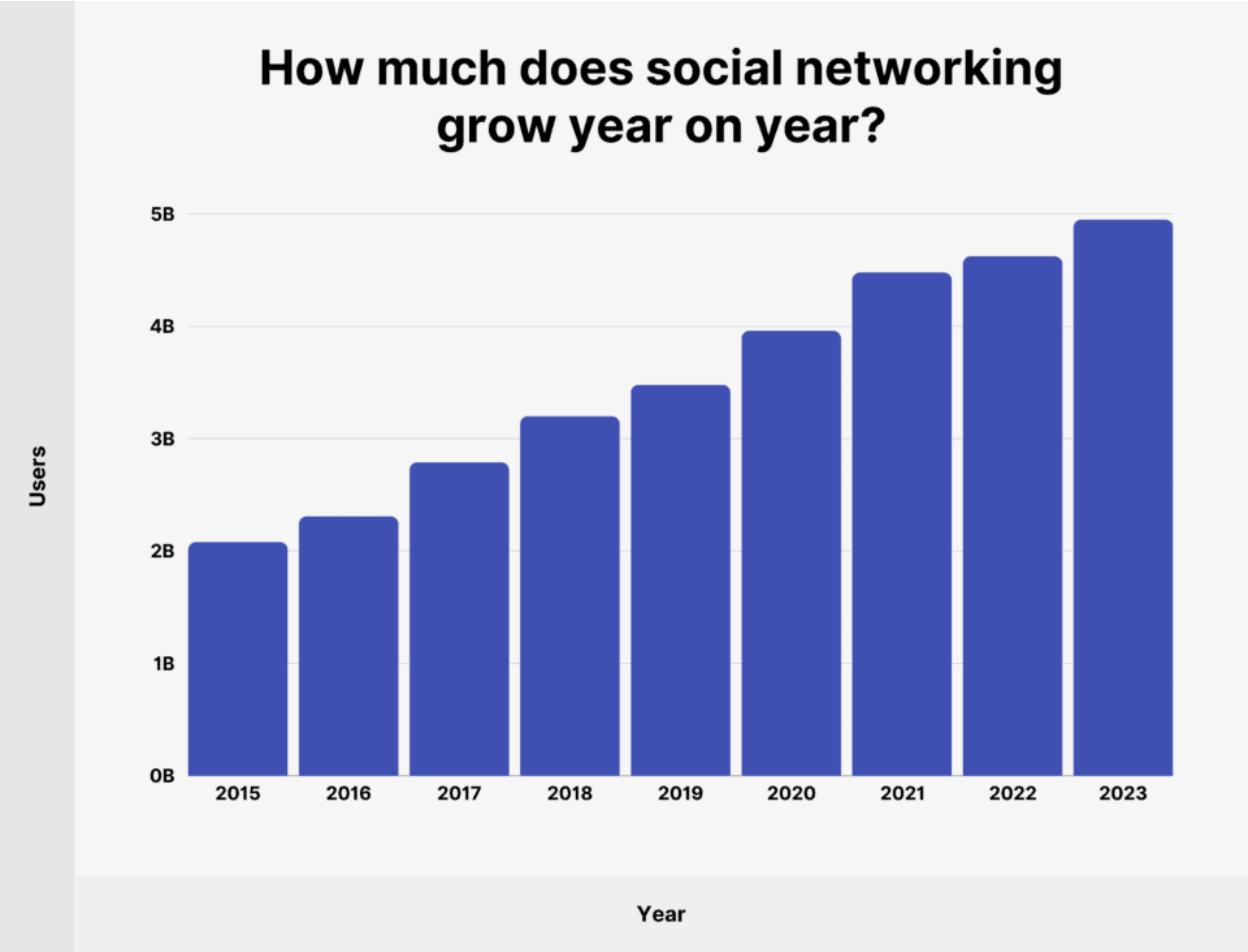


Рисунок 1.1 –Щорічне зростання користувачів соціальних мереж, млрд. осіб

Щоденно користувачі соціальних мереж у середньому проводять по декілька годин на день за улюбленим вебдодатком (рис. 1.2) [4].

Date	Time (minutes)
Q2 2021	**147**
Q3 2021	**147**
Q4 2021	**149**
Q1 2022	**149**
Q2 2022	**148**
Q3 2022	**151**
Q4 2022	**144**
Q1 2023	**146**
Q2 2023	**144**

Рисунок 1.2 – Середній час використання соціальних мереж на день, хв

Користувачі охоче створюють нові облікові записи у соціальних, оскільки середня їх кількість становить 6.7 (рис. 1.3) [4].

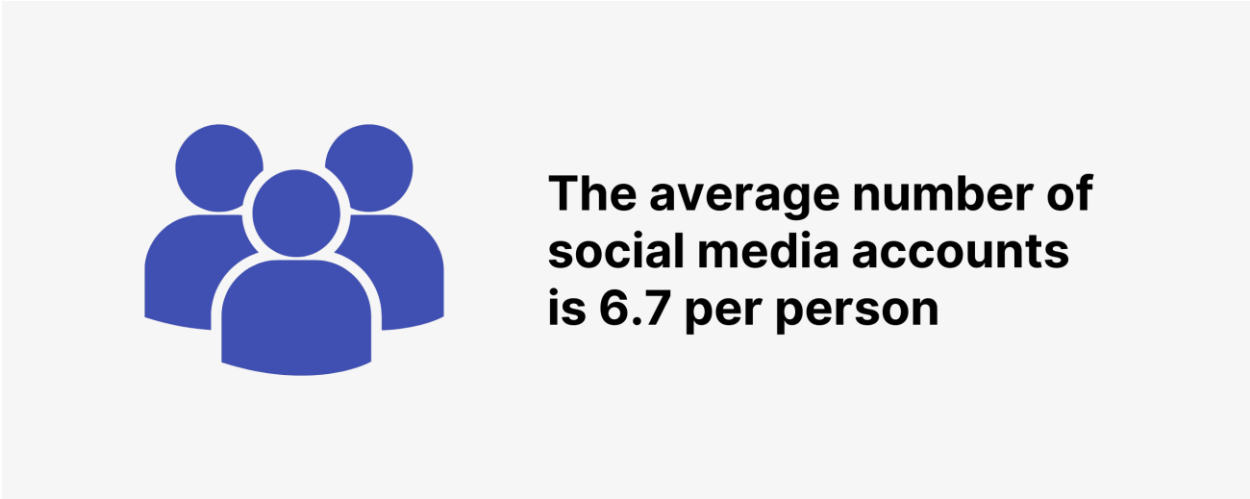


Рисунок 1.3 – Середнія кількість акаунтів соціальних мереж на особу

1.2 Проблеми предметної області вебдодатків соціальних мереж

Під час виконання роботи проведено аналіз області вебдодатків соціальних мереж та встановлено, що дана область містить ряд проблем. Перелік проблем соціальних мереж наведений у таблиці 1.1 [5].

Таблиця 1.1 – Перелік проблем соціальних мереж

№	Назва / Проблема	Зміст
1	Невідповідність існуючих соціальних мереж потребам користувачів	Існуючі соціальні мережі мають ряд недоліків, які не задовольняють потреби користувачів. Наприклад, деякі соціальні мережі не мають мобільної доступності, інші не забезпечують достатнього рівня безпеки або приватності.
2	Залежність від реклами	Багато соціальних мереж залежать від реклами як основного джерела доходу. Це може негативно позначитися на досвіді користувачів під час використання додатку.
3	Нестача інновацій	Ринок соціальних мереж стагнаційний. Існуючі соціальні мережі не пропонують нових інноваційних функцій, які могли б зацікавити користувачів.
4	Спам та фейки	Зростаюча кількість неправдивої інформації та спаму стає серйозною проблемою для соціальних мереж. Всі ці вебдодатки потребують механізми для ефективного виявлення та боротьби зі спамом, фейками та небажаним контентом.

1.3 Обґрунтування актуальності розробки вебдодатку соціальної мережі «ConnectHub»

Велика кількість людей має постійний та стабільний зв’язок з мережею Інтернет та бажає зручно та швидко взаємодіяти один з одним. Цю проблему якраз вирішують соціальні мережі.

Це є вкрай важливим, оскільки людина є соціальної істотою, яка прагне дізнаватися останні новини, реагувати на екстренні ситуації, створювати власний соціальний образ, об’єднуватись у спільноти однодумців і мати постійний контакт з близькими та друзями [6].

Це актуально зараз, адже майже кожен володіє смартфоном, що надає можливість постійного та стабільного зв’язку з мережею Інтернет. Це підключення до мережі Інтернет буде забезпечувати користувачів повноцінним доступом до соціальної мережі [7].

У результаті проведеного аналізу сфери соціальних мереж та її проблем встановлено, що є необхідність у створенні вебдодатку, який повинен мати багатофункціональний і, водночас, вичерпний інтерфейс та виконувати всі функції соціальної мережі, як-от: можливість створення унікальних сторінок профілю для користувачів, можливість обміну приватними повідомленнями між користувачами, можливість написання дописів та їх коментування [8].

1.4 Вимоги до інформаційної системи вебдодатку соціальної мережі «ConnectHub»

З метою вирішення проблем, що наведені у таблиці 1.1 визначено вимоги до розроблюваного вебдодатку. Вимоги представлені у таблиці 1.2.

Таблиця 1.2 – Перелік вимог до вебдодатку соціальної мережі «ConnectHub»

№	Назва	Зміст	Проблема, що вирішується
1	Ретельний аналіз відгуків та скарг користувачів	Адміністрація має забезпечувати безперервний зв’язок з користувачами для забезпечення найкращого досвіду для них	1
2	Незалежність від реклами	Вебдодаток повинен мати змогу функціонувати незалежно від реклами	2
3	Інноваційні функції	Команда розробки повинна тісно комунікувати з командою інновацій для створення просунутих функцій	3
4	Модерація контенту	Команда модерації має ретельно та регулярно перевіряти контент на невідповідність	4

1.5 Аналіз існуючих технологій вирішення задач проєктування вебдодатку соціальної мережі «ConnectHub»

Для розробки продукту можна використовувати популярні технології та дотримуватись сучасних методів створення вебдодатків. Full stack розробка передбачає реалізацію як front-end, так і back-end частин вебдодатку [9-10]. Варто розглянути та обрати засоби front-end розробки вебдодатку соціальної мережі «ConnectHub»:

− React.js – це відкрита бібліотека для створення інтерактивних вебдодатків. Він використовує компонентну архітектуру, яка дозволяє розробникам створювати web-сторінки з окремих компонентів. Кожний компонент відповідає за певну ділянку сторінки, наприклад, заголовок, меню або блок публікацій [11].

− Vue.js – це фреймворк для створення інтерактивних вебдодатків. Він також використовує компонентну архітектуру, але має дещо інший підхід до неї, ніж React.js. Vue.js дозволяє розробникам створювати компоненти, які можуть взаємодіяти один з одним [12].

− Angular – це фреймворк на мові програмування TypeScript для створення вебдодатків, який використовує шаблонну систему для описування інтерфейсу користувача. Angular також підтримує компонентну архітектуру, але вона дещо відрізняється від архітектури, яку використовують React.js та Vue.js [13].

Порівняльну характеристику інструментів front-end розробки наведено у таблиці 1.3.

Таблиця 1.3 – Порівняльна характеристика інструментів front-end розробки вебдодатку соціальної мережі «ConnectHub»

Показник / Можливість	**React.js**	**Vue.js**	**Angular**
Мова програмування	JavaScript (JSX)	JavaScript	TypeScript
Віртуальний DOM	Так	Так	Так
Компонентна архітектура	Так	Так	Так
Легкість вивчення	Середня	Легка	Середня
Спільнота та підтримка	Велика	Середня	Велика
Архітектурний підхід	Декларативний	Декларативний	Декларативний
Модульність	Так	Так	Так
Інтеграція з back-end	Легка	Легка	Потребує додаткових зусиль
Розширюваність	Висока	Середня	Висока
Швидкодія	Висока	Висока	Середня

Враховуючи характеристики цих інструментів, вибір React для створення front-end частини вебдодатку соціальної мережі «ConnectHub» є оптимальним, адже завдяки базуванню на мові JavaScript, великій спільноті, підтримці зі сторони розробників і досить значній можливості розширення функціоналу вебдодатку в майбутньому, він покриває всі необхідні завдання розробки front-end частини вебдодатку соціальної мережі «ConnectHub» [14-15].

1.6 Огляд існуючих програмних продуктів для вирішення поставлених задач проєктування вебдодатку соціальної мережі «ConnectHub»

Соціальні мережі вже давно є частиною нашого життя, і за весь час існування мережі Інтернет було створено незліченну кількість різноманітних web-застосунків. Деякі з них стали взірцевими, зайняли почесне місце у своїй ніші, деякі залишились менш популярними та невідомими широкому загалу, через їх вузьконаправленість, а інші навіть не пройшли відбір часом.

Важливим фактором конкурентоспроможності вебдодатку є слідування всім сучасним тенденціям розробки соціальних мереж. Для визначення вимог розроблюваного вебдодатку соціальної мережі «ConnectHub» проведено дослідження існуючих аналогів соціальних мереж, як-от: «Facebook», «Twitter» та «Instagram» [16–18].

Головна сторінка вебсайту «Facebook» (рис. 1.4) складається з навігаційного меню, яке представлене у вигляді пошукового рядка, що дозволяє знайти необхідну інформацію, п’яти окремих розділів, меню, месенджеру, сповіщень та профілю, бічної панелі керування і вмістом першого з розділів. Він представляє собою стрічку новин і дописів від інших учасників та рекомендованого контенту від платформи.

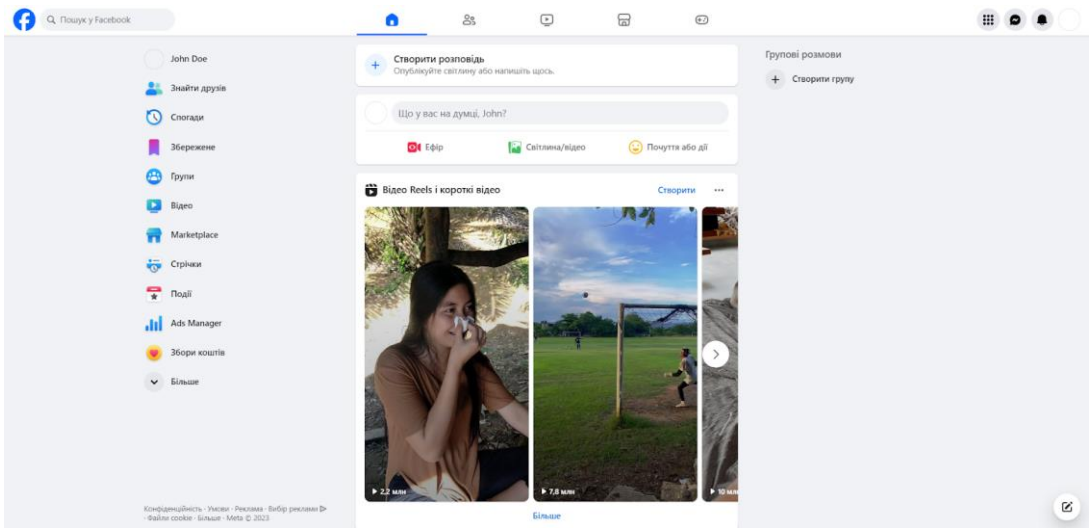


Рисунок. 1.4 – Головна сторінка «Facebook»

Наступний розділ – «Друзі» (рис. 1.5). Виконаний у більш мінімалістичному стилі і містить всю необхідну інформацію для взаємодії з друзями.

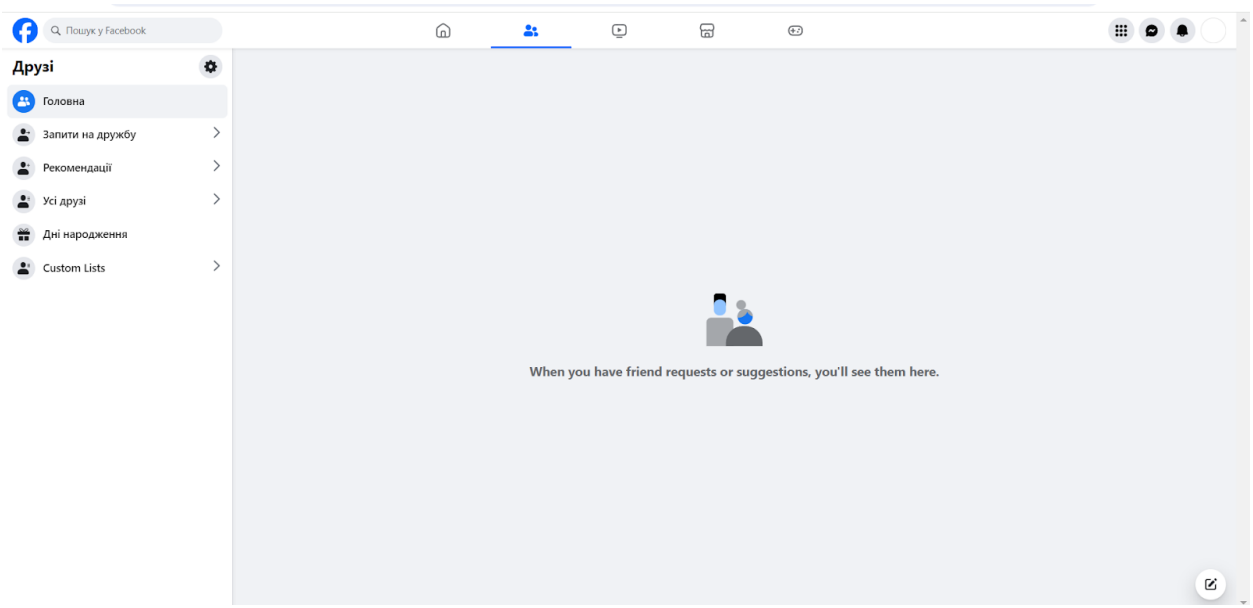


Рисунок. 1.5 – Другий розділ головної сторінки «Facebook»

Третій розділ – «Відео» (рис. 1.6). Також представлений у компактному стилі та є досить зручним та зрозумілим.

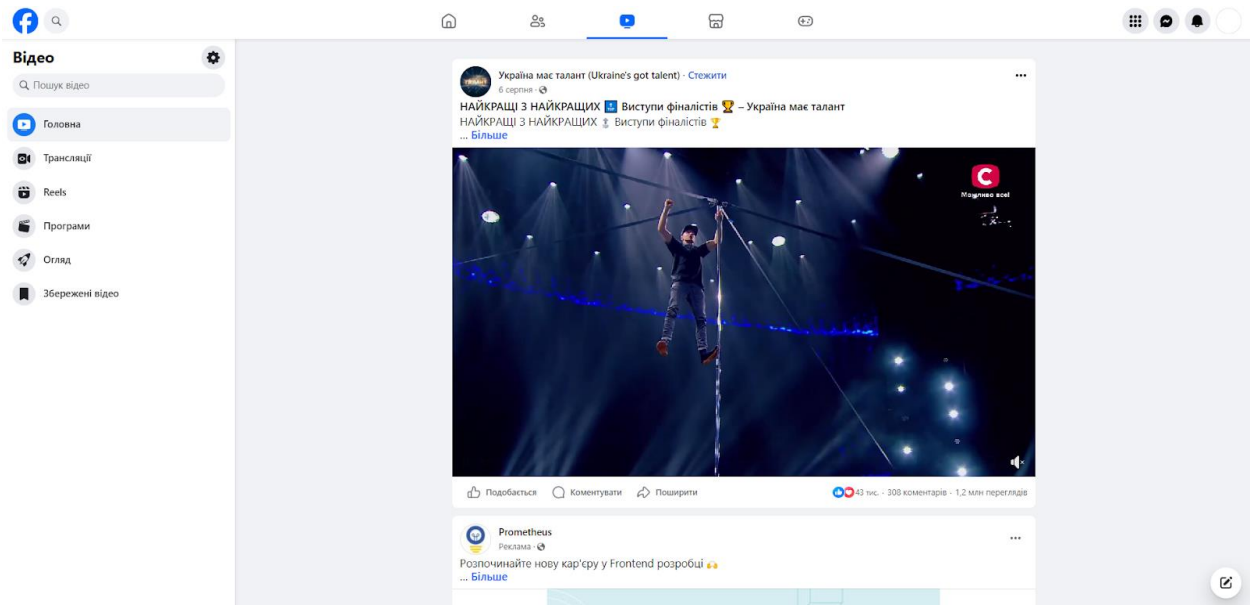


Рисунок. 1.6 – Третій розділ головної сторінки «Facebook»

Четвертий розділ – «Marketplace» (рис. 1.7). Сторінка для пошуку, продажу та покупки вживаних товарів.

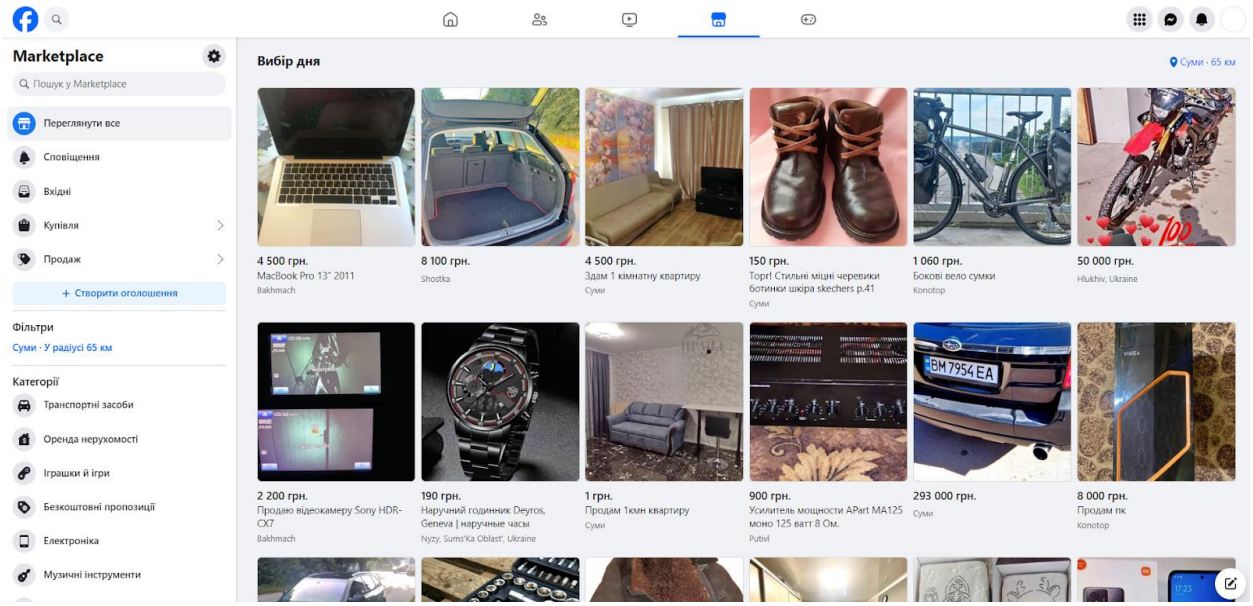


Рисунок. 1.7 – Четвертий розділ головної сторінки «Facebook»

Останній розділ = «Ігри» (рис. 1.8) являє собою набір різноманітних інді-ігор.

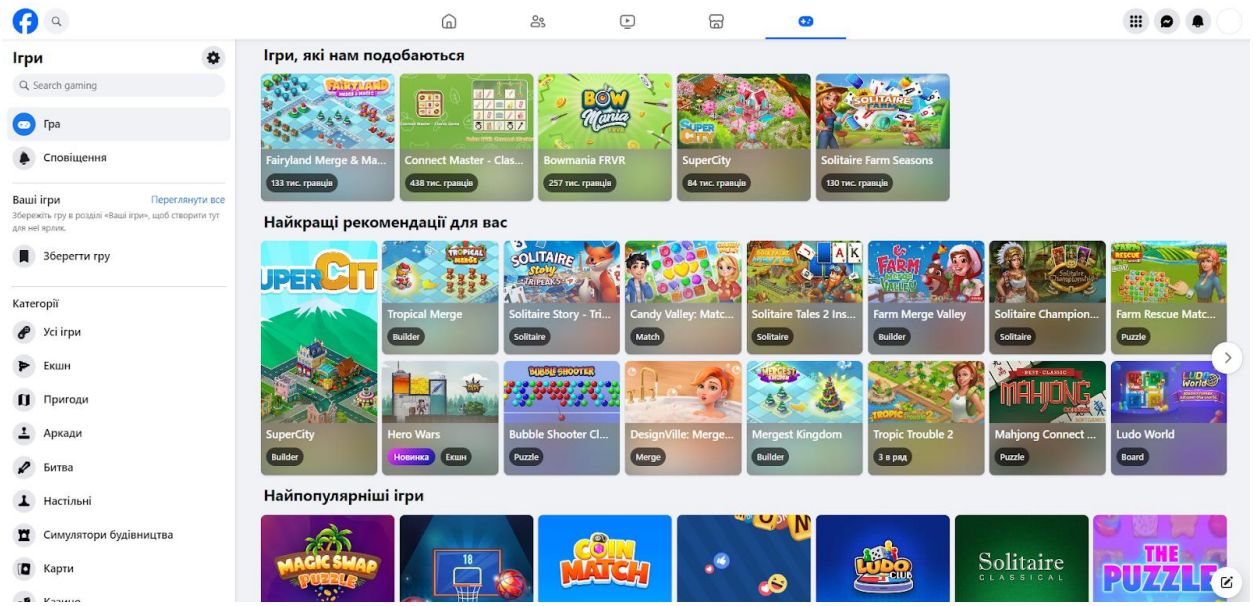


Рисунок. 1.8 – П’ятий розділ головної сторінки «Facebook»

Сторінка користувача (рис. 1.9) надає всю необхідну інформацію про нього, дає можливість редагувати дані про себе, додавати аватар та банер профілю, створювати дописи та ділитися різноманітним контентом.

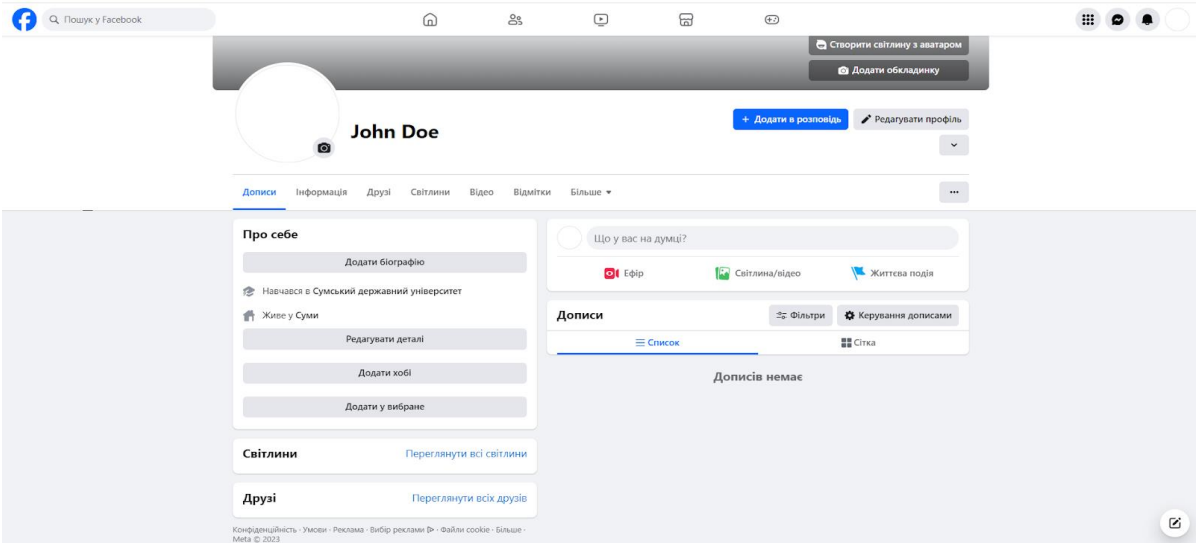


Рисунок. 1.9 – Сторінка користувача «Facebook»

Платформа також надає можливість виконувати налаштування системи, якщо перейти до відповідної сторінки (рис. 1.10). Цей розділ містить опції зміни конфіденційності користувача та мови інтерфейсу, керування сповіщеннями, підключення різноманітних додатків та сайтів й інтеграції бізнесу.

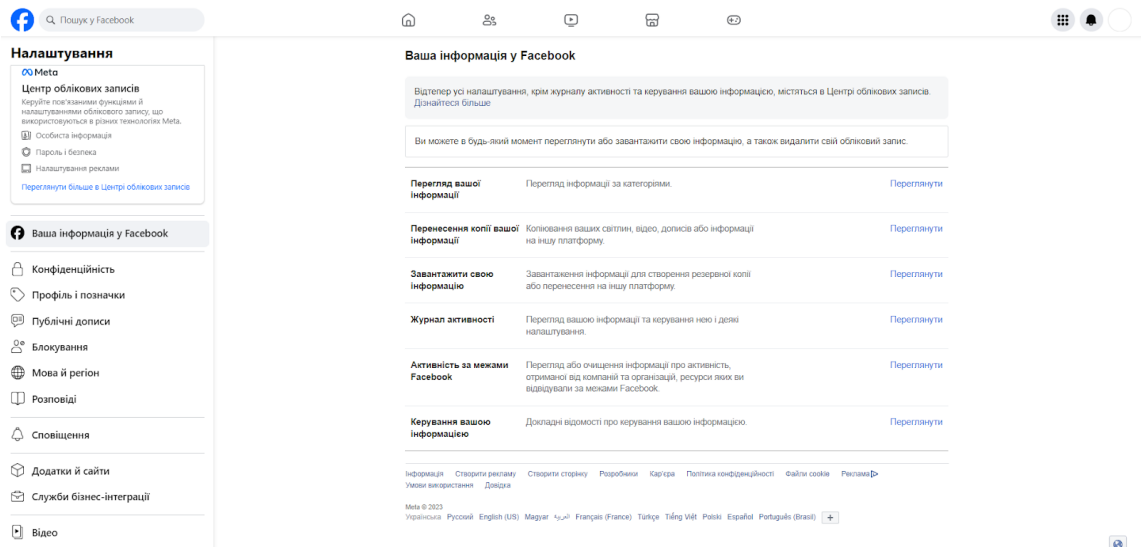


Рисунок. 1.10 – Сторінка налаштувань «Facebook»

В цілому, дизайн сайту «Facebook» приємний, виконаний у єдиному стилі з використанням білого, синього та сірого кольорів. Поряд з назвами розділів зображені відповідні значки, що полегшує навігацію і дозволяє швидко та інтуїтивно перемикатися між необхідними вкладками. Головна сторінка містить аж надто багато інформації, що створює враження перевантаження. Інші розділи виконано більш мінімалістично, вони відповідають лише за одну єдину функцію і не містять нічого зайвого. Варто зауважити, що така кількість сторінок є досить цікавим рішенням компанії охопити якомога ширший простір розваг та залучити більше різноманітних користувачів. Також завдяки гнучкій системі налаштувань користувач може керувати параметрами взаємодії з платформою за власним бажанням [19].

Головна сторінка соціальної мережі «Instagram» (рис. 1.11) має бічне навігаційне меню, стрічку новин та дописів від інших користувачів, список рекомендацій, що представляє собою потенційно цікаві акаунти для знайомства.

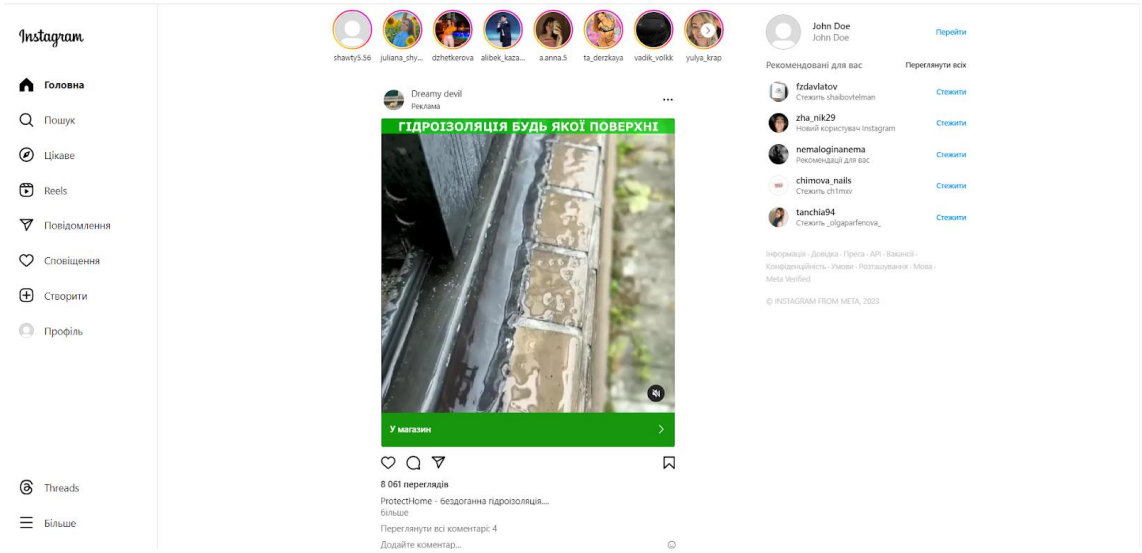


Рисунок. 1.11 – Головна сторінка «Instagram»

Сторінка користувача (рис. 1.12) містить базову інформацію про власника, кількість його дописів, читачів та підписок, можливість редагувати дані про себе та розділ зі світлинами й дописами.

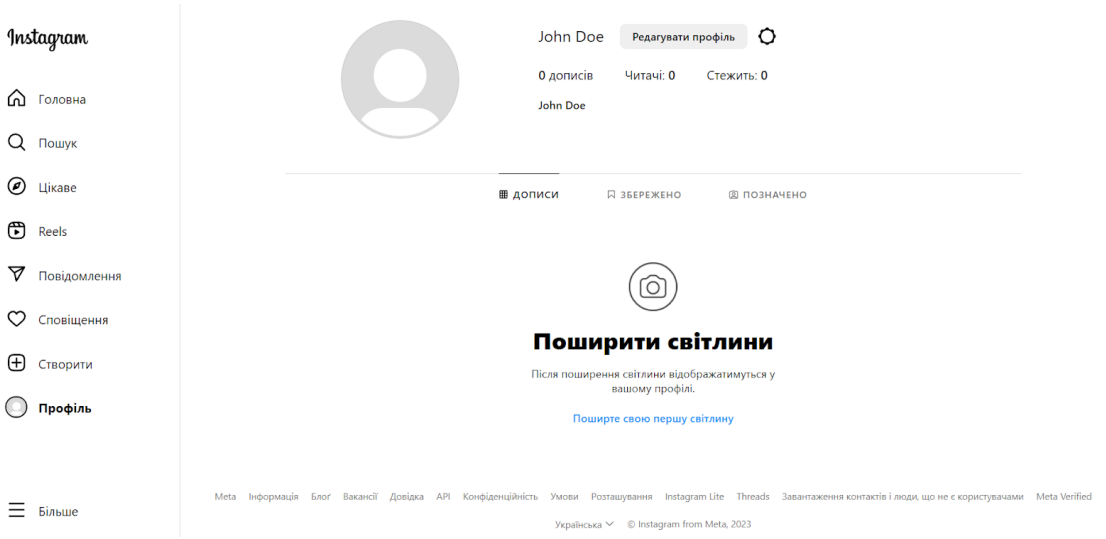


Рисунок. 1.12 – Сторінка користувача «Instagram»

Сторінка налаштувань (рис. 1.13) дозволяє редагувати профіль користувача, конфіденційність інформації про акаунт, керувати email- та push-сповіщеннями і містить довідку з сервісом підтримки для вирішення різноманітних технічних проблем.

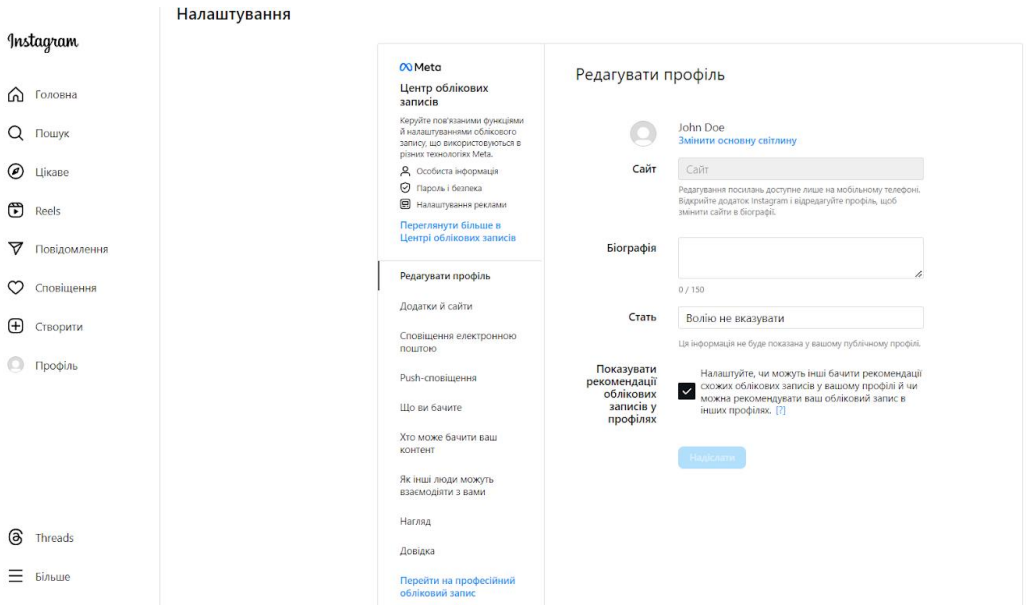


Рисунок. 1.13 – Сторінка налаштувань «Instagram»

Можна сказати, що платформа виглядає доволі сучасно та мінімалістично, адже має витриманий чорно-білий стиль, зручний інтерфейс з відповідними іконками та надає весь необхідний функціонал для користування [20].

Головна сторінка соціальної мережі «Twitter» (рис. 1.14) має бічне навігаційне меню, стрічку новин та постів від інших користувачів, список рекомендованих акаунтів, що підбираються в залежності від власних інтересів, місця перебування та вподобань друзів. Також є цікава панель з трендами, що набирають популярність.

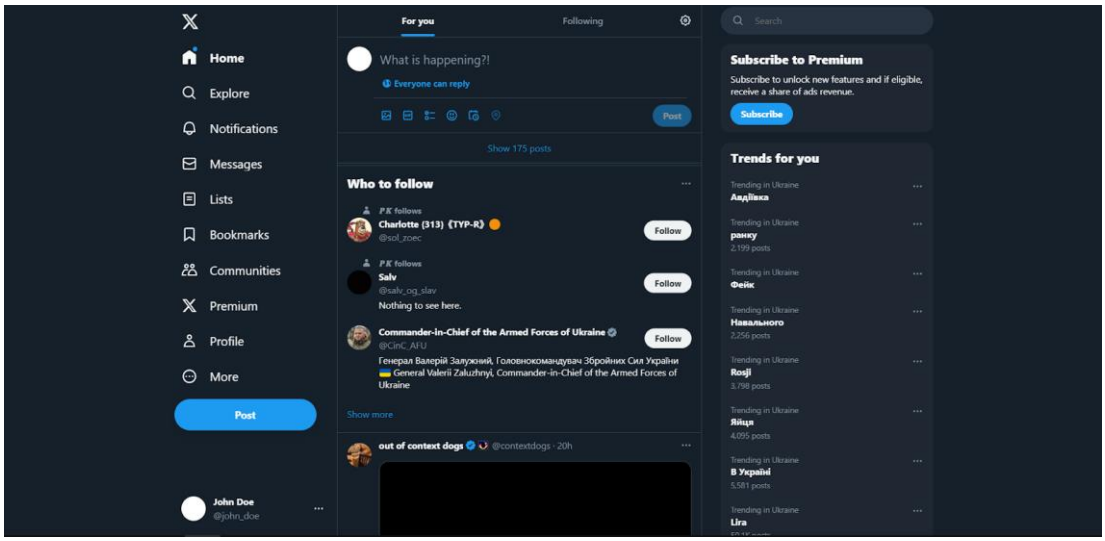


Рисунок. 1.14 – Головна сторінка «Twitter»

Сторінка користувача (рис. 1.15) містить необхідні дані про власника, кількість його підписників та підписок, розділ з постами, ретвітами та коментарями і надає можливість редагувати інформацію про себе.

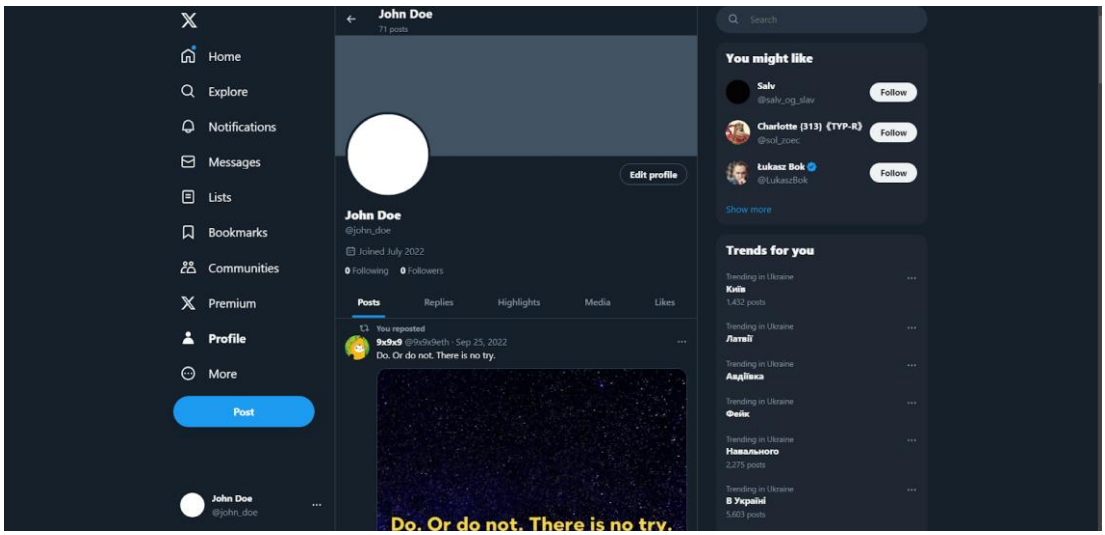


Рисунок. 1.15 – Сторінка користувача «Twitter»

Сторінка налаштувань (рис. 1.16) дозволяє редагувати профіль користувача, оформлювати Premium-підписку, змінювати параметри конфіденційності, обирати варіанти надходження сповіщень, змінювати мову, головну тему платформи та переглядати довідкову інформацію.

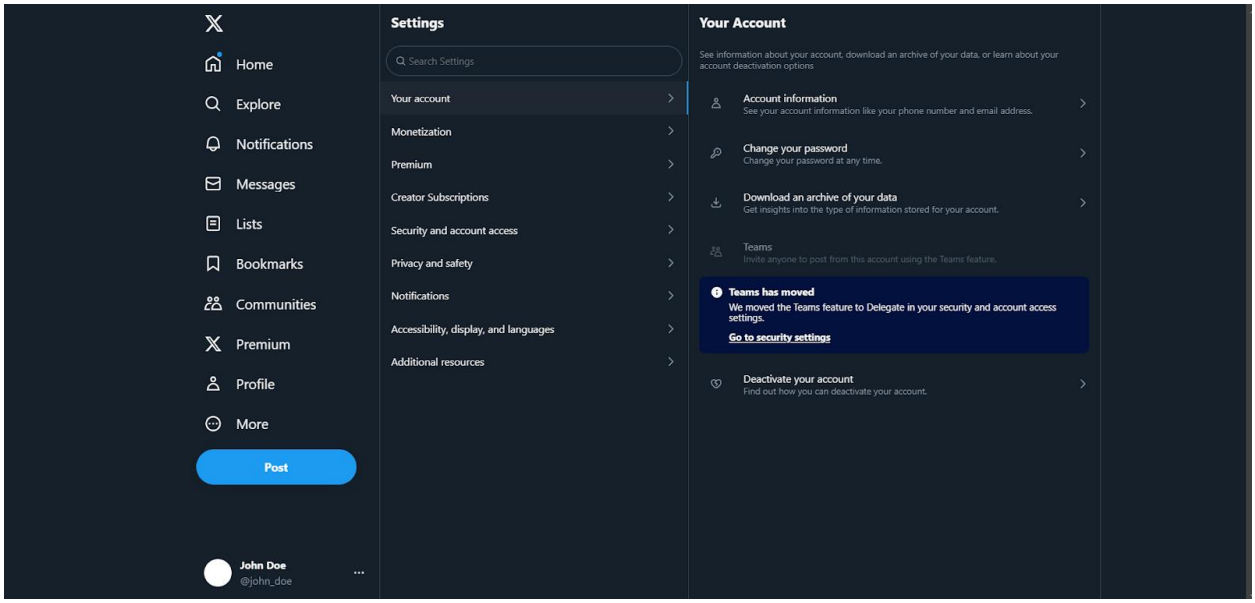


Рисунок. 1.16 – Сторінка налаштувань «Twitter»

Вцілому, вебзастосунок виконаний у досить мінімалістичному стилі, він виглядає досить сучасно, містить весь необхідний функціонал соціальної мережі, надає різноманітні рекомендації та пропонує гнучку систему налаштувань [21].

Після детального аналізу соціальних мереж описано їх порівняльну характеристику. Результати представлені в таблиці 1.4.

Дані з таблиці 1.4 дозволяють проаналізувати кожну з платформ, їх сильні сторони, які можна впровадити у вебдодаток, та слабкості, яких варто уникати. Розроблюваний вебдодаток соціальної мережі «ConnectHub» має наслідувати мережу «Twitter»: мати сучасний дизайн, в міру наповнений інтерфейс, зрозумілу та функціональну навігаційну панель. Також, очевидно, давати можливість реєструватися й авторизуватися, заводити знайомства, спілкуватися приватно, слідкувати за вподобаннями й підписками інших та створювати дописи з використанням різноманітних типів файлів.

Таблиця 1.4 – Порівняльна характеристика аналогів вебдодатку соціальної мережі «ConnectHub»

Особливість	**Соціальна мережа**		
	«Facebook»	**«Instagram»**	**«Twitter»**
Зручність інтерфейсу	+-	+	+
Сучасність дизайну	+-	+	+
Функціональність навігаційної панелі	+	+	+
Функціональність сторінки профілю	+	+	+
Універсальність	+	+-	-
Налаштування вигляду вебдодатку	+-	-	+
Можливість контакту з адміністрацією вебдодатку	-	-	-
Залежність від реклами	+	+	+

1.7 Постановка задачі вебдодатку соціальної мережі «ConnectHub»

Метою роботи є розробка вебдодатку соціальної мережі з інтуїтивно зрозумілим інтерфейсом, що забезпечить користувачів стабільним зв'язком у віртуальному середовищі. Він дозволить створювати унікальні сторінки особистого профілю, заводити нові знайомства, підтримувати контакт з друзями та рідними, обговорювати новини та важливі суспільні питання.

Для досягнення мети проєкту варто виділити задачі дипломної роботи:

– дослідити область вебдодатків соціальних мереж, виявити поточні проблеми досліджуваної предметної області та визначити актуальність роботи;

– провести аналіз аналогів вебдодатку соціальної мережі «ConnectHub» і виділити їх переваги та недоліки;

– описати функціональні вимоги до розроблюваного вебдодатку соціальної мережі «ConnectHub»;

– обрати необхідні технології для реалізації вебдодатку соціальної мережі «ConnectHub»;

– провести структурно-функціональне моделювання вебдодатку соціальної мережі «ConnectHub»;

– виконати програмну реалізацію вебдодатку соціальної мережі «ConnectHub», а саме: розробити front-end, back-end, провести тестування;

– запровадити реліз вебдодатку соціальної мережі «ConnectHub».

Основні вимоги до створюваного вебдодатку соціальної мережі «ConnectHub» є наступними:

– можливість реєстрації та авторизації;

– можливість перегляду наповнення головної сторінки та профілів користувачів;

– можливість редагування особистого профілю;

– можливість створення та видалення дописів;

– можливість створення та видалення коментарів під дописами користувачів;

– можливість заводити друзів серед інших користувачів;

– можливість відправляти приватні повідомлення друзям;

– можливість змінювати налаштування вебдодатку;

– можливість відправляти форми зворотнього зв’язку до адміністрації вебдодатку;

– наявність панелі адміністратора для додавання, редагування та видалення наповнення вебдодатку;

– можливість відповідати на форми зворотнього зв’язку для адміністратора.

Використання вебдодатку соціальної мережі «ConnectHub» необмежене по території, але потребує від користувачів знань англійської для комфортної роботи з інтерфейсом.

Технічне завдання на розробку вебдодатку соціальної мережі «ConnectHub» у повному обсязі наведено у додатку А.

1.8 Вибір та обґрунтування засобів реалізації вебдодатку соціальної мережі «ConnectHub»

Для створення вебдодатку соціальної мережі «ConnectHub» обрано технології, як-от: мову розмітки гіпертексту HTML для створення каркасу web-сторінок, каскадні таблиці стилів CSS для надання web-сторінкам візуальних ефектів та адаптивності, front-end бібліотеку React на основі мови програмування JavaScript для надання динамічності та асинхронної взаємодії з серверною частиною, фреймворк Express на основі Node.js для створення back-end частини вебдодатку та Webpack – інструмент для об’єднання окремих модулів (файлів) у один. Для реалізації бази даних розроблюваного вебдодатку соціальної мережі «ConnectHub» обрано СУБД MySQL.

1.9 Висновки по першому розділу

Виконано аналіз сфери соціальних мереж і отримано загальну характеристику цієї предметної області, виявлено її наявні проблеми, обґрунтовано актуальність розробки вебдодатку соціальної мережі «ConnectHub», перелічено системні вимоги інформаційної системи вебдодатку соціальної мережі «ConnectHub», поставлено задачі та мету проєкту, проаналізовано існуючі технології для вирішення задач проєктування вебдодатку соціальної мережі «ConnectHub» і обрано найращий інструмент для розробки вебдодатку, порівняно існуючі аналоги вебдодатку соціальної мережі «ConnectHub» і виявлено найращий, що слугуватиме орієнтиром для розроблюваного вебдодатку.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ВЕБДОДАТКУ СОЦІАЛЬНОЇ МЕРЕЖІ «CONNECTHUB»

2.1 Структурно-функціональне моделювання інформаційної системи вебдодатку соціальної мережі «ConnectHub»

IDEF0 (Integration Definition for Function Modeling) – це методологія функціонального моделювання, яка використовується для створення функціональних моделей систем, включаючи їхню структуру, функції та потоки інформації і матеріальних об'єктів. Вона може бути використана як для розробки нових систем, так і для аналізу вже існуючих.

Функціональне моделювання вебдодатку соціальної мережі «ConnectHub» в нотації IDEF0 представлене на рисунку 3.1.

Рисунок 2.1 – Діаграма IDEF0 вебдодатку соціальної мережі «ConnectHub»

Як показано на рис. 3.1 головною функцією є соціальна взаємодія за допомогою вебдодатку соціальної мережі «ConnectHub».

– Вхідні дані – облікові дані користувача, його бажані тексти допису, коментаря та приватного повідомлення.

– Управління – це правила, згідно з якими здійснюється автентифікація, політика конфіденційності, правила використання додатку, перелік користувачів вебдодатку, перелік друзів користувача та перелік дописів вебдодатку.

– Механізмами у даній діаграмі є: Інтернет, користувач, апаратне забезпечення, вебдодаток соціальної мережі «ConnectHub», вебсервер та база даних.

– Виходами є опублікований допис, коментар, відправлене приватне повідомлення та відправлений запит на дружбу.

Декомпозиція IDEF0 системи включає розгляд функції системи на більш детальному рівні. Оскільки кожен блок є певною функцією, декомпозиція досягається шляхом поділу такого блоку на його менші структурні частини.

Декомпозиція функціональної моделі вебдодатку соціальної мережі «ConnectHub» представлена на рисунку 2.2.

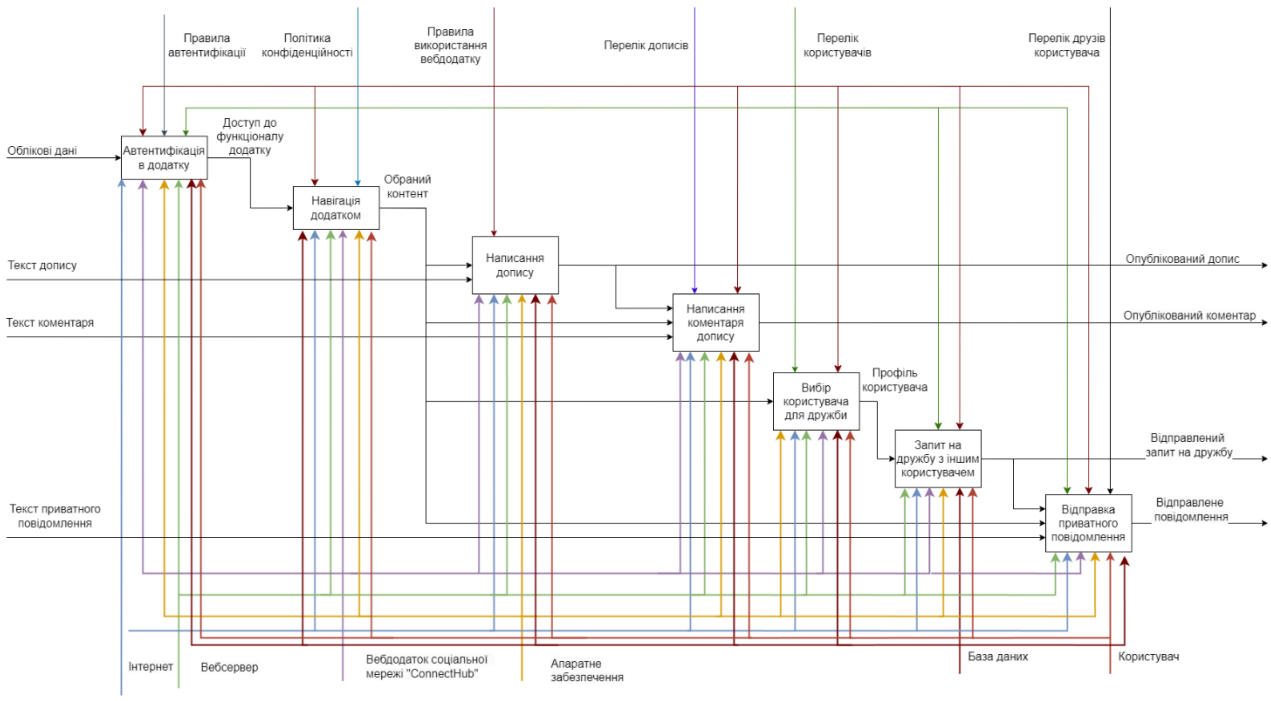


Рисунок 2.2 – Декомпозиція IDEF0 вебдодатку соціальної мережі «ConnectHub»

Згідно з рис. 2.2, можна виділити наступні блоки функціонування:

– До вебдодатку соціальної мережі «ConnectHub» мають надійти облікові дані користувача, аби здійснити автентифікацію;

– Після цього користувачу відкривається доступ до навігації по вебдодатку;

– Обравши бажаний контент, користувач може здійснити з ним взаємодію, як-от: опублікувати допис, прокоментувати допис, відшукати користувача для дружби, відправити запит на дружбу та приватне повідомлення другу;

– Публікація допису – користувач створює власний допис, у якому може висловити свої думки або ж поділитися новинами;

– Коментування допису – користувач коментує власний або допис іншого користувача, ділиться враженнями та обговорює вміст публікацій;

– Вибір користувача для дружби здійснюється безпосередньо через навігацію по вебдодатку та пошук друзів серед авторів дописві;

– Запит на дружбу з іншим користувачем – користувач відправляє запит на дружбу з будь-яким іншим користувачем;

– Відправка приватного повідомлення другу – користувач має змогу приватної бесіди з одним зі своїх друзів.

2.2 Проєктування інформаційної системи вебдодатку соціальної мережі «ConnectHub»

Для досягнення цілей функціонування спочатку будується модель у формі діаграми варіантів використання (use-case diagram), яка описує функціональне призначення системи. Діаграма варіантів використання являє собою вищий рівень концептуальної моделі системи і не намагається описати низькорівневі процеси.

Важливою вимогою також є побудова діаграми з точки зору користувачів системи (акторів), а не розробника. Перелік акторів системи наведено нижче:

– Відвідувач – особа, що вперше відвідує вебдодаток або ще не реєструвалась на ньому;

– Користувач – особа, яка створила обліковий запис на сайті вебдодатку;

– Адміністратор – особа, що володіє правами доступу до адміністративної панелі.

Діаграма варіантів використання вебдодатку соціальної мережі «ConnectHub» в нотації UML представлена на рисунках 2.3-2.5.

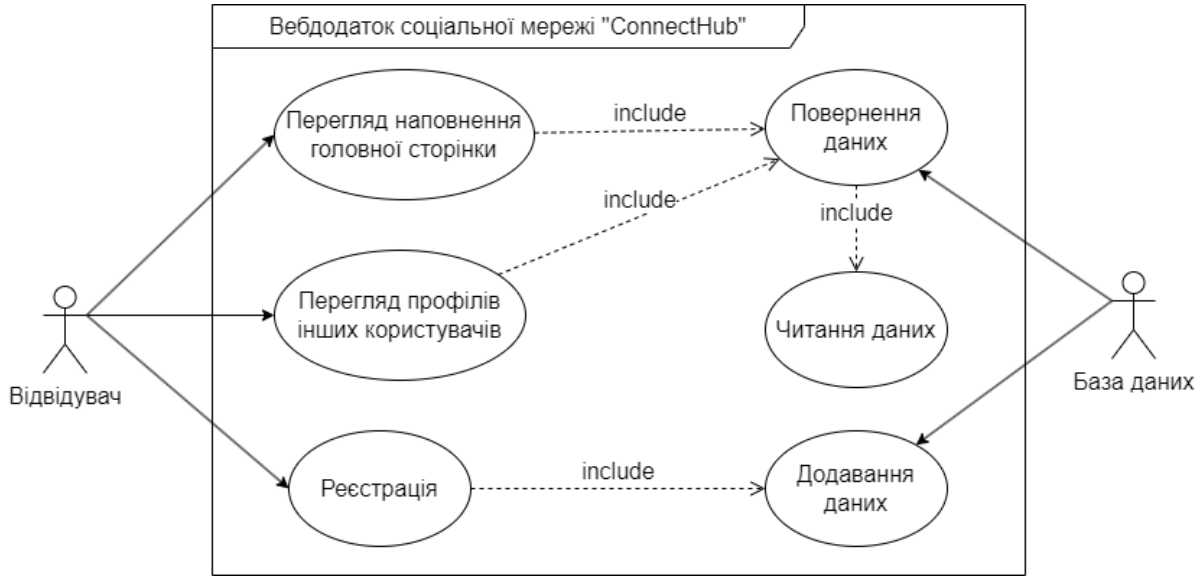


Рисунок 2.3 – Діаграма варіантів використання вебдодатку соціальної мережі «ConnectHub» . Частина 1

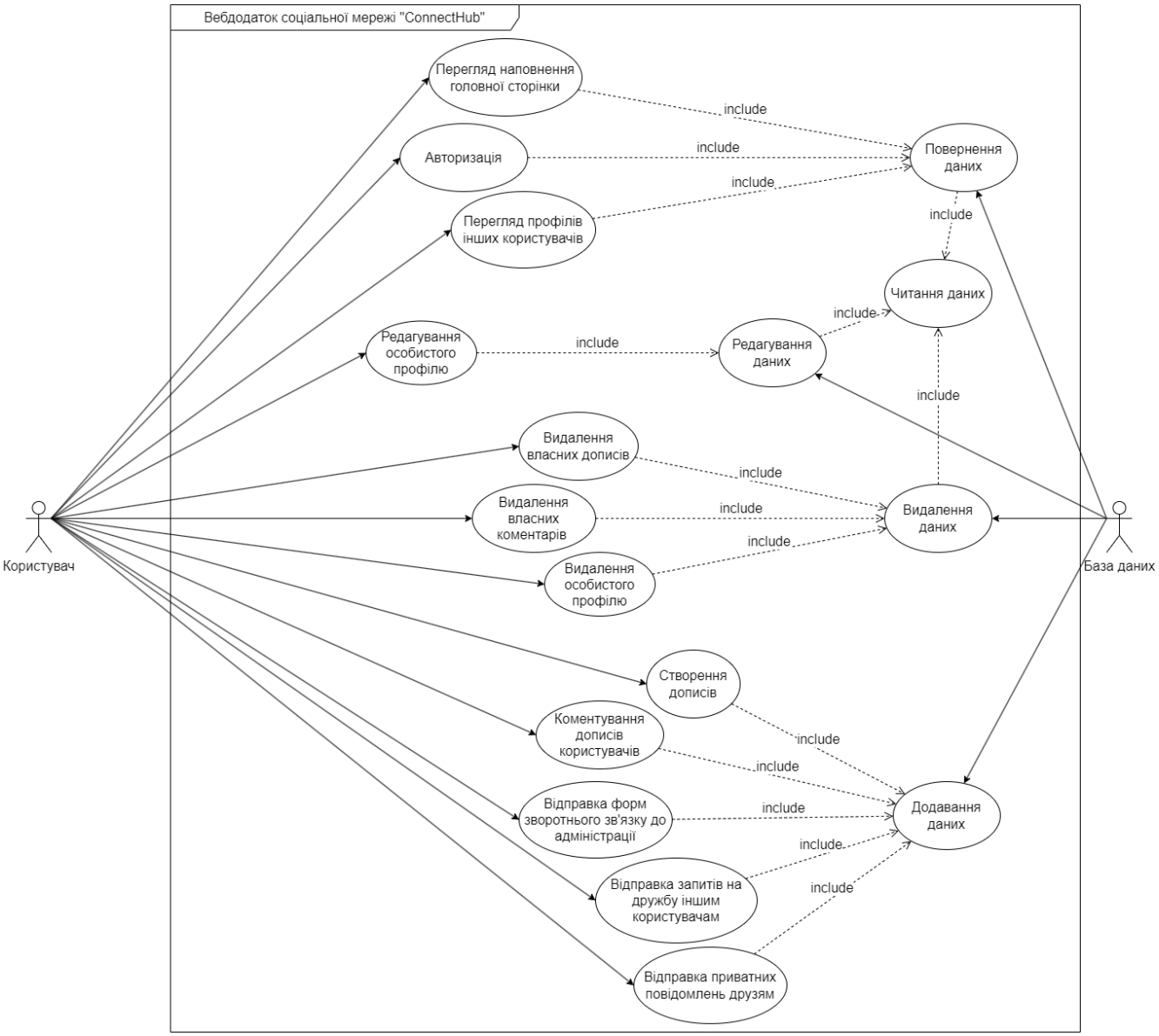


Рисунок 2.4 – Діаграма варіантів використання вебдодатку соціальної мережі «ConnectHub» . Частина 2

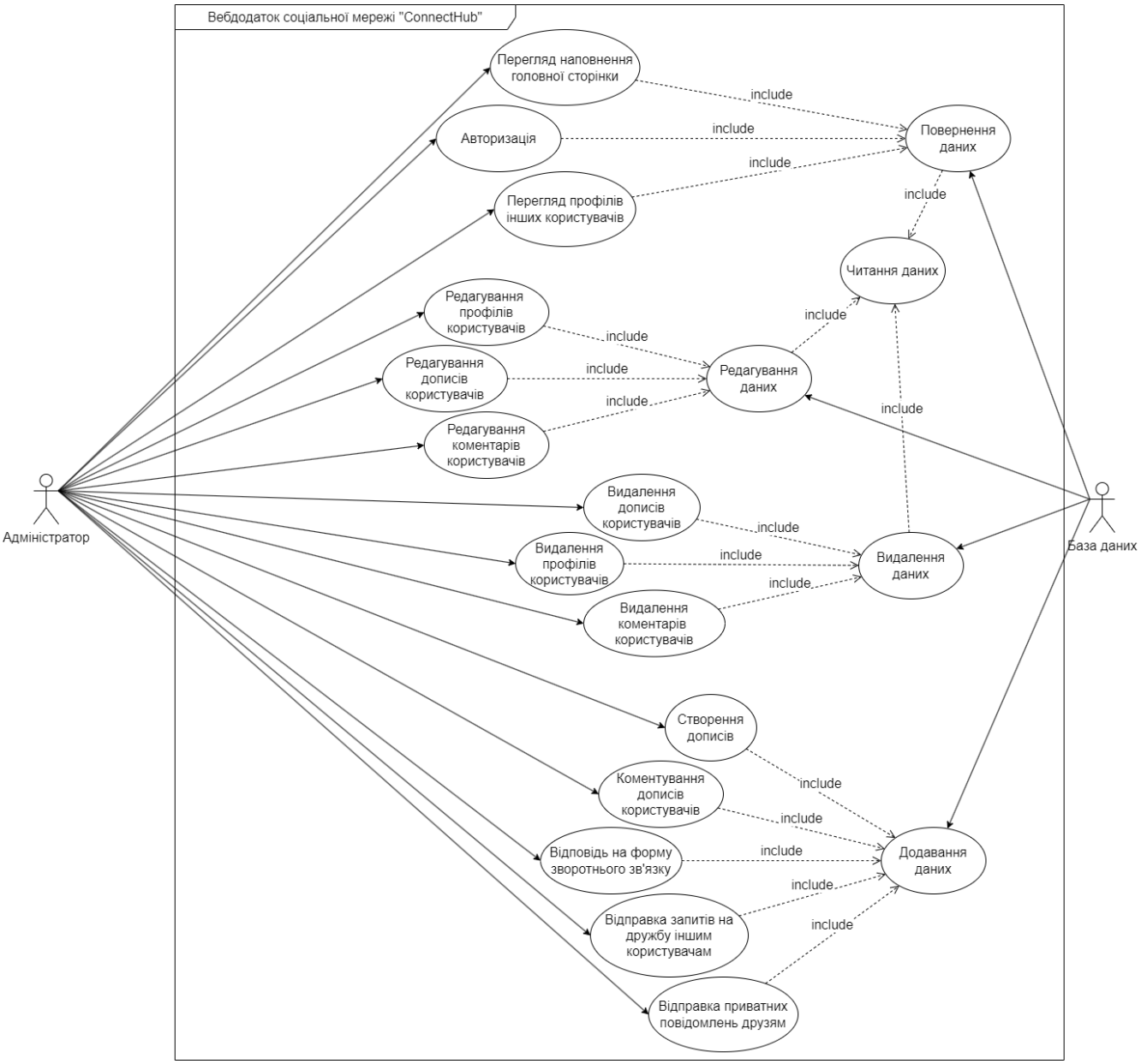


Рисунок 2.5 – Діаграма варіантів використання вебдодатку соціальної мережі «ConnectHub» . Частина 3

На рисунках вище відображено варіанти використання вебдодатку соціальної мережі «ConnectHub» в нотації UML. Вона включає 4 актори: Відвідувач, Користувач, Адміністратор та База даних. Кожен актор має відповідні варіанти використання.

2.3 Проєктування моделі бази даних вебдодатку соціальної мережі «ConnectHub»

2.3.1 Діаграма потоків даних

Для побудови моделі бази даних необхідно спочатку реалізувати структуру потоків даних, або так звану DFD-діаграму.

Діаграма потоків даних (DFD) – це графічний метод моделювання потоків даних всередині системи. Вона використовується для візуалізації обробки даних та взаємодії різних компонентів системи. DFD дозволяє розкрити, як вхідні дані проходять через різні процеси.

Діаграма потоків даних вебдодатку соціальної мережі «ConnectHub» на концептуальному рівні представлена на рисунку 2.6. Назва інформаційної системи – «Вебдодаток соціальної мережі «ConnectHub»». Виконавцями системи є клієнт та адміністратор вебдодатку.

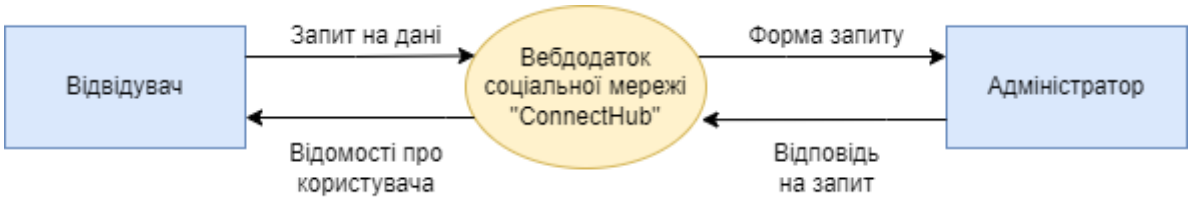


Рисунок 2.6 – Діаграма потоків даних DFD вебдодатку соціальної мережі «ConnectHub»

Діаграма потоків даних на нульовому рівні декомпозиції представлена на рисунку 2.7. Актори системи взаємодіють між собою за допомогою потоків даних та процесів, що виникають між ними. У ході розробки діаграми було створено сховище даних – «Користувачі».

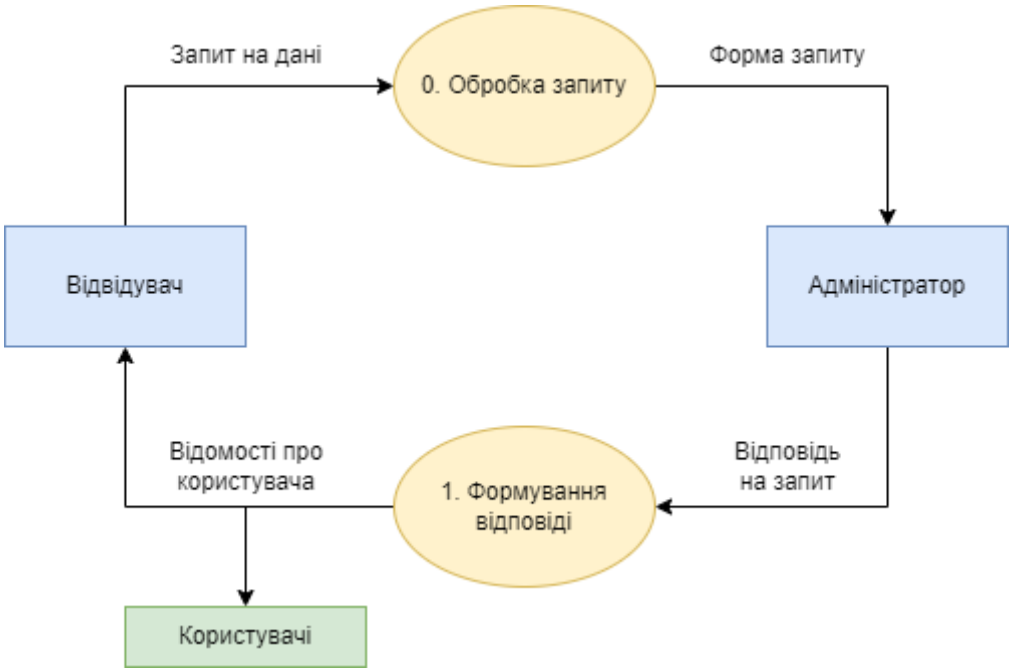


Рисунок 2.7 – Діаграма потоків даних DFD-0 вебдодатку соціальної мережі «ConnectHub»

2.3.2 Логічна модель бази даних

Логічна модель даних – це вищий рівень абстракції в моделюванні даних, який визначає структуру даних та їх взаємозв'язки без прив'язки до конкретної бази даних чи системи управління базами даних. Цей рівень моделювання зорієнтований на бізнес-процеси та логічні відносини між елементами даних.

Логічна модель даних допомагає створити уніфіковане розуміння структури та взаємозв'язків даних у системі, що полегшує розробку бази даних на наступному етапі.

На рисунку 2.8 представлена діаграма потоків даних вебдодатку соціальної мережі «ConnectHub».

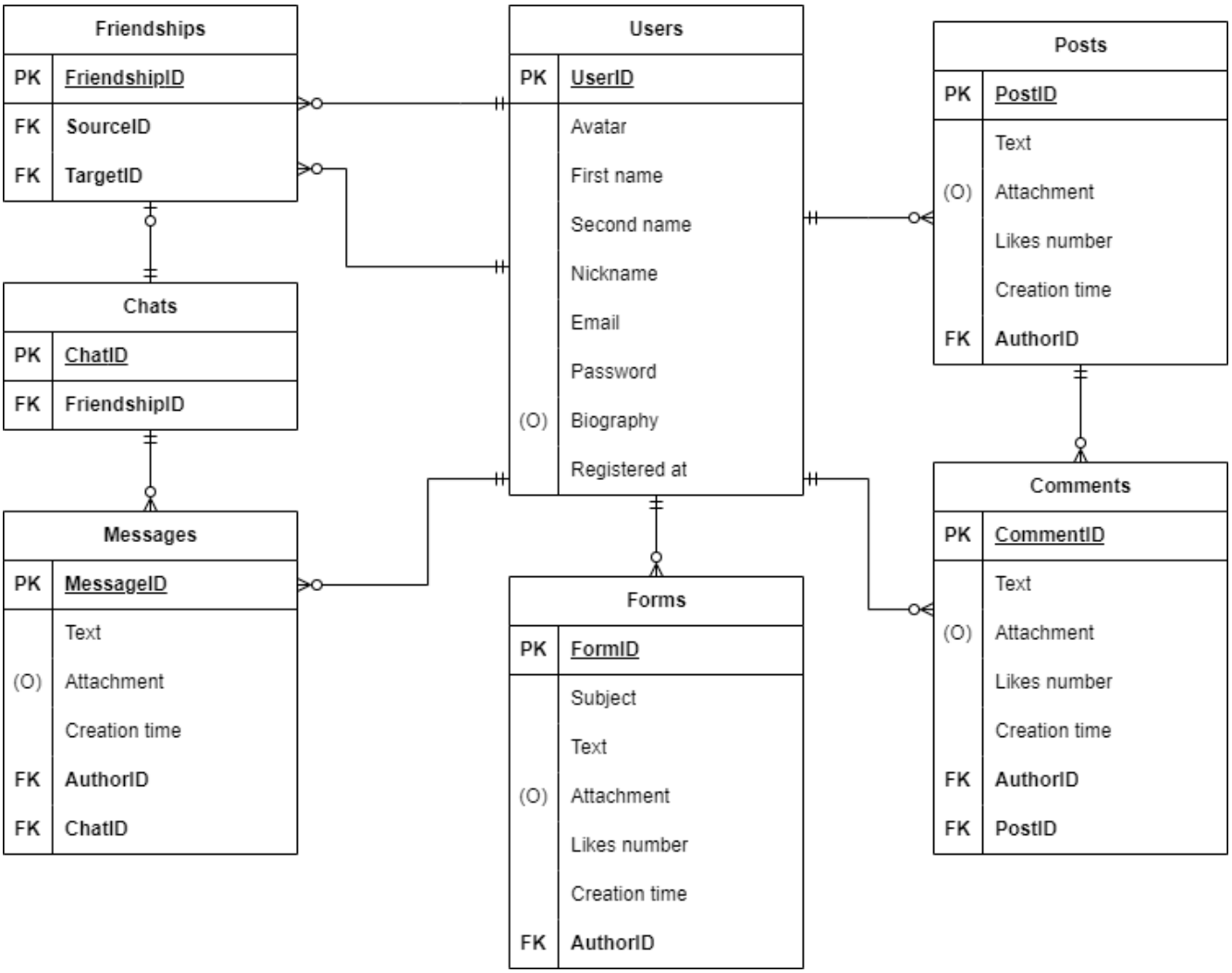


Рисунок 2.8– Логічна модель даних вебдодатку соціальної мережі «ConnectHub»

Згідно з рисунком 2.8 майбутня БД буде мати наступні таблиці: Users, Posts, Comments, Friends та Messages. Таблиці бази даних:

– Користувачі обов’язково повинні мати електронну пошту, пароль, нікнейм, ім’я та прізвище для створення власного профілю, у той час як інші поля є опціональними.

– Дописи містять певний текст, дату публікації, кількість вподобань та необов’язкове поле вкладених файлів.

– Коментарі мають певний текст та автоматично встановлювані дату коментування, ідентифікаційний номер автора коментаря та автора допису, кількість вподобань та можливі вкладення файлів.

– Дружби містять ідентифікаційні номери відправника та цільового користувача запиту на дружбу.

– Чати створюються для певних пар друзів і необхідні для зберігання повідомлень.

– Повідомлення мають ідентифікаційні номери автора та відповідного чату, певний текст, опціональне вкладення файлу та дату відправки.

2.3.3 Фізична модель бази даних

Проаналізувавши сутності, використовувані в моделі інформаційної системи, можна перейти до реалізації структури БД. Для цього варто представити імена необхідних таблиць, їх атрибути, типи, призначення та обмеження (табл. 2.1).

Таблиця 2.1 – Фізична модель бази даних

Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
users	userid	Унікальний ідентифікатор користувача	INTEGER	**PK**	Не пустий
	email	Електронна пошта користувача	VARCHAR(50)		Не пустий, Унікальний
	password	Пароль облікового запису користувача	VARCHAR(100)		Не пустий
	nickname	Нікнейм облікового запису користувача	VARCHAR(50)		Не пустий, Унікальний
	avatar	Аватар облікового запису користувача	VARCHAR(200)		Не пустий
	registration_time	Час реєстрації користувача	TIMESTAMP		Не пустий
	first_name	Ім’я користувача	VARCHAR(50)		Не пустий
	second_name	Прізвище користувача	VARCHAR(50)		Не пустий
	biography	Опис облікового запису користувача	VARCHAR(200)		
posts	post_id	Унікальний ідентифікатор допису	INTEGER	**PK**	Не пустий
	text	Текст допису	VARCHAR(200)		Не пустий
	attachment	Файл допису	VARCHAR(200)		
	creation_time	Час публікації допису	TIMESTAMP		Не пустий
	likes_number	Кількість вподобань допису	INTEGER		Не пустий
	author_id	Унікальний ідентифікатор автора допису	INTEGER	**FK**	Не пустий

Продовження таблиці 2.1

Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
comments	comment_id	Унікальний ідентифікатор коментаря	INTEGER	**PK**	Не пустий
	text	Текст коментаря	VARCHAR(200)		Не пустий
	attachment	Файл коментаря	VARCHAR(200)		
	creation_time	Час публікації коментаря	TIMESTAMP		Не пустий
	likes_number	Кількість вподобань коментаря	INTEGER		Не пустий
	author_id	Унікальний ідентифікатор автора коментаря	INTEGER	**FK**	Не пустий
	post_id	Унікальний ідентифікатор допису	INTEGER	**FK**	Не пустий
friendships	friendship_id	Унікальний ідентифікатор дружби	INTEGER	**PK**	Не пустий
	source_id	Унікальний ідентифікатор відправника запиту на дружбу	INTEGER	**FK**	Не пустий
	target_id	Унікальний ідентифікатор отримувача запиту на дружбу	INTEGER	**FK**	Не пустий
chats	chat_id	Унікальний ідентифікатор чату	INTEGER	**PK**	Не пустий
	friendship_id	Унікальний ідентифікатор дружби	INTEGER	**FK**	Не пустий

Продовження таблиці 2.1

Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
messages	message_id	Унікальний ідентифікатор повідомлення	INTEGER	**PK**	Не пустий
	text	Текст повідомлення	VARCHAR(200)		Не пустий
	attachment	Файл повідомлення	VARCHAR(200)		
	creation_time	Час відправки повідомлення	TIMESTAMP		Не пустий
	author_id	Унікальний ідентифікатор автора повідомлення	INTEGER	**FK**	Не пустий
forms	form_id	Унікальний ідентифікатор форми зворотнього зв’язку	INTEGER	**PK**	Не пустий
	subject	Заголовок форми зворотнього зв’язку	VARCHAR(100)		Не пустий
	text	Текст форми зворотнього зв’язку	VARCHAR(500)		Не пустий
	attachment	Файл форми зворотнього зв’язку	VARCHAR(200)		
	creation_time	Час відправки форми зворотнього зв’язку	TIMESTAMP		Не пустий
	author_id	Унікальний ідентифікатор автора форми зворотнього зв’язку	INTEGER	**FK**	Не пустий

Відповідно до наведеної вище таблиці, варто навести детальний опис створення необхідних таблиць у нотації MySQL.

```
CREATE TABLE users (
    user_id integer NOT NULL AUTO_INCREMENT,
    avatar varchar(200) NOT NULL,
    first_name varchar(50) NOT NULL,
    second_name varchar(50) NOT NULL,
    nickname varchar(30) NOT NULL UNIQUE,
    email varchar(50) NOT NULL UNIQUE,
    password varchar(200) NOT NULL,
    biography varchar(200),
    registration_time timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (user_id));

CREATE TABLE posts (
    post_id integer NOT NULL AUTO_INCREMENT,
    text varchar(200) NOT NULL,
    attachment varchar(200),
    likes_number integer NOT NULL DEFAULT 0,
    creation_time timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
    author_id integer NOT NULL,
    PRIMARY KEY (post_id),
    FOREIGN KEY (author_id) REFERENCES users (user_id) ON DELETE CASCADE);

CREATE TABLE comments (
    comment_id integer NOT NULL AUTO_INCREMENT,
    text varchar(200) NOT NULL,
    attachment varchar(200),
    likes_number integer NOT NULL DEFAULT 0,
    creation_time timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
    author_id integer NOT NULL,
    post_id integer NOT NULL,
    PRIMARY KEY (comment_id),
    FOREIGN KEY (author_id) REFERENCES users (user_id) ON DELETE CASCADE,
    FOREIGN KEY (post_id) REFERENCES posts(post_id) ON DELETE CASCADE);

CREATE TABLE friendships (
    friendship_id integer NOT NULL AUTO_INCREMENT,
    source_id integer NOT NULL,
    target_id integer NOT NULL,
    PRIMARY KEY (friendship_id),
    FOREIGN KEY (source_id) REFERENCES users (user_id) ON DELETE CASCADE,
    FOREIGN KEY (target_id) REFERENCES users (user_id) ON DELETE CASCADE);

CREATE TABLE chats (
    chat_id integer NOT NULL AUTO_INCREMENT,
    friendship_id integer NOT NULL UNIQUE,
    PRIMARY KEY (chat_id),
    FOREIGN KEY (friendship_id) REFERENCES friendships (friendship_id) ON DELETE
CASCADE);
```

```
CREATE TABLE messages (
    message_id integer NOT NULL AUTO_INCREMENT,
    text varchar(200) NOT NULL,
    attachment varchar(200),
    creation_time timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
    author_id integer NOT NULL,
    chat_id integer NOT NULL,
    PRIMARY KEY (message_id),
    FOREIGN KEY (author_id) REFERENCES users (user_id) ON DELETE CASCADE,
    FOREIGN KEY (chat_id) REFERENCES chats (chat_id) ON DELETE CASCADE);

CREATE TABLE forms (
    form_id integer NOT NULL AUTO_INCREMENT,
    subject varchar(100) NOT NULL,
    text varchar(500) NOT NULL,
    attachment varchar(200),
    creation_time timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
    author_id integer NOT NULL,
    PRIMARY KEY (form_id),
    FOREIGN KEY (author_id) REFERENCES users (user_id) ON DELETE CASCADE);
```

Таким чнном вдалося реалізувати структуру фізичної моделі БД у вигляді ERD-діаграми (рис. 2.9).

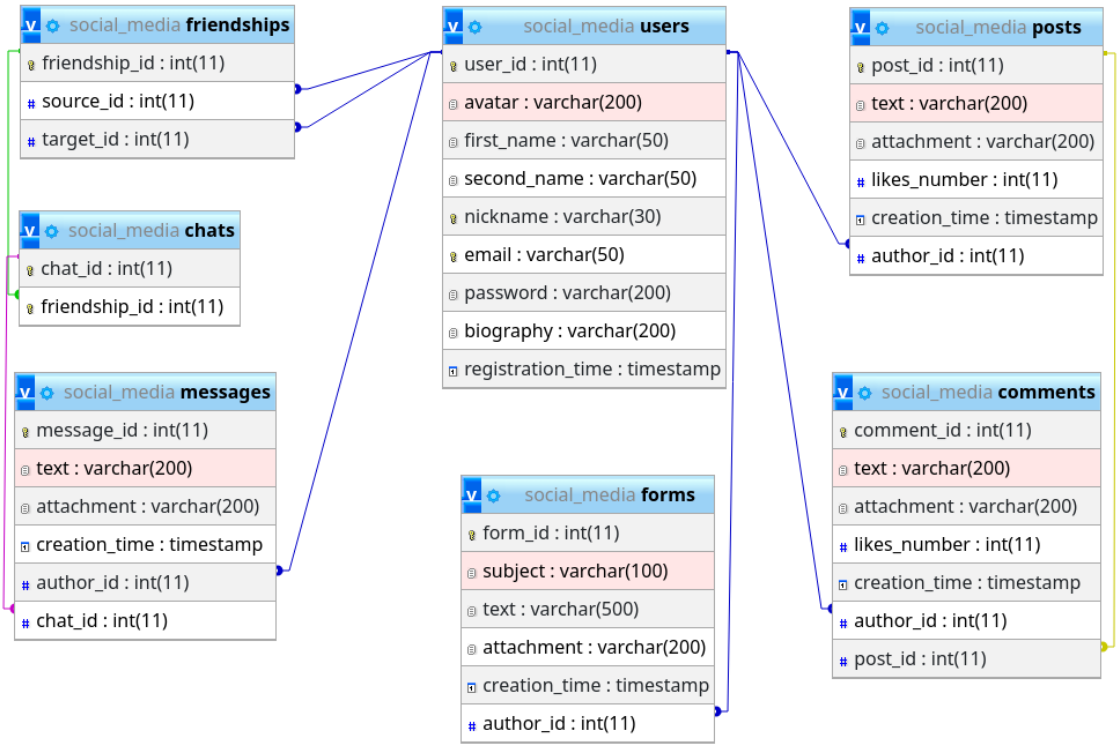


Рисунок 2.9 – Фізична модель даних вебдодатку соціальної мережі «ConnectHub»

2.4 Архітектура вебдодатку соціальної мережі «ConnectHub»

У сучасному світі розробки існує безліч архітектур проєктування. Одними з таких є **MVC (Model — View — Controller)** та **MVP (Model — View — Presenter)**. Останній вважається деяким переосмисленням стандартної схеми модель-представлення-контроллер, адже вирішує деякі її ключові проблеми:

– Більшість бізнес-логіки залишається у контроллері, у той час під час життєвого циклу програми кількість файлів збільшується і це робить неможливим підтримувати код;

– Через тугозв'язні компоненти представлення та контроллера стає дуже важко впроваджувати нові особливості додатку;

– Різні шари додаткустає важчим для тестування.

Архітектура **MVP** допомагає вирішити ці проблеми та дозволяє будувати чітку та надійну архітектуру. Основними компонентами, як було зазначено раніше є:

– Модель – вона відповідає за виконання операцій обробки, збереження, сортування даних, генерацію вебсторінок та зв'язок з базою даних;

– Представлення – клієнтська частина, що реагує на дії користувача і надсилає сигнали контроллеру. Саме вона забезпечує взаємодію користувача з вебдодатком;

– Пред'явник – шар, що об'єднує модель та представлення. Він приймає запити, взаємодіє з базою даних і відображає зміни у клієнтській частині вебдодатку.

Саме така архітектура є вкрай зручною для реалізації та підтримки вебдодатків.

Загальну схему роботи архітектури MVP зображено на рис. 2.10.

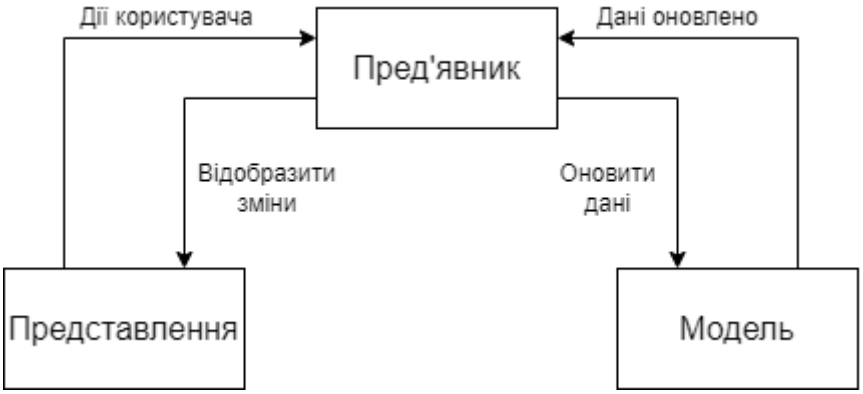


Рисунок 2.10 – MVP-архітектура

2.5 Висновки по другому розділу

Під час виконання моделювання та проєктування вебдодатку соціальної мережі «ConnectHub» створено структурно-функціональну модель соціальної взаємодії за допомогою вебдодатку у нотації IDEF0 та її декомпозицію, що дало змогу глибше оцінити функціонал системи, розроблено діаграму варіантів використання, яка описує основні можливості використання вебдодатку для різних акторів, починаючи з діаграми потоків даних виявлено необхідну сутність системи – користувачі та спроєктовано логічну та фізичну моделі бази даних для даної сутності, оглянуто сучасні архітектури створення вебдодатів та обрано найкращий аналог для подальшої розробки.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ВЕБДОДАТКУ СОЦІАЛЬНОЇ МЕРЕЖІ «CONNECTHUB»

3.1 Програмна реалізація вебдодатку соціальної мережі «ConnectHub»

Реалізація вебдодатку виконана за допомогою методології RESTful API, згідно з якою вебсторінка функціонує на одному доменному адресі, проте може обробляти різні запити в залежності від кінцевою точки адреси. Для того, щоб реалізувати дану технологію і використовується вебсервер, який побудовано за допомогою бібліотеки «Express». У даному фрагменті коду показано ініціалізацію програми та підключення основних бібліотек у файлі «index.js»:

```
const app = express();
const port = 3080;

const __dirname = dirname(fileURLToPath(import.meta.url));

app.use(express.json());
app.use(fileUpload());
app.use(express.urlencoded({ extended: true }));
app.set('views', join(__dirname, 'views'));
app.set('view engine', 'pug');
app.use(express.static(join(__dirname, 'static')));

initRoutes(app);

app.use((req, res) => {
  res.status(404).render('errors/404');
});

app.listen(port, () => {
  console.log(`Listening on port ${port}`);
});
```

Функція ініціалізації шляхів дозволяє використати REStful API для побудови різноманітних шляхів вебдодатку:

```
const initRoutes = (app) => {
  app.use('/signup', sighUpRouter);
  app.use('/login', loginRouter);
  app.use('/home, feedRouter);
  app.use('/chats', chatRouter);
  app.use('/settings', settingsRouter);
  app.use('/user', userRouter);
  app.get('/', (req, res) => {
    res.redirect('/home);
  });
};
```

Перелік основних компонентів програми наведено на рис. 3.1

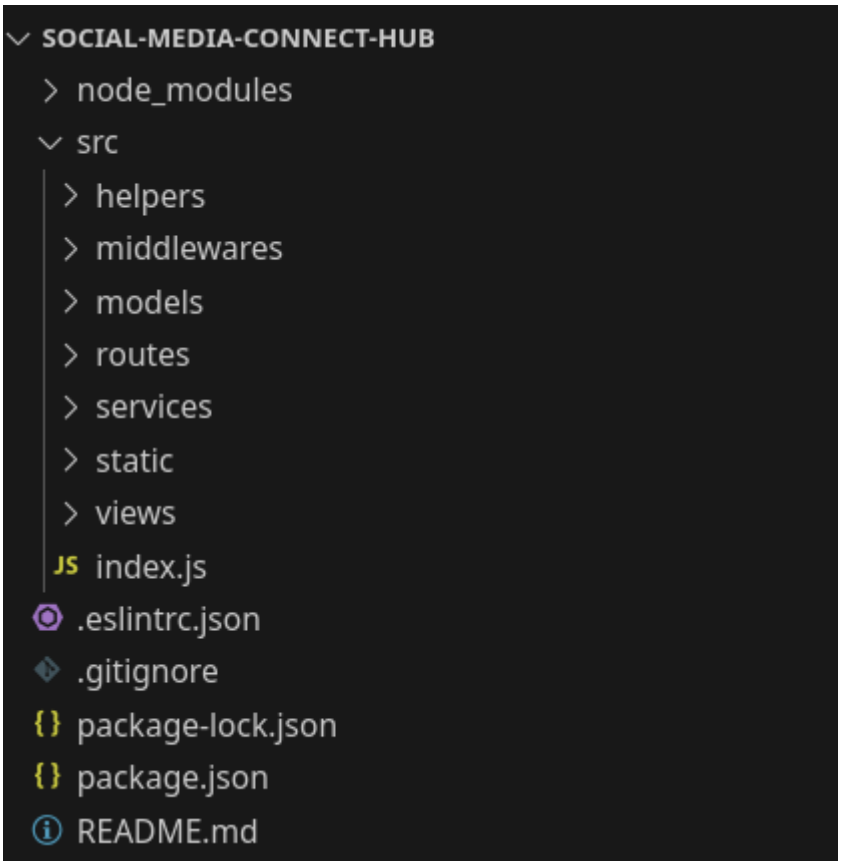


Рисунок 3.1 – Основні компоненти програми

Як можна побачити з рисунку вище програма складається з компонентів:

– Helpers – функції загального призначення (підключення до БД, шаблон запиту до БД, збереження файлу на сервері);

– Middleware – шар функцій-посередників, що здійснюють валідацію, хешування даних, обробку помилок ;

– Models – відповідає за представлення та тип даних, що зберігається у базі даних;

– Routes – шляхи вебдодатку, що приймають різноманітні типи запитів і повертають результат. У даному випадку це блок «Presenter» ;

– Services– відповідає за запити до бази даних та повернення результатів запитів;

– Static – статичні файли, що використовуються для рендеру сторінок (фото, стилі тощо) ;

– Views – сторінки вебдодатку, які вебсервер рендерить і повертає до клієнта.

3.2 Використання вебдодатку соціальної мережі «ConnectHub»

Використання вебдодатку соціальної мережі «ConnectHub» складається з трьох основних частин: з точки зору відвідувача, користувача та адміністратора.

– Відвідувач має невелику кількість привілеїв. Він може лише переглядати контент вебдодатку: головну сторінку та профілі користувачів. Також йому доступні опції реєстрації або авторизації.

– Користувач у свою чергу є авторизованим у вебдодаток, тож може не тільки переглядати вміст вебдодатку, але взаємодіяти з іншими: створювати дописи, коментарі, заводити знайомства та обмінюватись приватними повідомленнями.

– Адміністратор має всі ті ж права, що й авторизований користувач, проте йому також відкритий доступ до адміністративної панелі, де він може змінювати наповнення вебдодатку.

3.2.1 Використання вебдодатку соціальної мережі «ConnectHub» відвідувачем

На початку роботи програми відвідувач може побачити перед собою наповнення головної сторінки. Воно містить у собі дописи від користувачів вебдодатку (рис. 3.2).

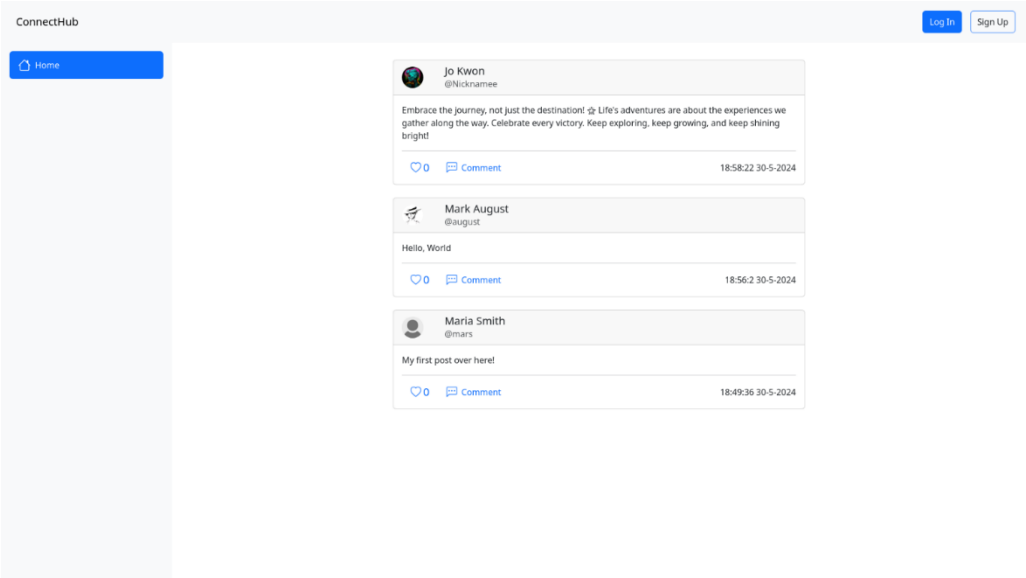


Рисунок 3.2 – Головна сторінка з точки зору відвідувача

Відвідувач може переглянути профілі користувачів, якщо обере та натисне на бажаний (рис. 3.3).

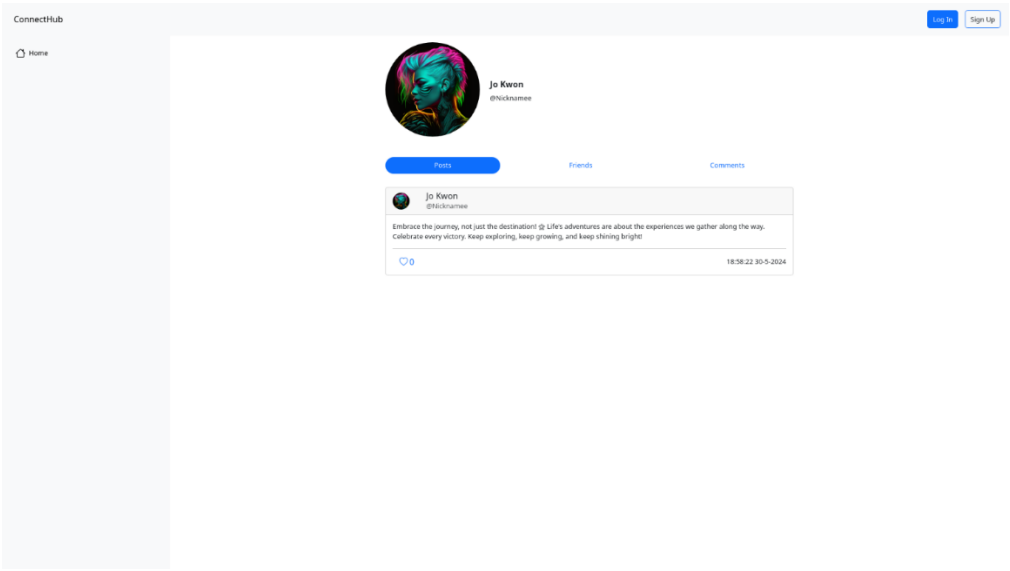


Рисунок 3.3 –Профіль користувача з точки зору відвідувача

Звичайно, йому доступні варіанти авторизації або реєстрації (рис. 3.4-3.5).

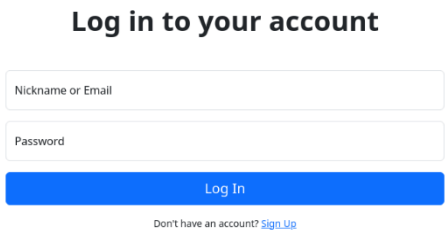


Рисунок 3.4 – Сторінка авторизації

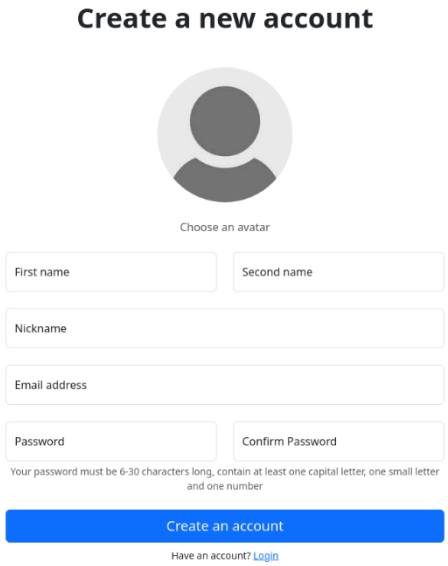


Рисунок 3.5 – Сторінка реєстрації

3.2.2 Використання вебдодатку соціальної мережі «ConnectHub» користувачем

Головна сторінка для користувача тепер містить можливість додавати коментарі (рис. 3.6-3.8).

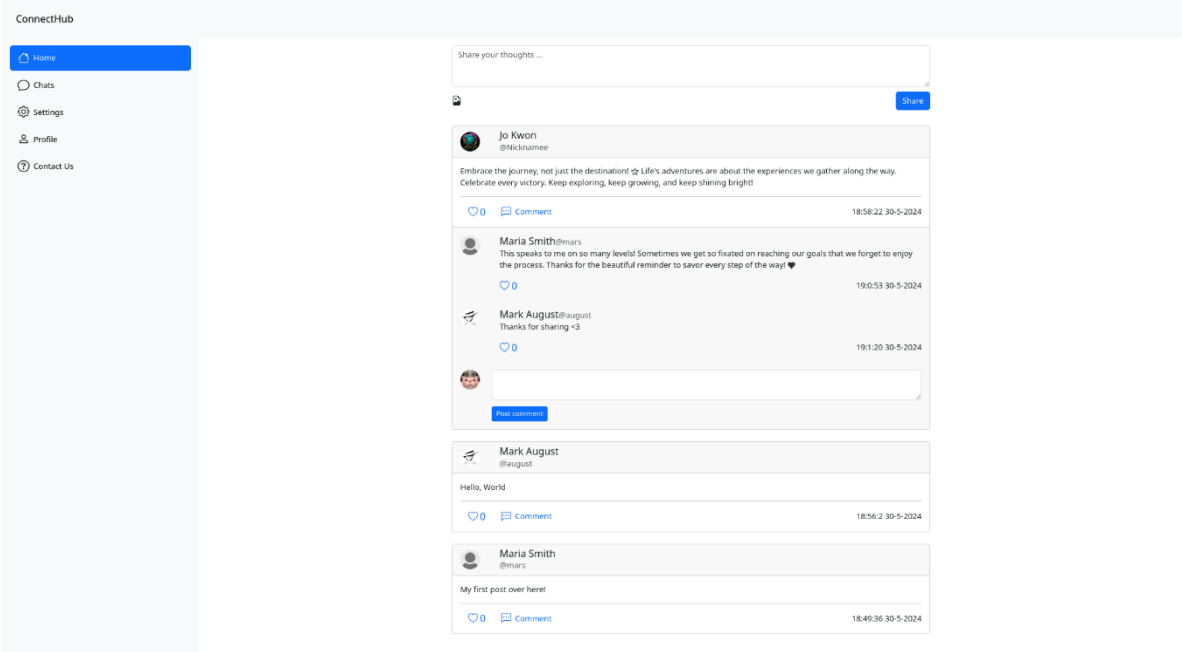


Рисунок 3.6 – Перегляд коментарів з точки зору користувача

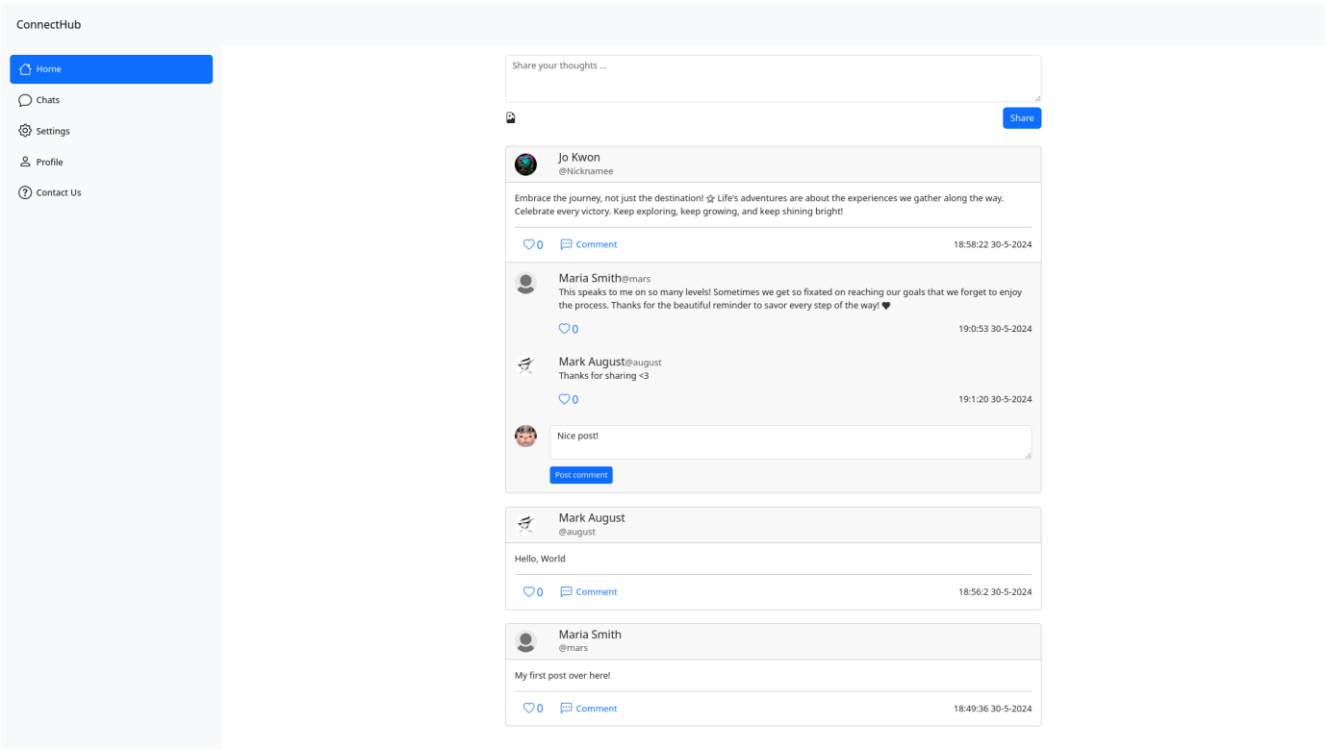


Рисунок 3.7 – Коментування допису з точки зору користувача

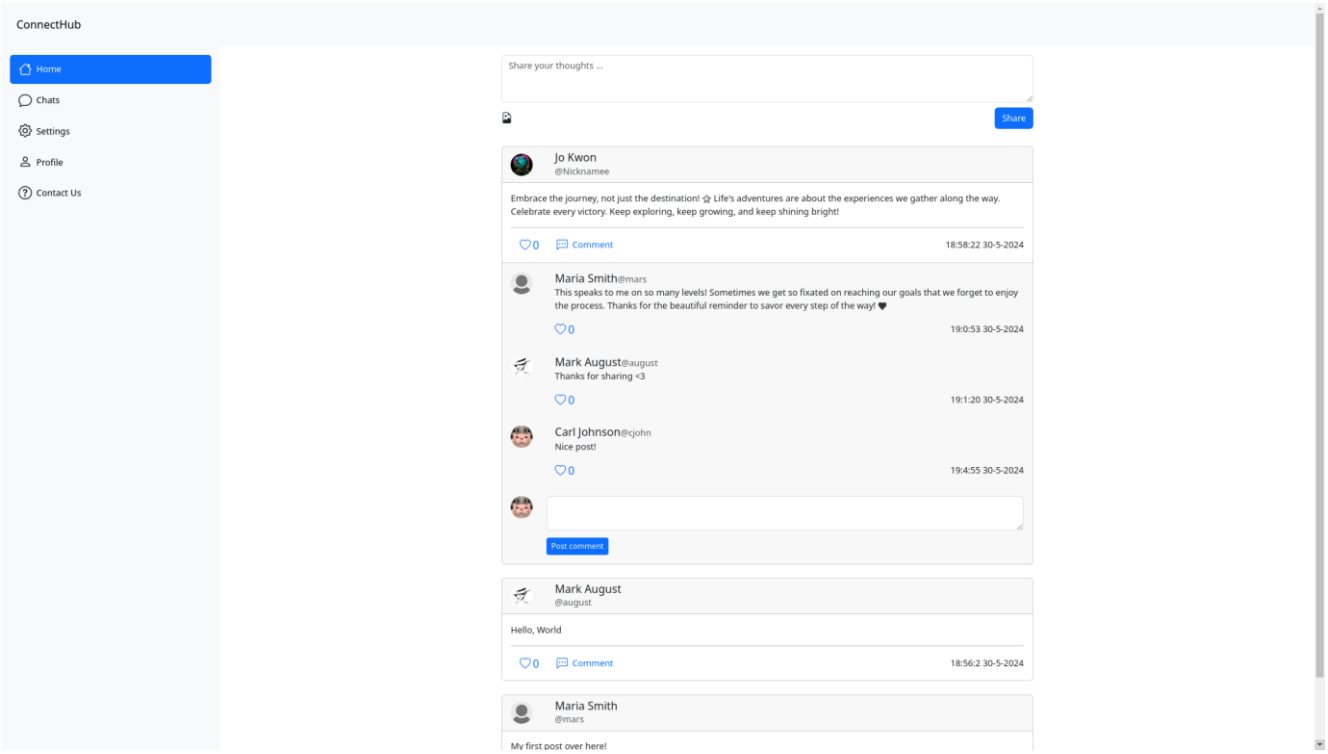


Рисунок 3.8 – Створений коментар з точки зору користувача

Також користувачу відкривається опція додавання дописів (рис. 3.9-3.10).

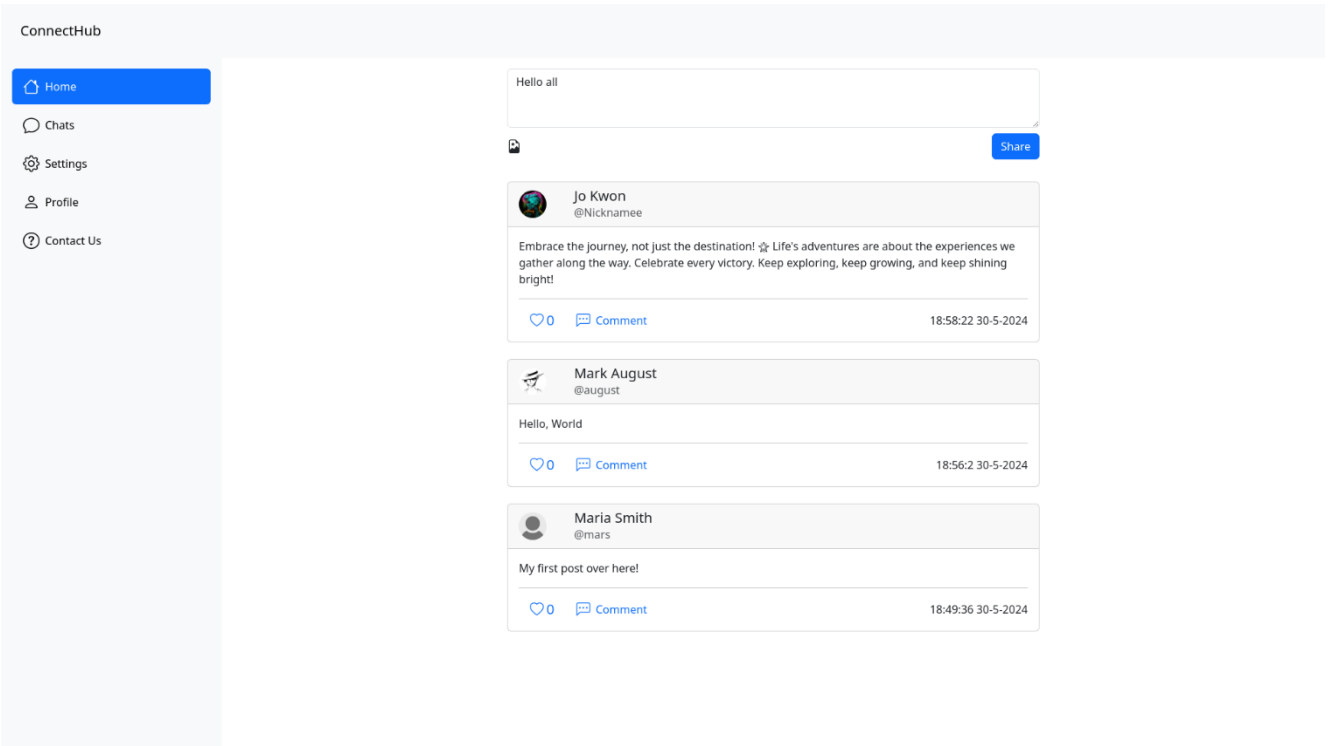


Рисунок 3.9 – Створення допису з точки зору користувача

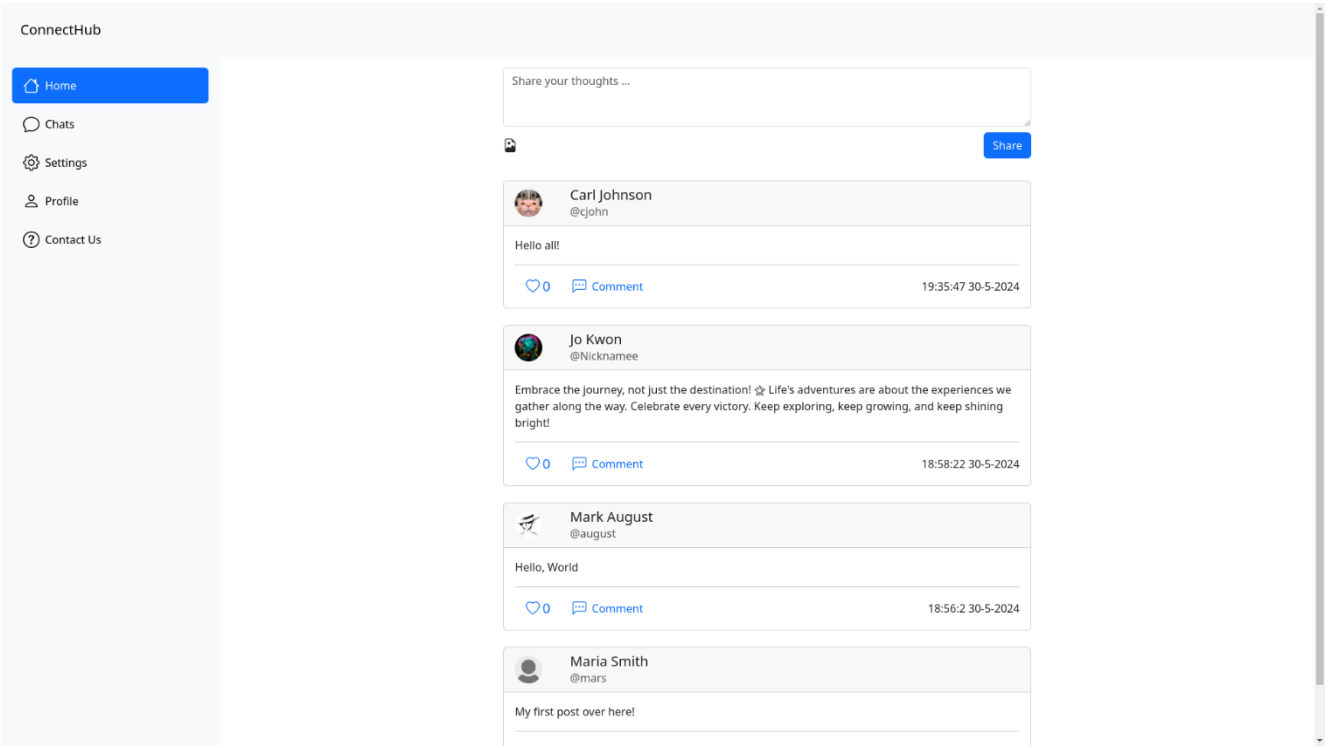


Рисунок 3.10 – Створений допис з точки зору користувача

Він може переглянути власний профіль: свої дописи, повідомлення та список друзів (рис. 3.11).

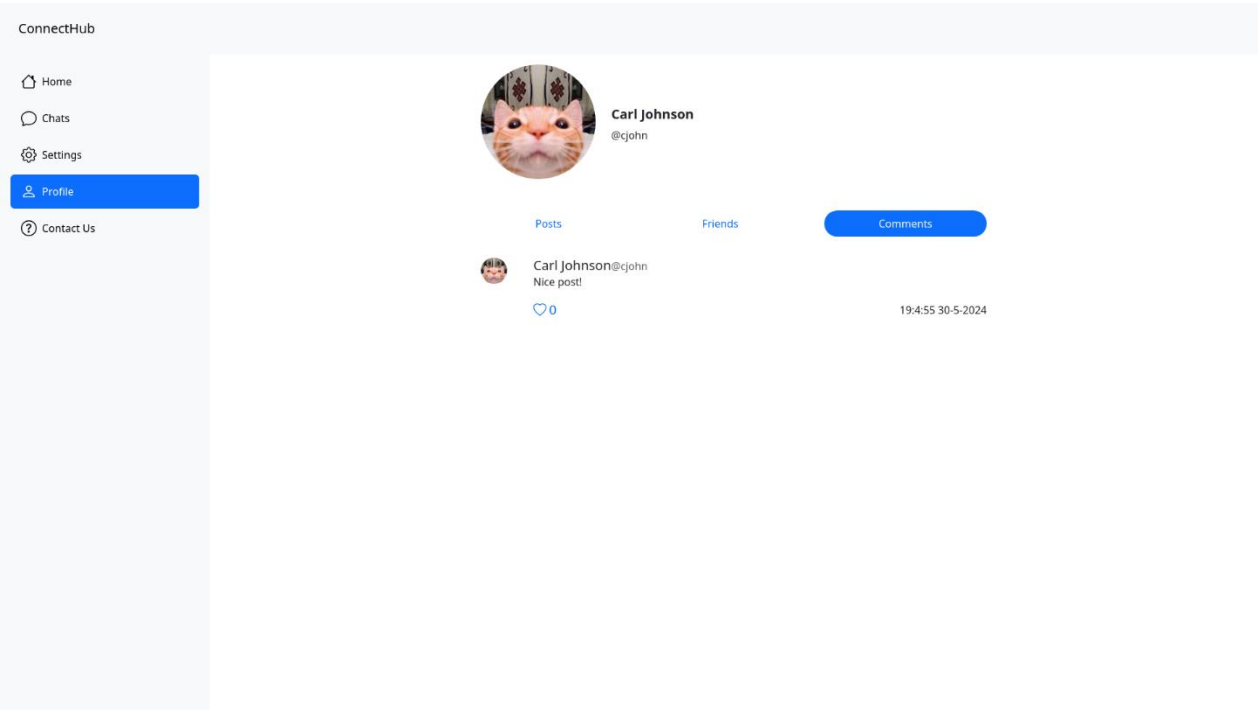


Рисунок 3.11 – Власний профіль користувача

Також він може переглядати сторінки інших користувачів і відправляти їм запити на дружбу (рис. 3.12).

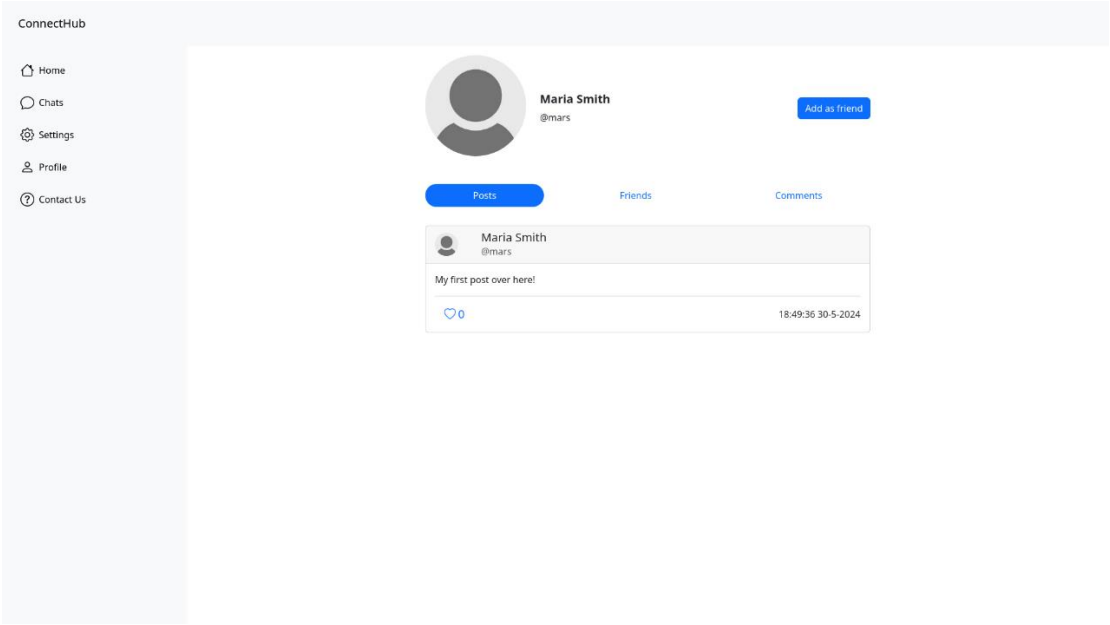


Рисунок 3.12 – Профіль іншого користувача з точки зору користувача

Після того, як користувачі стали друзями, вони можуть обмінюватися приватними повідомленнями (рис. 3.13).

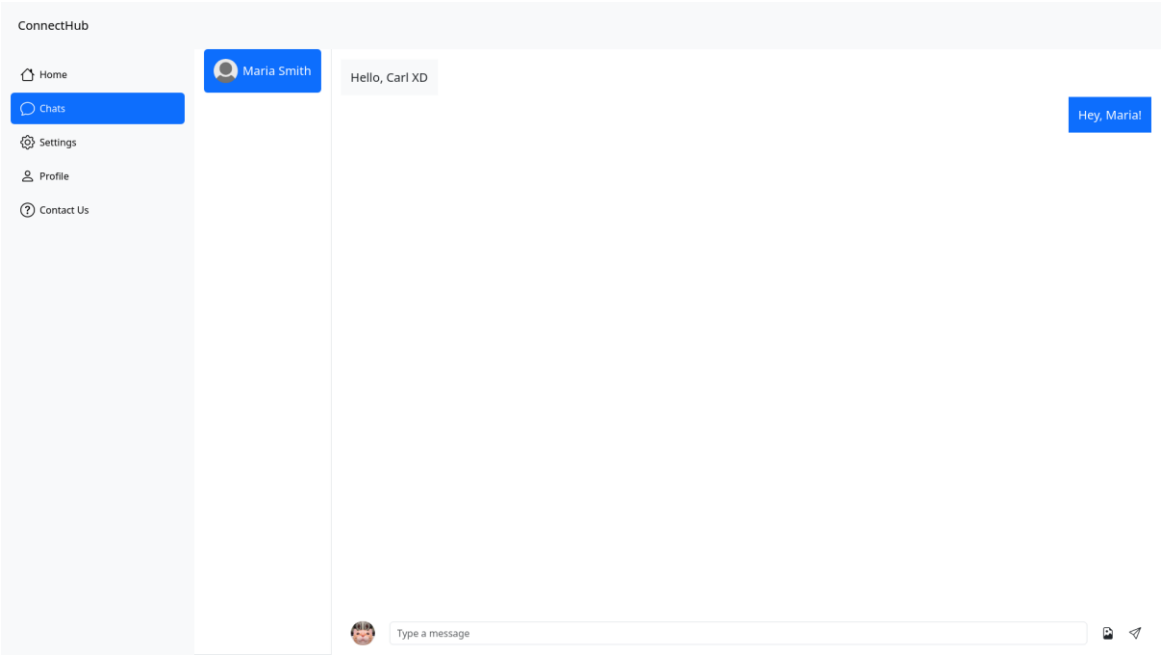


Рисунок 3.13 – Обмін повідомленнями між користувачами

Користувачу доступна зміна налаштувань власного профілю (рис. 3.14).

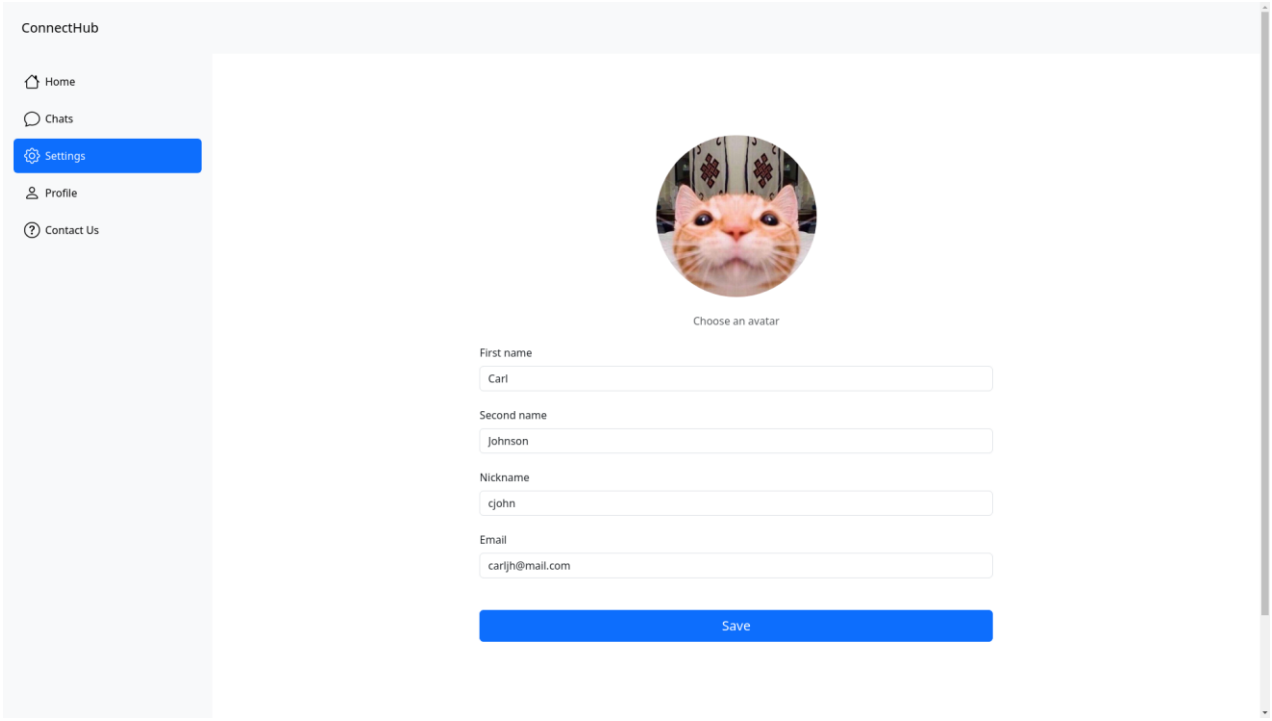


Рисунок 3.14 – Редагування профілю з точки зору користувача

Також є можливість видалити сторінку профілю (рис. 3.15).

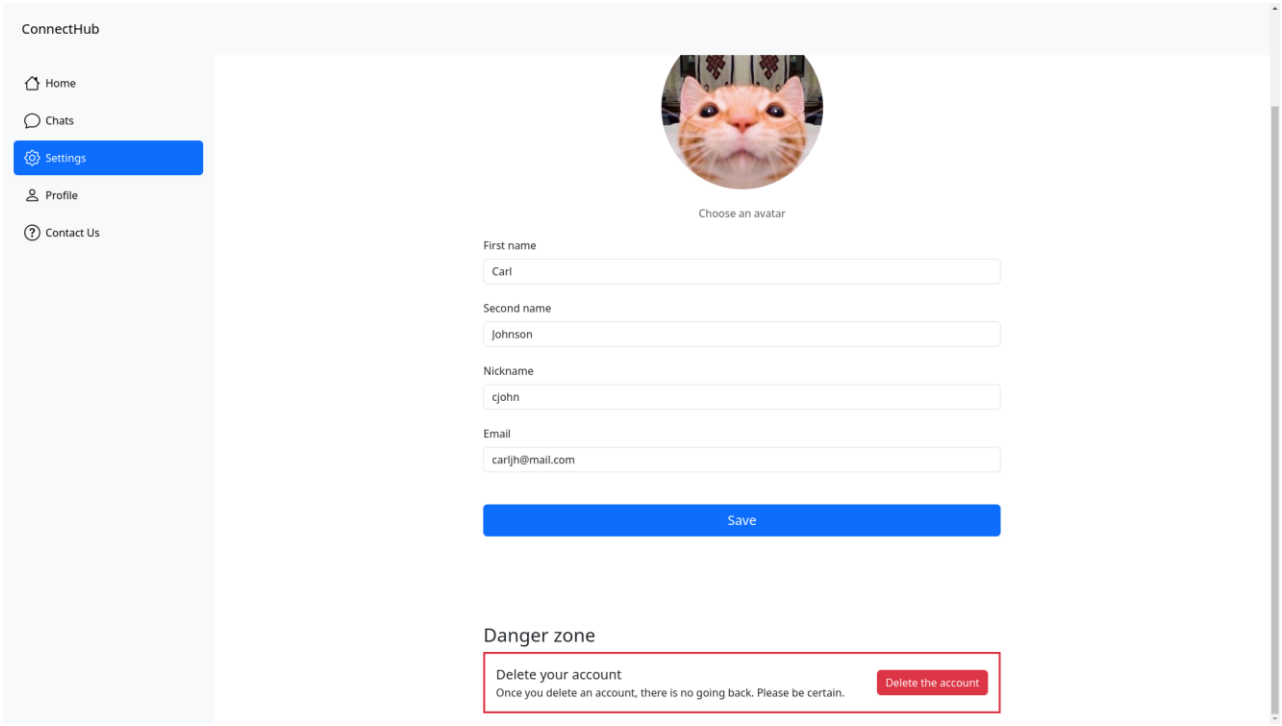


Рисунок 3.15 – Можливість видалення профілю з точки зору користувача

Відправка форм до адміністрації також особливість авторизованого користувача (рис. 3.16).

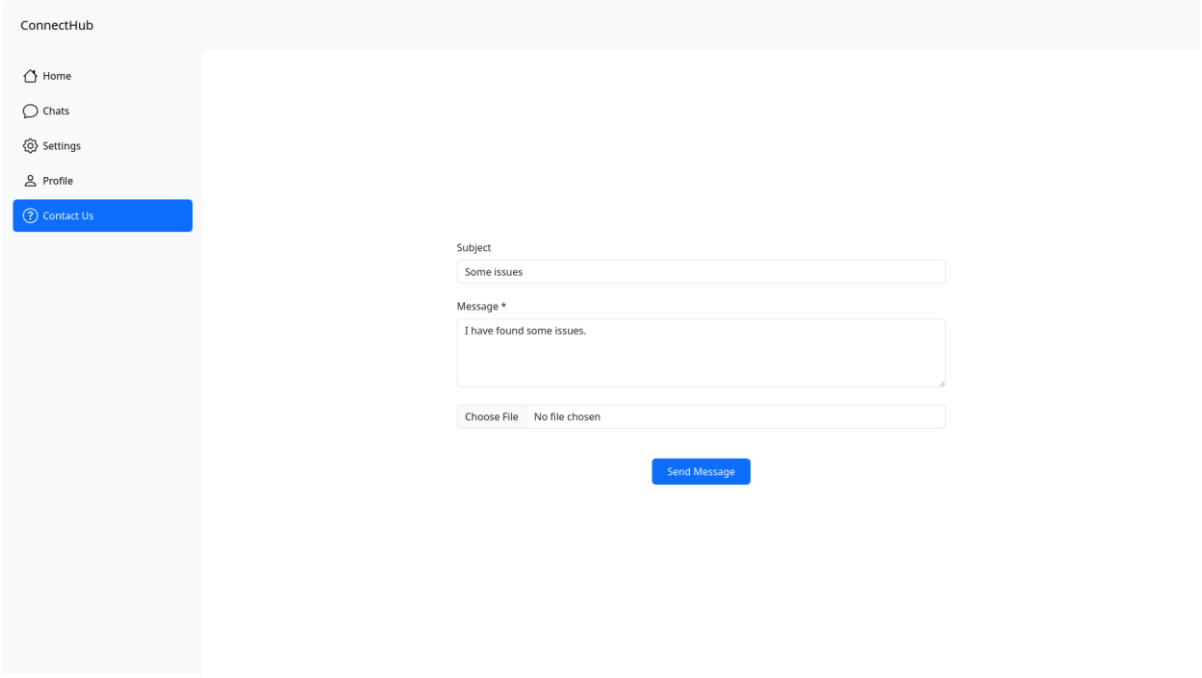


Рисунок 3.16 – Відправка форм зворотнього зв’язку з точки зору користувача

3.2.3 Використання вебдодатку соціальної мережі «ConnectHub» адміністратором

Адміністратор має унікальні можливості для повного керування наповненням вебдодатку. Однією з таких є перегляд, редагування та видалення профілів інших користувачів (рис. 3.17-3.18).

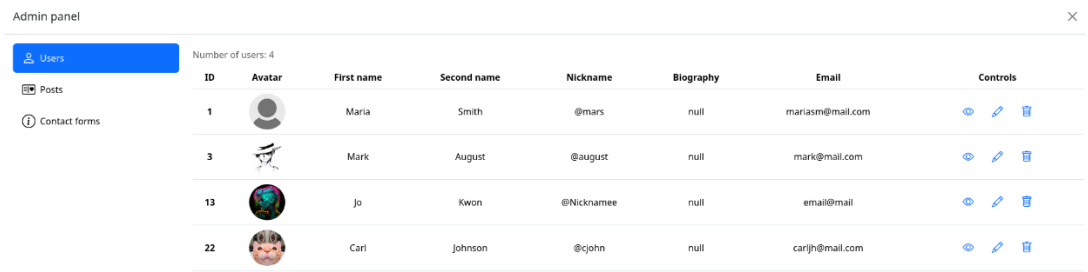


Рисунок 3.17 – Перегляд профілів користувачів вебдодатку з точки зору адміністратора

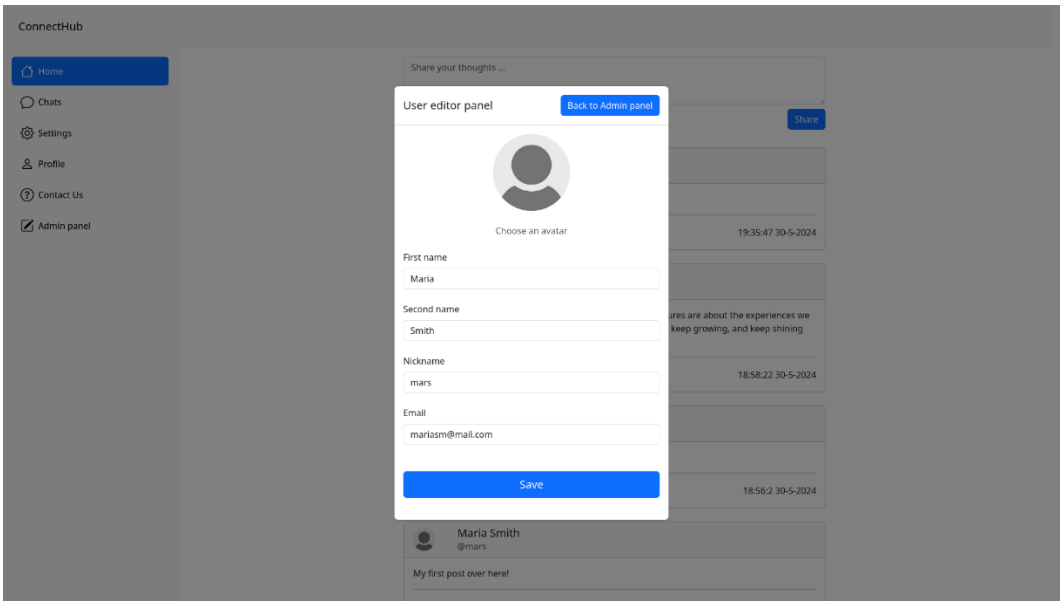


Рисунок 3.18 – Редагування профілю користувача вебдодатку з точки зору адміністратора

Також адміністратор може здійснювати моніторинг та маніпулювання даними з дописів користувачів (рис. 3.19).

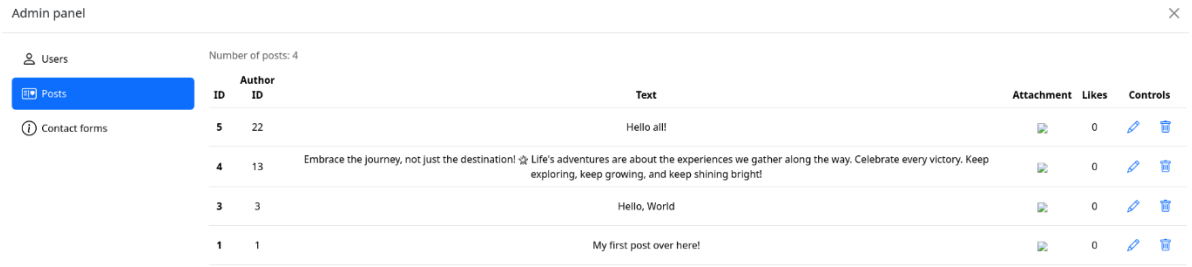


Рисунок 3.19 – Маніпулювання дописами користувачів вебдодатку з точки зору адміністратора

І звісно, він також має можливість переглядати та відповідати на форми зворотнього зв’язку (рис. 3.20-3.21).

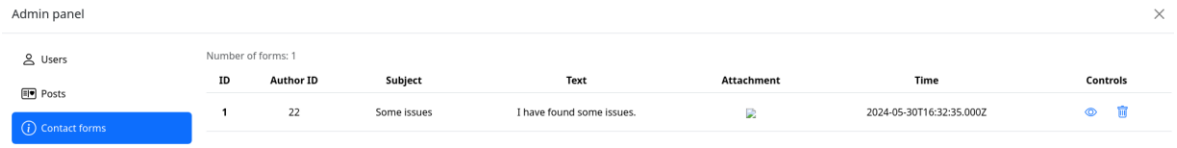


Рисунок 3.20 – Перегляд форм зворотнього зв’язку з точки зору адміністратора

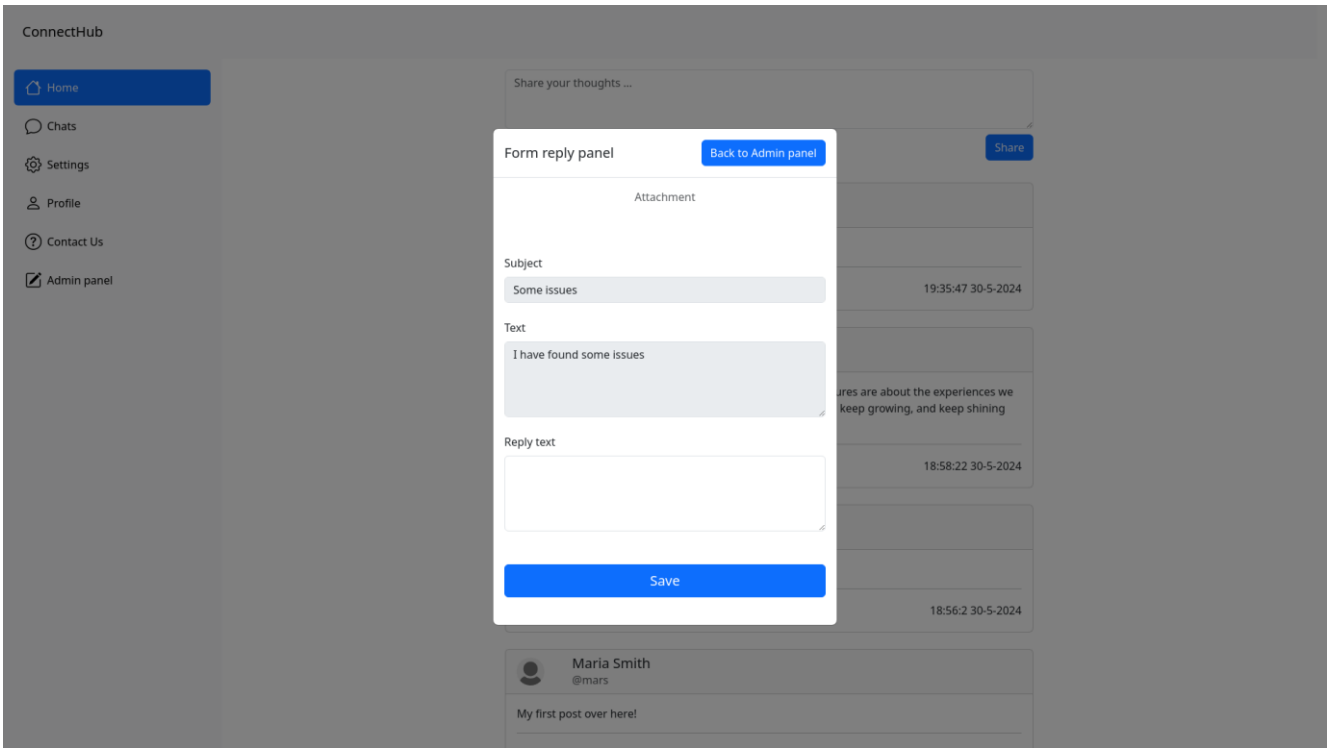


Рисунок 3.21 – Відповідь на форму зворотнього зв'язку з точки зору адміністратора

3.3 Тестування вебдодатку соціальної мережі «ConnectHub»

Основними вразливими компонентами системи є авторизація, реєстрація користувачів та режими доступу до різноманітних ресурсів вебдодатку. Це вкрай важливі аспекти вебдодатку, що впливають на захист, працездатність та відмовостійкость.

Валідація форми реєстрації може допомогти уникнути проблем з одним із вищенаведених компонентів та зберегти надійну роботу вебдодатку. Основні помилки некоректності внесених облікових даних мають чітко відображатись на сторінці (рис. 3.22-3.23).

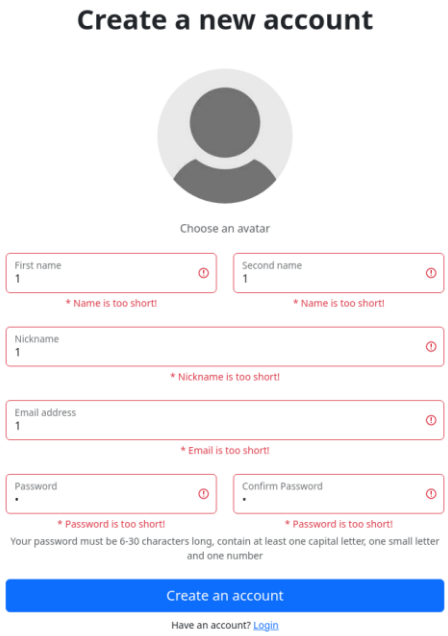


Рисунок 3.22 – Невалідні дані при спробі реєстрації

Рисунок 3.23 – Невалідні дані при спробі реєстрації

Форма авторизації є досить схожою і працює за таким же принципом, надаючи короткі та вичерпні повідомлення щодо невірних облікових даних (рис. 3.24-3.27).

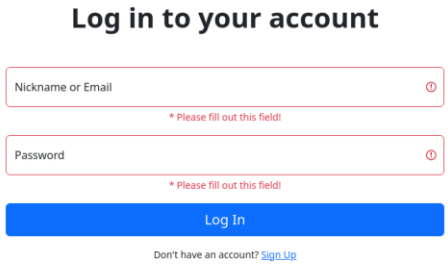


Рисунок 3.24 – Невалідні дані при спробі авторизації

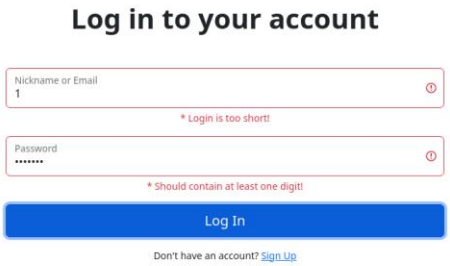


Рисунок 3.25 – Невалідні дані при спробі авторизації

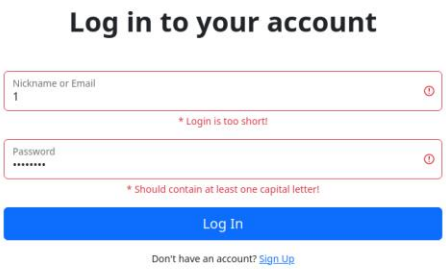


Рисунок 3.26 – Невалідні дані при спробі авторизації

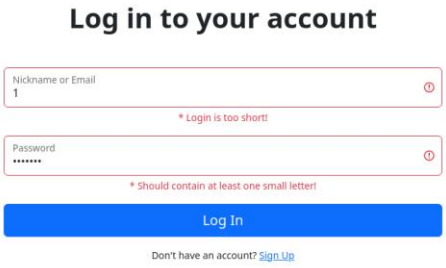


Рисунок 3.27 – Невалідні дані при спробі авторизації

Такий же принцип працює і для форми редагування власного профілю (рис. 3.28-3.29).

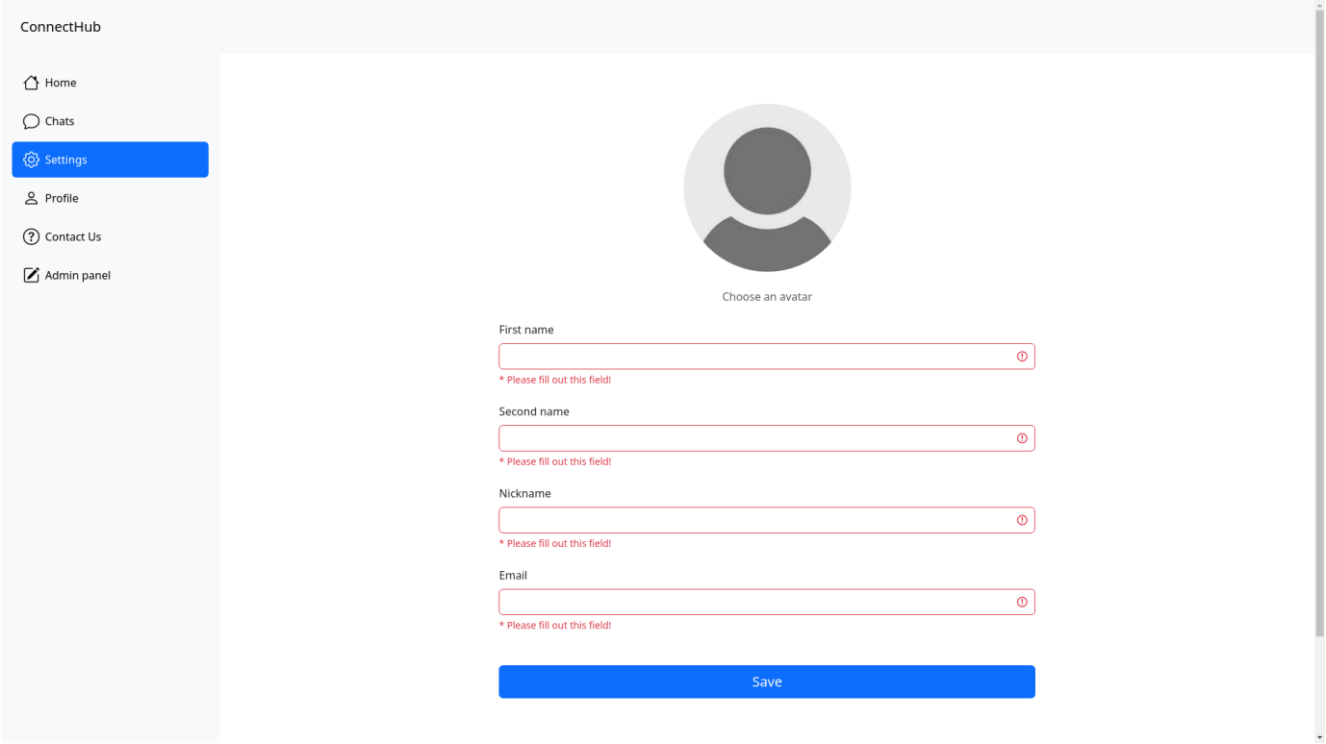


Рисунок 3.28 – Невалідні дані при спробі редагування профілю

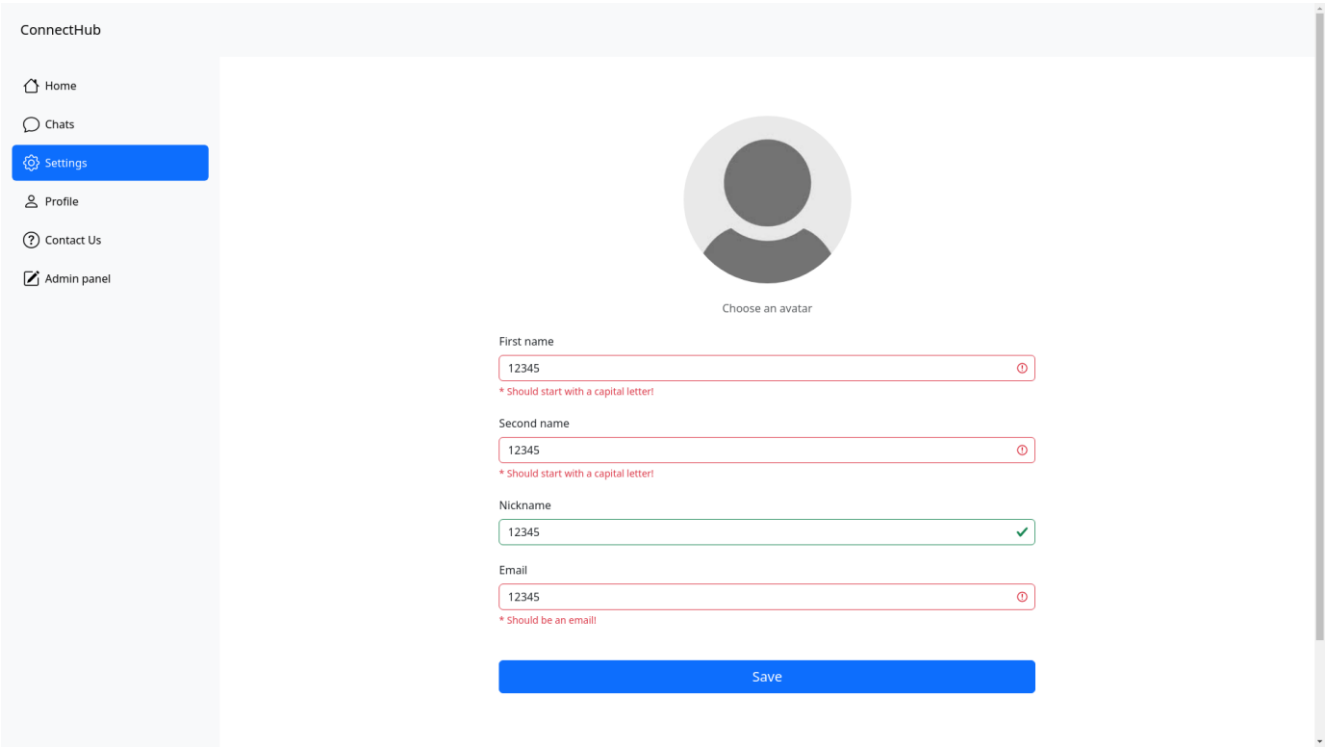


Рисунок 3.29 – Невалідні дані при спробі редагування профілю

Якщо ж відвідувач захоче переглянути контент, який йому недоступний, вебдодаток відправить відповідне повідомлення та запропонує повернутися на головну сторінку (рис. 3.30).

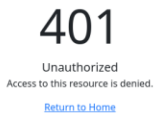


Рисунок 3.30 – Помилка доступу до ресурсу

Якщо ж раптом дехто з користувачів спробує відвідати неіснуючу сторінку вебдодатку, йому також надійде повідомлення про помилку запиту (рис. 3.31).

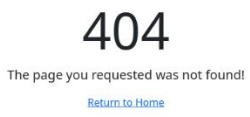


Рисунок 3.31 – Помилка існування ресурсу

У разі, якщо база даних, вебдодаток або вебсервер не зможуть опрацювати запит, вийдуть з ладу, відмовлять в роботі тощо, то користувачу надійде сторінка з повідомлення про дану помилку (рис. 3.32).

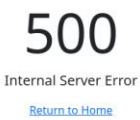


Рисунок 3.32 – Помилка в роботі компонентів системи вебдодатку

3.4 Висновки по третьому розділу

Під час виконання практичної реалізації вебдодатку соціальної мережі «ConnectHub» відтворено програмну частину, зображено та описано основні компоненти програми, продемонстровано та прокоментовано варіанти використання вебдодатку зі сторони різних акторів, протестовано вебдодаток і отримано прийнятний рівень працездатності вебдодатку соціальної мережі «ConnectHub».

ВИСНОВКИ

Під час виконання кваліфікаційної роботи виконано ряд поставлених завдань, що полягали у дослідженні області вебдодатків соціальних мереж, виявленні поточних проблем цієї сфери та визначенні актуальності роботи. Також проведено аналіз аналогів вебдодатку соціальної мережі «ConnectHub», в результаті якого були виявлені їх переваги та недоліки. Описано функціональні вимоги до розроблюваного вебдодатку соціальної мережі «ConnectHub». Крім того, обрано необхідні технології для реалізації цього проекту.

У першій частині роботи проведено аналіз області вебдодатків соціальних мереж, виявлено її проблеми. Докладно описано актуальність розробки вебдодатку. Розглянуто можливі технологічні засоби, що дозволили б вирішити поставлені задачі, обрано необхідні інструменти для розробки вебдодатку соціальної мережі «ConnectHub» та проаналізовано існуючі аналоги вебдодатку соціальної мережі «ConnectHub», чітко сформульовано мету роботи, детально описано задачі для її досягнення та функціональні вимоги розроблюваного вебдодатку соціальної мережі «ConnectHub».

У другій частині роботи виконано моделювання вебдодатку соціальної мережі «ConnectHub» за допомогою нотації IDEF, що дало змогу побачити основні структурно-функціональні процеси вебдодатку. Спроєктовано інформаційну систему за допомогою діаграми варіантів використання, що показало, які можливості мають різні актори вебдодатку. Розроблено логічну та фізичну моделі бази даних, яка являє собою сховище для різноманітних сутностей вебдодатку, що дозволило коректно реалізувати логіку вебдодатку.

У третій частині роботи наведено практичну реалізацію створення вебдодатку. Зображено використання вебдодатку з точки зору різних акторів вебдодатку. Протестовано вебдодаток соціальної мережі «ConnectHub».

У ході виконання кваліфікаційної роботи бакалаврра детально описано кожен крок та підбито підсумки результатів роботи усіх розділів.

У результаті кваліфікаційної роботи розроблено вебдодаток соціальної мережі «ConnectHub». Створено сторінки реєстрації, авторизації, головна, приватні повідомлення, налаштування, профіль користувача, форма зворотнього зв’язку та адміністративна панель.

У рамках кваліфікаційної роботи описано та підготовлено додатки, як-от: додаток А – технічне завдання, додаток Б – планування робіт, додаток В – програмний код, додаток Г – апробація результатів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dollarhide M. Social media: definition, importance, top websites & apps. *Investopedia*. URL: https://www.investopedia.com/terms/s/social-media.asp (date of access: 24.04.2024).

2. The 22 benefits of social media for business in 2022 and beyond. *HR Technologies France*. URL: https://www.hrtechnologiesfrance.com/latest-news-and-updates/22-benefits-of-social-media-for-business-lqyb (date of access: 24.04.2024).

3. Вплив соціальних мереж на суспільство і бізнес: переваги, недоліки та майбутні перспективи. *InProject - IT компанія, яка створює неймовірні digital продукти*. URL: https://inproject.org/vplyv-soczialnuh-merezh-na-suspilstvo-i-biznes/ (дата звернення: 24.04.2024).

4. Dean B. Social network usage & growth statistics: how many people use social media in 2024?. *Backlinko*. URL: https://backlinko.com/social-media-users (date of access: 24.04.2024).

5. Ahmad R. Is social media a social problem or a social solution?. *LinkedIn: Log In or Sign Up*. URL: https://www.linkedin.com/pulse/social-media-problem-solution-rashid-ahmad/ (date of access: 24.04.2024).

6. Sari D. A., Pasaribu R. A. Humans as individual and social beings in everyday life. *International journal of students education*. 2023. URL: https://doi.org/10.62966/ijose.v1i1.361 (date of access: 24.04.2024).

7. Global: number of smartphone users 2014-2029 | Statista. *Statista*. URL: https://www.statista.com/forecasts/1143723/smartphone-users-in-the-world (date of access: 24.04.2024).

8. Preston. Media vs messaging apps. *Smirk Smirk | Preston | Substack*. URL: https://smirk.substack.com/p/media-vs-messaging-apps (date of access: 24.04.2024).

9. What is full stack. *W3Schools Online Web Tutorials*. URL: https://www.w3schools.com/whatis/whatis_fullstack.asp (date of access: 24.04.2024).

10. Якушенко Д. Full Stack Developer: хто це, чим займається, скільки заробляє та як ним стати. *Make it in Ukraine: Hire Brilliant Talent Fast*. URL: https://www.makeitinua.com/posts-ua/full-stack-developer-hto-ce-chim-zaymaietsya-skilki-zaroblyaie-ta-yak-nim-stati (дата звернення: 24.04.2024).

11. Herbert D. What is React.js? Uses, Examples, & More. *HubSpot Blog | Marketing, Sales, Agency, and Customer Success Content*. URL: https://blog.hubspot.com/website/react-js (date of access: 24.04.2024).

12. Vue.js. *Vue.js - The Progressive JavaScript Framework | Vue.js*. URL: https://vuejs.org/ (date of access: 24.04.2024).

13. Angular. *Angular*. URL: https://angular.io/guide/what-is-angular (date of access: 24.04.2024).

14. ReactJS vs. other front-end frameworks: choosing the right tool for your project. *Medium*. URL: https://medium.com/@GuruTechnolabs/reactjs-vs-other-front-end-frameworks-choosing-the-right-tool-for-your-project-b713d5214f71 (date of access: 24.04.2024).

15. Syftet Ltd. How does react differ from other javascript frameworks?. *LinkedIn: Log In or Sign Up*. URL: https://www.linkedin.com/pulse/how-does-react-differ-from-other-javascript-frameworks-syftet/ (date of access: 24.04.2024).

16. Facebook – log in or sign up. *Facebook*. URL: https://www.facebook.com/ (date of access: 24.04.2024).

17. Twitter. *Twitter*. URL: https://twitter.com/ (date of access: 24.04.2024).

18. Instagram. *Instagram*. URL: https://www.instagram.com/ (date of access: 24.04.2024).

19. Facebook overview. *Small business marketing tips and advice | Marketing Donut*. URL: https://www.marketingdonut.co.uk/social-media/facebook/facebook-overview (date of access: 24.04.2024).

20. Instagram overview. *Small business marketing tips and advice | Marketing Donut*. URL: https://www.marketingdonut.co.uk/social-media/instagram/instagram-overview (date of access: 24.04.2024).

21. Twitter overview. *Small business marketing tips and advice | Marketing Donut*. URL: https://www.marketingdonut.co.uk/social-media/twitter/twitter-overview (date of access: 24.04.2024).

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ
на розробку інформаційної системи
«Вебдодаток соціальної мережі «ConnectHub»»

ПОГОДЖЕНО:

Старший викладач
кафедри інформаційних технологій

_____ к.т.н., Бойко О.В.

Студент групи ІТ-01

_____ Денисенко Ф. М.

2024

1 Призначення й мета вебдодатку соціальної мережі «ConnectHub»

1.1 Призначення вебдодатку

Вебдодаток призначений для організації діяльності соціальної мережі «ConnectHub».

1.2 Мета створення вебдодатку

Метою роботи є розробка вебдодатку соціальної мережі з інтуїтивно зрозумілим інтерфейсом, що забезпечить користувачів стабільним зв'язком у віртуальному середовищі. Він дозволить створювати унікальні сторінки особистого профілю, заводити нові знайомства, підтримувати контакт з друзями та рідними, обговорювати новини та важливі суспільні питання.

1.3 Цільова аудиторія

Цільовою аудиторією даного проєкту є користувачі мережі Інтернет, що бажають мати змогу контактувати зі своїми рідними, друзями та колегами у віртуальному просторі.

2 Вимоги до проєкту

2.1 Вимоги до проєкту в цілому

2.1.1 Вимоги до структури й функціонування

Вебдодаток організації діяльності соціальної мережі «ConnectHub» повинен бути реалізований за допомогою web-інструментів та забезпечувати визначений набір функціональних можливостей.

Кінцевий продукт даного проєкту має бути представлений у вигляді вебдодатку соціальної мережі, який має багатофункціональний і в той же час вичерпний інтерфейс, виконує всі функції соціальної мережі, як-от: створення унікальних сторінок для користувачів, обмін приватними повідомленнями між користувачами, можливість написання дописів та їх коментування для користувачів.

2.1.2 Вимоги до персоналу

Персонал для підтримки мережі являє собою адміністратора та модераторів. Від персоналу не вимагається мати особливі технічні навички для роботи та підтримки вебдодатку. Єдиною вимогою є вміння користування персональним комп’ютером та браузером. Адміністратор додатково має здійснювати огляд та, у разі необхідності, налагодження бази даних і вебсерверу.

2.1.3 Вимоги до збереження інформації

Уся інформація, надана у вебдодатку, повинна зберігатися у базі даних реалізованій засобами системи управління базами даних MySQL.

2.1.4 Вимоги до розмежування доступ

Розроблюваний вебдодаток має бути загальнодоступним у мережі Інтернет. Права доступу до інформації розмежовані за групами користувачів: адміністратор, відвідувач та клієнт:

– Адміністратор має необмежений доступ до даних з правами перегляду, додавання, редагування та видалення інформаційного наповнення вебдодатку. Доступ до адміністративної панелі надається за спеціальним логіном та паролем.

– Відвідувач вебдодатку може тільки переглядати інформацію на сторінках та профілях інших користувачів і надсилати повідомлення за допомогою форми зворотнього зв’язку.

– У клієнта вебдодатку спектр доступу до інформації ширший за відвідувача, але менший за адміністратора. До переліку його можливостей входять ті, які визначені в групі користувачів «Відвідувач», та додатково доступ до редагування особистого профілю, створення, редагування та видалення власних дописів та коментарів.

2.2 Структура вебдодатку

2.2.1 Загальна інформація про структуру вебдодатку

До структури вебдодатку входять наступні сторінки:

– «Головна» сторінка містить навігаційне меню, яке закріплене на кожній сторінці вебдодатку, з можливістю переходу до особистого кабінету та стрічку з останніми дописами інших користувачів мережі;

– для переходу на сторінку «Особистого профілю» користувач може клікнути по закріпленій у навігаційній панелі кнопці «Увійти». Ця сторінка містить інформацію про особистий профіль користувача та можливість його редагування;

– сторінка «Приватних повідомлень» відображає усіх користувачів, з якими спілкується користувач, а також надає можливість відправляти повідомлення;

– на сторінці «Налаштування» користувач може здійснювати базові налаштування вебдодатку, редагувати тему, приватність профілю тощо;

– сторінка «Реєстрація» відкриється, коли користувач обере відповідну дію на формі, що відкривається при натисканні на кнопку «Увійти». Сторінка містить 4 обов'язкових поля для створення облікового запису: ім'я користувача, пошта, пароль та повторне введення для перевірки пароля;

– «Адміністративна панель» дозволяє особам з правами адміністратора редагувати наповнення вебдодатку та здійснювати моніторинг за користувачами.

2.2.2 Навігаційне меню

Для зручної навігації повинне бути створене меню, що забезпечить користувача можливістю зручно та швидко переміщатися по всім доступним сторінкам вебдодатку. Меню має бути закріплене і розташовуватися у бічній частині зліва кожної сторінки.

2.2.3 Управління контентом

Управління контентом вебдодатку має здійснюватися за допомогою адміністративної панелі та функцій створення, редагування та видалення власних дописів та коменарів для користувачів. Особи з доступом до адміністративної панелі можуть додавати, редагувати та видаляти наповнення вебдодатку. Уся інформація про користувачів вебдодатку має міститись у базі даних.

2.2.4 Дизайн вебдодатку

Дизайн вебдодатку має бути виконаний у мінімалістичному та сучасному стилі. Найбільш влучними кольорами для створення вебдодатку є білий, чорний та палітра відтінків синього. Тому під час розробки вебдодатку треба використовувати саме ці кольори.

Види і розміри шрифтів повинні бути комфортними для перегляду. Інформаційні блоки, графічні матеріали та інші елементи web-сторінок повинні мати зручне і логічне розташування. Шаблон розроблюваного вебдодатку наведено нижче (рис. А.1).

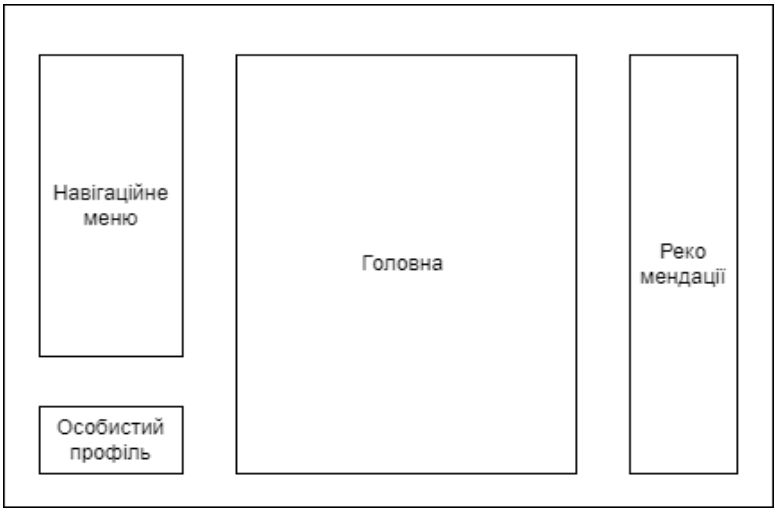


Рисунок А.1 – Схема головної сторінки

2.2.5 Система навігації (карта вебдодатку)

Карту вебдодатку наведено нижче (рис. А.2).

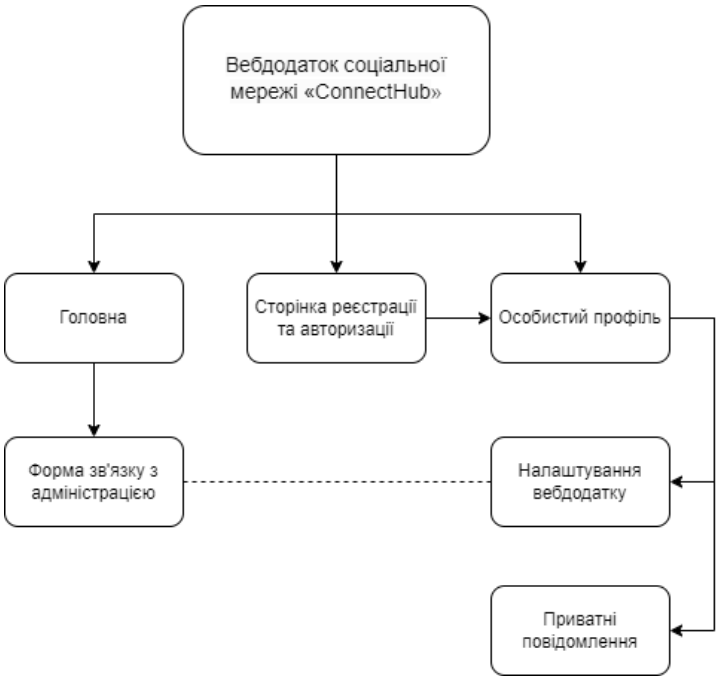


Рисунок А.2 – Система навігації

2.3 Вимоги до видів забезпечення

2.3.1 Вимоги до лінгвістичного забезпечення

Для забезпечення популярності та поширення вебдодатку серед користувачів усього світу, все текстове наповнення вебдодатку має бути виконане англійською мовою.

2.3.2 Вимоги до програмного забезпечення

Для забезпечення стабільної роботи вебдодатку версія web-браузера має бути: Internet Explorer 11 і вище, або Firefox 6 і вище, або Opera 15 і вище, або Safari 10 і вище, або Edge 12 і вище, або Chrome 20 і вище.

2.4 Вимоги до функціонування системи

2.4.1 Потреби користувача

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	Потреби користувача	Джерело
UN-01	Можливість реєстрації	Користувач, Відвідувач
UN-02	Можливість авторизації	Адміністратор, Користувач
UN-03	Перегляд наповнення головної сторінки	Адміністратор, Користувач, Відвідувач

Продовження таблиці А.1

ID	Потреби користувача	Джерело
UN-04	Перегляд інформації про інших користувачів	Адміністратор, Користувач, Відвідувач
UN-05	Доступ до редагування профілю	Адміністратор, Користувач
UN-06	Можливість заводити друзів серед інших користувачів вебдодатку	Користувач
UN-07	Можливість відправляти приватні повідомлення друзям	Адміністратор, Користувач
UN-08	Можливість налаштовувати профіль вебдодатку	Адміністратор, Користувач
UN-09	Можливість відправки форм зворотнього зв’язку до адміністрації вебдодатку	Користувач
UN-10	Додавання, видалення дописів та коментарів	Користувач
UN-11	Додавання, редагування, видалення дописів та коментарів	Адміністратор
UN-12	Можливість переглядати та відповідати на запити з форм зворотнього зв’язку	Адміністратор

У таблиці потреб користувача А.1 наведені 3 можливі категорії користувачів:

– Відвідувач – особа, що вперше відвідує вебдодаток або ще не реєструвалась на ньому;

– Користувач – особа, яка створила обліковий запис на сайті вебдодатку;

– Адміністратор – особа, що володіє правами доступу до адміністративної панелі.

2.4.2 Функціональні вимоги

Проаналізувавши потреби користувачів та персоналу закладу визначено наступні функціональні вимоги вебдодатку:

– можливість реєстрації та авторизації;

– можливість перегляду наповнення головної сторінки та профілів користувачів;

– можливість редагування особистого профілю;

– можливість створення та видалення дописів;

– можливість створення та видалення коментарів під дописами користувачів;

– можливість заводити друзів серед інших користувачів;

– можливість відправляти приватні повідомлення друзям;

– можливість змінювати налаштування вебдодатку;

– можливість відправляти форми зворотнього зв’язку до адміністрації вебдодатку;

– наявність панелі адміністратора для додавання, редагування та видалення наповнення вебдодатку;

– можливість відповідати на форми зворотнього зв’язку для адміністратора.

3 Склад і зміст робіт зі створення вебдодатку соціальної мережі «ConnectHub»

Опис робіт для створення вебдодатку соціальної мережі «ConnectHub» наведено в таблиці А.2.

Таблиця А.2 –Опис робіт для створення вебдодатку соціальної мережі «ConnectHub»

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Розробка дизайну вебдодатку	2 дні
2	Верстка сторінок вебдодатку	14 днів
3	Розробка бази даних	7 днів
4	Наповнення контентом вебдодатку	14 днів
5	Розробка модулю реєстрації та авторизації	7 днів
6	Розробка модулю створення дописів та їх коментування	7 днів
7	Розробка модулю приватних повідомлень	7 днів
8	Розробка модулю редагування налаштувань вебдодатку	7 днів
9	Alpha-тестування	5 днів
10	Beta-тестування	7 днів
11	Розміщення на хостингу	1 день
12	Збір відгуків про працездатність	2 дні
13	Перевірка та налагодження працездатності	3 дні
14	Написання супровідної документації	3 дні

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Розробка дизайну вебдодатку	2 дні
2	Верстка сторінок вебдодатку	14 днів
3	Розробка бази даних	7 днів
4	Наповнення контентом вебдодатку	14 днів
5	Розробка модулю реєстрації та авторизації	7 днів
6	Розробка модулю створення дописів та їх коментування	7 днів
7	Розробка модулю приватних повідомлень	7 днів
8	Розробка модулю редагування налаштувань вебдодатку	7 днів
9	Alpha-тестування	5 днів
15	Реліз вебдодатку	1 день
	Загальна тривалість робіт	87 днів

4 Вимоги до складу й змісту робіт із введення вебдодатку соціальної мережі «ConnectHub» в експлуатацію

Вебдодаток соціальної мережі «ConnectHub» має бути затверджено та розміщено на web-хостингу.

ДОДАТОК Б
ПЛАНУВАННЯ РОБІТ

Деталізація мети проєкту методом SMART. Для виконавця даного проєкту формат постановки SMART-мети такий: *«Метою роботи є розробка вебдодатку соціальної мережі з інтуїтивно зрозумілим інтерфейсом, що забезпечить користувачів стабільним зв'язком у віртуальному середовищі. Він дозволить створювати унікальні сторінки особистого профілю, заводити нові знайомства, підтримувати контакт з друзями та рідними, обговорювати новини та важливі суспільні питання».* Детальне представлення мети проєкту методом SMART продемонстровано у таблиці Б.1.

Таблиця Б.1 – Деталізація мети проєкту методом SMART

Specific	Розробити вебдодаток соціальної мережі з інтуїтивно зрозумілим інтерфейсом
Measurable	Створено вебдодаток, що відповідає вимогам, описаним у технічному завданні
Achievable	Технічне завдання затверджене й досяжне та наявні всі технологічні й програмні засоби для створення продукту
Relevant	Для забезпечення зручним взаємозв’язком користувачів у віртуальному просторі
Time-framed	До попереднього захисту дипломної роботи

Планування змісту робіт. WBS – це графічне подання згрупованих елементів проєкту у вигляді пакета робіт, які ієрархічно пов’язані з вебдодатком проєкту. На верхньому першому рівні WBS фіксується вебдодаток проєкту. Наступний рівень відповідає діям або основним заходам для досягнення мети проєкту. Подальша розбивка цих дій відбувається до моменту, поки не буде досягнуто елементарних робіт.

Елементарні роботи – це роботи, які мають один чіткий результат, призначеного одного конкретного відповідального. Зазвичай декомпозиція завершується тоді, коли для розкриття змісту потрібні вузькі фахівці, що знають технологічні особливості їх виконання. WBS-структура проєкту наведена нижче (рис. Б.1).

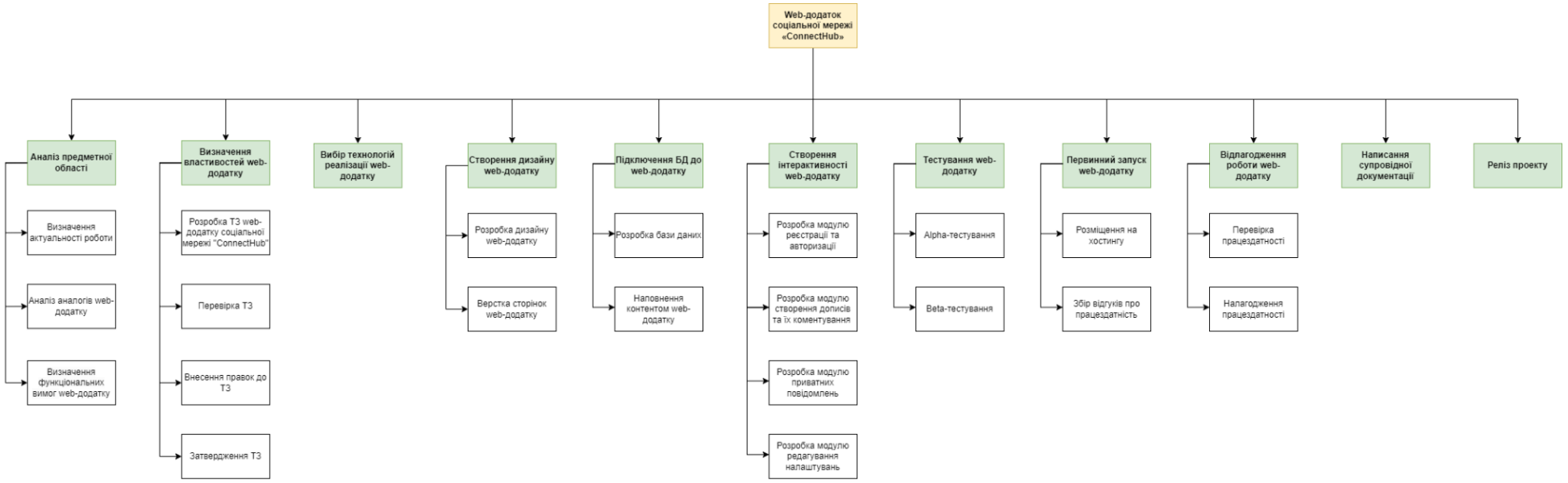


Рисунок Б.1 – WBS-структура планування змісту робіт проєкту

Планування структури виконавців. OBS-структура проєкту – організаційна структура виконавців (організацій) проєкту. Визначається за переліком пакетів робіт нижнього рівня кожної гілки WBS-структури. Організаційна структура представляє собою графічне відображення учасників проєкту та їх відповідальних осіб, які задіяні в реалізації проєкту.

На верхньому рівні OBS розташована команда проєкту. На наступному рівні фіксуються виконавці: організації, відділи тощо. Потім, рівнем нижче, для кожного виконавця вказують прізвища конкретних осіб, які будуть відповідати за виконання елементарних робіт WBS. Це не обов'язково керівники, але ті співробітники, які безпосередньо організовують і відповідають за виконання елементарної роботи, зазначеної у WBS. OBS-структуру проєкту зображено на рисунку нижче (рис. Б.2). Список виконавців проєкту представлено у таблиці Б.2.

Таблиця Б.2 – Виконавці проєкту

Роль	**ПІБ**	**Проєктна роль**
Розробник	Денисенко Ф.М.	Виконує front-end та back-end розробку
Дизайнер	Денисенко Ф.М.	Розроблює дизайн вебдодатку
Тестувальник	Денисенко Ф.М.	Відповідає за тестування функціоналу вебдодатку
Менеджер	Денисенко Ф.М.	Відповідає за виконання термінів, розподіл ресурсів та завдань між учасниками. Виконує збір та аналіз даних.
Керівник	Бойко О. В..	Формує завдання на розробку проєкту

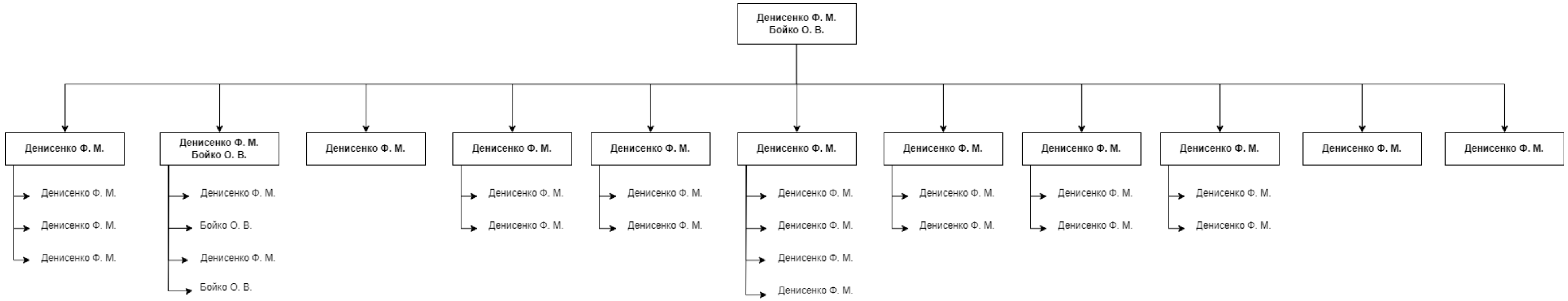
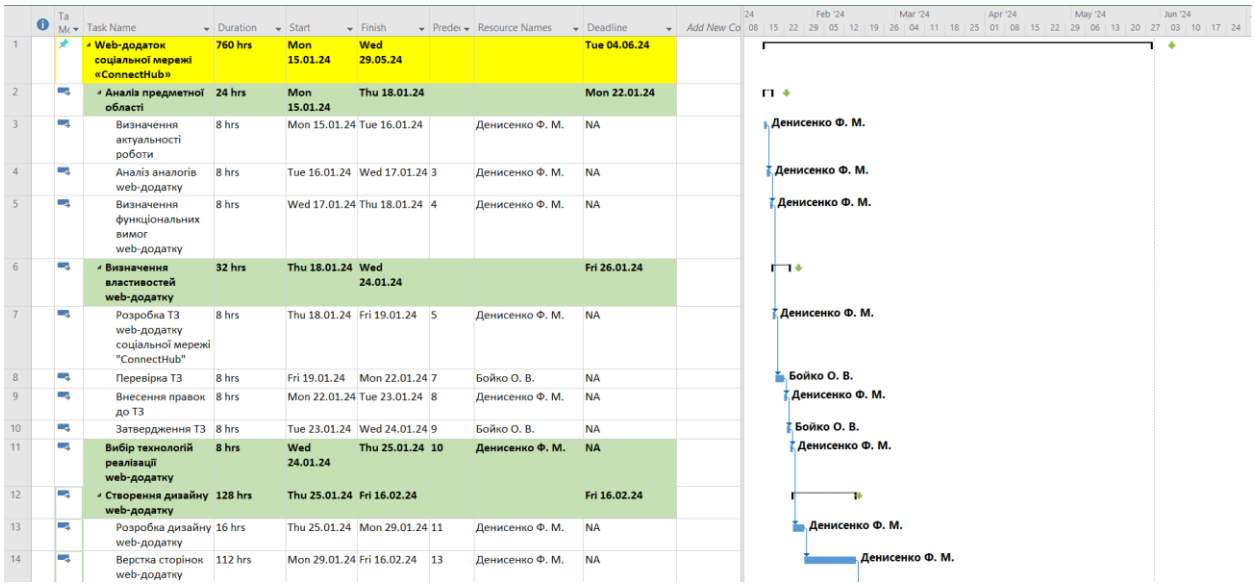


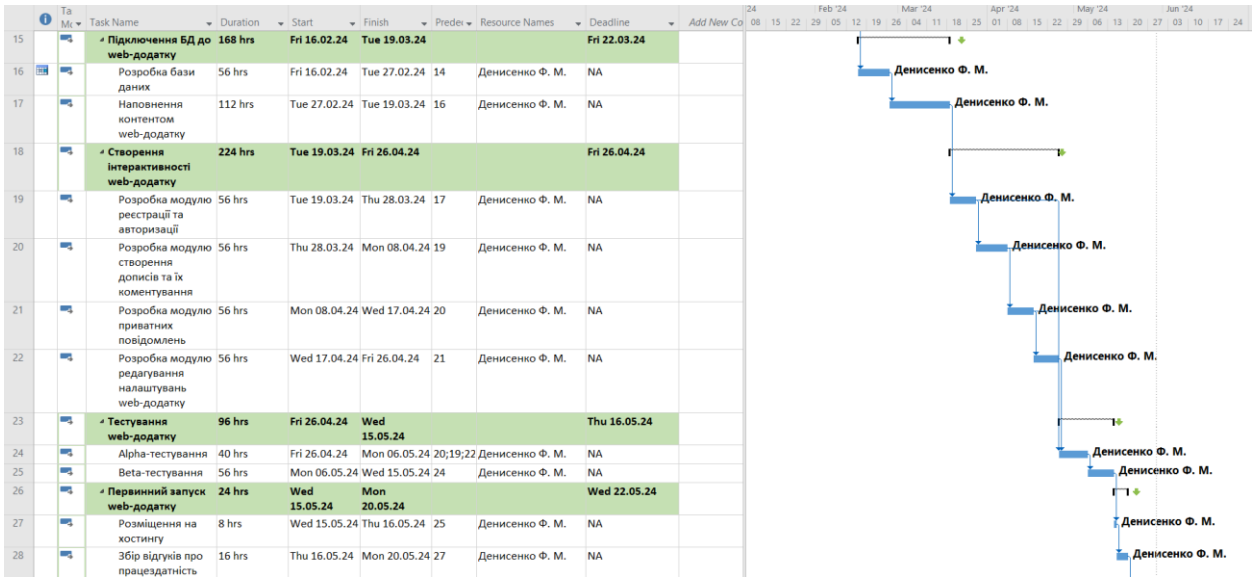
Рисунок Б.2 – OBS-структура планування організації проєкту

Діаграма Ганта. Побудова календарного графіку (діаграми Ганта) є одним з важливих етапів планування проєкту, що виглядає як розклад виконання робіт з реальним розподілом дат. Завдяки йому можна отримати достовірне уявлення про тривалість процесів з обмеженнями у ресурсах, урахуванням вихідних днів та свят (рис. Б.3-Б.5).



	Task Name	Duration	Start	Finish	Predec	Resource Names	Deadline
1	**Web-додаток соціальної мережі «ConnectHub»**	**760 hrs**	**Mon 15.01.24**	**Wed 29.05.24**			**Tue 04.06.24**
2	**Аналіз предметної області**	**24 hrs**	**Mon 15.01.24**	**Thu 18.01.24**			**Mon 22.01.24**
3	Визначення актуальності роботи	8 hrs	Mon 15.01.24	Tue 16.01.24		Денисенко Ф. М.	NA
4	Аналіз аналогів web-додатку	8 hrs	Tue 16.01.24	Wed 17.01.24	3	Денисенко Ф. М.	NA
5	Визначення функціональних вимог web-додатку	8 hrs	Wed 17.01.24	Thu 18.01.24	4	Денисенко Ф. М.	NA
6	**Визначення властивостей web-додатку**	**32 hrs**	**Thu 18.01.24**	**Wed 24.01.24**			**Fri 26.01.24**
7	Розробка ТЗ web-додатку соціальної мережі "ConnectHub"	8 hrs	Thu 18.01.24	Fri 19.01.24	5	Денисенко Ф. М.	NA
8	Перевірка ТЗ	8 hrs	Fri 19.01.24	Mon 22.01.24	7	Бойко О. В.	NA
9	Внесення правок до ТЗ	8 hrs	Mon 22.01.24	Tue 23.01.24	8	Денисенко Ф. М.	NA
10	Затвердження ТЗ	8 hrs	Tue 23.01.24	Wed 24.01.24	9	Бойко О. В.	NA
11	**Вибір технологій реалізації web-додатку**	**8 hrs**	**Wed 24.01.24**	**Thu 25.01.24**	**10**	**Денисенко Ф. М.**	**NA**
12	**Створення дизайну web-додатку**	**128 hrs**	**Thu 25.01.24**	**Fri 16.02.24**			**Fri 16.02.24**
13	Розробка дизайну web-додатку	16 hrs	Thu 25.01.24	Mon 29.01.24	11	Денисенко Ф. М.	NA
14	Верстка сторінок web-додатку	112 hrs	Mon 29.01.24	Fri 16.02.24	13	Денисенко Ф. М.	NA

Рисунок Б.3 – Діаграма Ганта проєкту. Частина 1



	Task Name	Duration	Start	Finish	Predec	Resource Names	Deadline
15	**Підключення БД до web-додатку**	**168 hrs**	**Fri 16.02.24**	**Tue 19.03.24**			**Fri 22.03.24**
16	Розробка бази даних	56 hrs	Fri 16.02.24	Tue 27.02.24	14	Денисенко Ф. М.	NA
17	Наповнення контентом web-додатку	112 hrs	Tue 27.02.24	Tue 19.03.24	16	Денисенко Ф. М.	NA
18	**Створення інтерактивності web-додатку**	**224 hrs**	**Tue 19.03.24**	**Fri 26.04.24**			**Fri 26.04.24**
19	Розробка модулю реєстрації та авторизації	56 hrs	Tue 19.03.24	Thu 28.03.24	17	Денисенко Ф. М.	NA
20	Розробка модулю створення дописів та їх коментування	56 hrs	Thu 28.03.24	Mon 08.04.24	19	Денисенко Ф. М.	NA
21	Розробка модулю приватних повідомлень	56 hrs	Mon 08.04.24	Wed 17.04.24	20	Денисенко Ф. М.	NA
22	Розробка модулю редагування налаштувань web-додатку	56 hrs	Wed 17.04.24	Fri 26.04.24	21	Денисенко Ф. М.	NA
23	**Тестування web-додатку**	**96 hrs**	**Fri 26.04.24**	**Wed 15.05.24**			**Thu 16.05.24**
24	Alpha-тестування	40 hrs	Fri 26.04.24	Mon 06.05.24	20;19;22	Денисенко Ф. М.	NA
25	Beta-тестування	56 hrs	Mon 06.05.24	Wed 15.05.24	24	Денисенко Ф. М.	NA
26	**Первинний запуск web-додатку**	**24 hrs**	**Wed 15.05.24**	**Mon 20.05.24**			**Wed 22.05.24**
27	Розміщення на хостингу	8 hrs	Wed 15.05.24	Thu 16.05.24	25	Денисенко Ф. М.	NA
28	Збір відгуків про працездатність	16 hrs	Thu 16.05.24	Mon 20.05.24	27	Денисенко Ф. М.	NA

Рисунок Б.4 – Діаграма Ганта проєкту. Частина 2

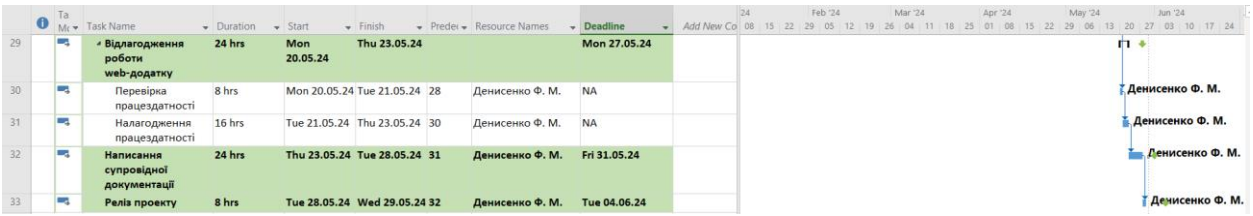


Рисунок Б.5 – Діаграма Ганта проєкту. Частина 3

Управління ризиками проєкту. Під час виконання якісної оцінки ризиків треба визначити ті, які мають бути усунені якнайшвидше. В залежності від ступеня важливості ризику – реагування буде відповідне. Наступним етапом є виконання кількісного оцінювання. У таблиці Б.3 надано перелік ризиків даного проєкту. Результати оцінки ризиків надано у таблиці Б.4. Таблиця Б.5 представляє шкалу для класифікації ризиків за величиною впливу на проєкт та ймовірністю їх виникнення.

Таблиця Б.3 – Ризики проєкту

№ ризику	Назва (опис) ризику
1	Масові відключення електроживлення
2	Глобальна відсутність доступу до мережі Інтернет
3	Нестача часу для вчасного випуску вебдодатку
4	Некоректна робота певних модулей вебдодатку
5	Злам у роботі хостингу
6	Пандемія
7	Нездужання розробника
8	Некоректне розуміння вимог
9	Несумісність коду вебдодатку з браузером

№ ризику	Назва (опис) ризику
10	Повітряні тривоги

Таблиця Б.4 – Визначення ймовірності, впливу та рангу ризиків проєкту

№ ризику	Назва (опис) ризику	Ймовірність (0,1-0,9)	Вплив (0,05-0,8)	Ранг
1	Масові відключення електроживлення	0.5	0.2	0.1
2	Глобальна відсутність доступу до мережі Інтернет	0.1	0.8	0.08
3	Нестача часу для вчасного випуску продукту	0.3	0.4	0.12
4	Некоректна робота певних модулей продукту	0.3	0.4	0.12
5	Злам у роботі хостингу	0.3	0.4	0.12
6	Пандемія	0.1	0.8	0.08
7	Нездужання розробника	0.5	0.2	0.1
8	Некоректне розуміння вимог	0.1	0.4	0.04
9	Несумісність коду продукту з браузером	0.5	0.2	0.1
10	Повітряні тривоги	0.9	0.1	0.09

Таблиця Б.5 – Шкала оцінювання ризиків за ймовірністю виникнення та величиною впливу

Оцінка	**Ймовірність виникнення**	**Вплив ризику**	**Тип ризику**
1	Низька	Низький	Прийнятний
2	Середня	Середній	Виправданий
3	Висока	Високий	Недопустимий

Для того, щоб знизити негативний вплив ризиків на проєкт треба виконати планування реагування на них. До нього входить оцінка наслідків впливу на проєкт і розробка відповідних заходів. Аналіз виконується за показниками, які описані в таблиці Б.4. У результаті планування заходів реагування на ризики проєкту отримано матрицю ймовірності виникнення та впливу ризиків, що наведена у таблиці Б.6. Зеленим кольором на матриці позначають прийнятні ризики, жовтим – виправдані, а червоним – недопустимі.

Таблиця Б.6 – Матриця ймовірності та впливу згідно проєкту

Ймовірність ризику (Й)	**Вплив загрози (ризику)**				
	Дуже малий	**Малий**	**Середній**	**Великий**	**Дуже великий**
	0,05	0,1	0,2	0,4	0,8
0,9	0.045	0.09 **R10**	0.18	0.36	0.72
0,7	0.035	0.07	0.14	0.28	0.56
0,5	0.025	0.05	0.1 **R1, R7, R9**	0.15	0.4
0,3	0.015	0.03	0.06	0.12 **R3, R4, R5**	0.24
0,1	0.005	0.01	0.01	0.04 **R8**	0.08 **R2, R6**

Класифікація ризиків проєкту за рівнем, відповідно до отриманого значення індексу, представлена у таблиці Б.7.

Таблиця Б.7 – Шкала оцінювання за рівнем ризику

№	**Назва**	**Межі**	**Ризики, які входять (номера)**
1	Прийнятні	0,005≤R≤0,05	8
2	Виправдані	0,05<R≤0,14	1, 2, 3, 4, 5, 6, 7, 9, 10
3	Недопустимі	0,14<R≤0,72	

У таблиці Б.8 описано усі ризики та стратегії реагування на кожен із них.

Таблиця Б.8 – Ризики проєкту та стратегії реагування

ID ризику	**Статус ризику**	**Опис ризику**	**Ймовірність виникнення**	**Вплив ризику**	**Ранг ризику**	**Тип стратегії реагування (який може бути застосовано до ризику)**	**План А (заходи запобігання виникнення ризику)**	**План Б (заходи усунення наслідків ризику)**
1	NEW	Масові відключення електроживлення	0.5	0.2	0.1	Зменшення	Впровадження альтернативного джерела живлення	Перехід на альтернативне джерело живлення
2	NEW	Глобальна відсутність доступу до мережі Інтернет	0.1	0.8	0.08	Зменшення	Регулярна перевірка стабільності мережі	Відкладення розробки на невизначений термін
3	NEW	Нестача часу для вчасного випуску вебдодатку	0.3	0.4	0.12	Зменшення	Формування проєкту з запасом по часу	Обмеження функціональності вебдодатку
4	NEW	Некоректна робота певних модулей вебдодатку	0.3	0.4	0.12	Зменшення	Написання тестів для перевірки працездатності	Тестування та виділення часу на відлагодження

Продовження таблиці Б.8

ID ризику	**Статус ризику**	**Опис ризику**	**Ймовірність виникнення**	**Вплив ризику**	**Ранг ризику**	**Тип стратегії реагування (який може бути застосовано до ризику)**	**План А (заходи запобігання виникнення ризику)**	**План Б (заходи усунення наслідків ризику)**
5	NEW	Злам у роботі хостингу	0.3	0.4	0.12	Зменшення	Розміщення вебсайту на декількох хостингах	Швидке перемикання на інший хостинг
6	NEW	Пандемія	0.1	0.8	0.08	Зменшення	Моніторинг новин стану вірусних захворювань	Створення можливостей працювати віддалено
7	NEW	Нездужання розробника	0.5	0.2	0.1	Зменшення	Виділення часу на регулярний медичний огляд	Використання запасу по часу для надолуження роботи
8	NEW	Некоректне розуміння вимог	0.1	0.4	0.04	Зменшення	Складання чіткої структури робіт	Використання резерву часу для налагодження помилок

Продовження таблиці Б.8

ID ризику	**Статус ризику**	**Опис ризику**	**Ймовірність виникнення**	**Вплив ризику**	**Ранг ризику**	**Тип стратегії реагування (який може бути застосовано до ризику)**	**План А (заходи запобігання виникнення ризику)**	**План Б (заходи усунення наслідків ризику)**
9	NEW	Несумісність коду вебдодатку з браузером	0.5	0.2	0.1	Зменшення	Використання транспіляторів коду	Рефакторинг та впровадження поліфілів
10	NEW	Повітряні тривоги	0.9	0.1	0.09	Зменшення	Переїзд у місце, віддалене від бойових дій	Здійснення евакуації

ДОДАТОК В

КОД РЕАЛІЗАЗІЇ ВЕБДОДАТКУ СОЦІАЛЬНОЇ МЕРЕЖІ «CONNECTHUB»

Src/

Директорія, де розміщуються всі компоненти вебдодатку.

Index.js

```
import './routes/routes.js';
import { initRoutes } from './routes/routes.js';
import express from 'express';
import 'express-async-errors';
import fileUpload from 'express-fileupload';
import { dirname, join } from 'node:path';
import { fileURLToPath } from 'node:url';

const app = express();
const port = 3080;

const __dirname = dirname(fileURLToPath(import.meta.url));

app.use(express.json());
app.use(fileUpload());
app.set('views', join(__dirname, 'views'));
app.set('view engine', 'pug');
app.use(express.static(join(__dirname, 'static')));

initRoutes(app);

app.listen(port, () => {
  console.log(`Listening on port ${port}`);
});
```

Src/Routes/

Кінцеві шляхи для обробки запитів до вебдодатку.

Routes.js

```
import sighUpRouter from './singup.js';
import loginRouter from './login.js';
import homeRouter from './home.js';
import chatRouter from './chats.js';
import settingsRouter from './settings.js';
import userRouter from './user.js';
import contactRouter from './contact.js';
import apiRouter from './api.js';

const initRoutes = (app) => {
  app.use('/signup', sighUpRouter);
  app.use('/login', loginRouter);
  app.use('/home', homeRouter);
  app.use('/chats', chatRouter);
  app.use('/settings', settingsRouter);
  app.use('/user', userRouter);
  app.use('/contact', contactRouter);
  app.use('/api', apiRouter);
  app.get('/', (req, res) => {
    res.redirect('/home');
  });
  app.use((req, res) => {
    res.status(404).render('errors/404');
  });
};

export { initRoutes };
```

Chats.js

```
import express from 'express';
const router = express.Router();

router.get('/', (req, res) => {
  res.render('chats');
});

export default router;
```

Contact.js

```
import express from 'express';
const router = express.Router();

router.get('/', (req, res) => {
  res.render('contact');
});
export default router;
```

Home.js

```
import express from 'express';
import asyncHandler from 'express-async-handler';
import { getPosts } from '../services/post-service.js';
import { getPostComments } from '../services/comment-service.js';
import { getUser } from '../services/user-service.js';
const router = express.Router();

async function getAuthorInfo(authorId) {
  const author = await getUser(authorId);

  const { avatar, first_name, second_name, nickname } = author;

  return {
    author_avatar: avatar,
    author_first_name: first_name,
    author_second_name: second_name,
    author_nickname: nickname,
  };
}

router.get(
  '/',
  asyncHandler(async (req, res) => {

    const posts = await getPosts();

    const data = {};

    data.posts = await Promise.all(
      posts.map(async (post) => {
        const postAuthor = await getAuthorInfo(post.author_id);
        post = { ...post, ...postAuthor };

        const comments = await getPostComments(post.post_id);

        const postComments = await Promise.all(
          comments.map(async (comment) => {
            const commentAuthor = await getAuthorInfo(comment.author_id);

            comment = { ...comment, ...commentAuthor };

            return comment;
          })
        );

        post.comments = postComments;

        return post;
      })
    );

    res.render('home', data);
  })
);

export default router;
```

Login.js

```
import express from 'express';
const router = express.Router();

router.get('/', (req, res) => {
  res.render('login');
});

export default router;
```

Settings.js

```
import express from 'express';
import { deleteUser } from '../services/user-service.js';
const router = express.Router();

router.get('/', (req, res) => {
  res.render('settings');
});

router.put('/', (req, res) => {});

router.delete('/', deleteUser, (req, res) => {});
export default router;
```

Signup.js

```
import express from 'express';
import { validateUser } from '../middlewares/user-validation.js';
import { getAvatarPath } from '../middlewares/file-path.js';
import { hashPassword } from '../middlewares/password-hash.js';
import asyncHandler from 'express-async-handler';
import { createUser } from '../services/user-service.js';

const router = express.Router();

router.get('/', (req, res) => {
  res.render('signup');
});

router.post(
  '/',
  validateUser,
  getAvatarPath,
  hashPassword,
  asyncHandler(async (req, res) => {
    const insertData = await createUser(res.user);
    res.redirect(`/user/${insertData.insertId}`);
  })
);

export default router;
```

User.js

```
import express from 'express';
import asyncHandler from 'express-async-handler';
import { getUser } from '../services/user-service.js';
import { getUserPosts } from '../services/post-service.js';
import { getUserFriends } from '../services/friendship-service.js';
import { getUserComments } from '../services/comment-service.js';
const router = express.Router();

router.get(
  '/:id',
  asyncHandler(async (req, res) => {
    const userId = req.params.id;
    const user = await getUser(userId);

    if (!user) {
      res.render('errors/404');
      return;
    }

    user.posts = await getUserPosts(userId);
    user.comments = await getUserComments(userId);
    const friendIds = await getUserFriends(userId);
    user.friends = await Promise.all(
      friendIds.map(async (friendId) => {
        const userId = friendId.user_id;
        return await getUser(userId);
      })
    );

    res.render('user', user);
  })
);
export default router;
```

Api.js

```
import express from 'express';
import { handleServerError } from '../middlewares/error-handler.js';
import usersApi from './api/users-api.js';
import postsApi from './api/posts-api.js';
import commentsApi from './api/comments-api.js';
import formsApi from './api/forms-api.js';
import friendshipsApi from './api/friendships-api.js';
import chatsApi from './api/chats-api.js';
import messagesApi from './api/messages-api.js';
const router = express.Router();

router.use('/users', usersApi);
router.use('/posts', postsApi);
router.use('/comments', commentsApi);
router.use('/forms', formsApi);
router.use('/friendships', friendshipsApi);
router.use('/chats', chatsApi);
router.use('/messages', messagesApi);

router.use(handleServerError);
export default router;
```

Src/Routes/Api/

Прикладний програмний інтерфейс вебдодатку.

Chats-api.js

```
import express from 'express';
import asyncHandler from 'express-async-handler';
import { responseMiddleware } from '../../middlewares/response-middleware.js';
import { createChat, getChat, getChats } from '../../services/chat-service.js';
import { getChatMessages } from '../../services/message-service.js';
const router = express.Router();

router.get(
  '/',
  asyncHandler(async (req, res) => {
    const chats = await getChats();
    res.json(chats);
  }),
  responseMiddleware
);

router.get(
  '/:id',
  asyncHandler(async (req, res) => {
    const chatId = req.params.id;

    const chat = await getChat(chatId);
    res.json(chat);
  }),
  responseMiddleware
);

router.get(
  '/:id/messages',
  asyncHandler(async (req, res) => {
    const chatId = req.params.id;

    const chatMessages = await getChatMessages(chatId);
    res.json(chatMessages);
  }),
  responseMiddleware
);

router.post(
  '/',
  asyncHandler(async (req, res) => {
    res.chat = req.body;

    const insertData = await createChat(res.chat);
    const chat = await getChat(insertData.insertId);
    res.json(chat);
  }),
  responseMiddleware
);

export default router;
```

Comments-api.js

```
import express from 'express';
import asyncHandler from 'express-async-handler';
import { responseMiddleware } from '../../middlewares/response-middleware.js';
import {
  createComment,
  getComment,
  getComments,
} from '../../services/comment-service.js';
const router = express.Router();

router.get(
  '/',
  asyncHandler(async (req, res) => {
    const comments = await getComments();
    res.json(comments);
  }),
  responseMiddleware
);

router.get(
  '/:id',
  asyncHandler(async (req, res) => {
    const commentId = req.params.id;

    const comment = await getComment(commentId);
    res.json(comment);
  }),
  responseMiddleware
);

router.post(
  '/',
  asyncHandler(async (req, res) => {
    res.comment = req.body;

    const insertData = await createComment(res.comment);
    const comment = await getComment(insertData.insertId);
    res.json(comment);
  }),
  responseMiddleware
);

export default router;
```

Forms-api.js

```
import express from 'express';
import asyncHandler from 'express-async-handler';
import { responseMiddleware } from '../../middlewares/response-middleware.js';
import { createForm, getForm, getForms } from '../../services/form-service.js';
const router = express.Router();

router.get(
  '/',
  asyncHandler(async (req, res) => {
    const forms = await getForms();
    res.json(forms);
```

```
  }),
  responseMiddleware
);

router.get(
  '/:id',
  asyncHandler(async (req, res) => {
    const formId = req.params.id;

    const form = await getForm(formId);
    res.json(form);
  }),
  responseMiddleware
);

router.post(
  '/',
  asyncHandler(async (req, res) => {
    res.form = req.body;
    // upload file

    const insertData = await createForm(res.form);
    const form = await getForm(insertData.insertId);
    res.json(form);
  }),
  responseMiddleware
);

export default router;
```

Friendships-api.js

```
import express from 'express';
import asyncHandler from 'express-async-handler';
import { responseMiddleware } from '../../middlewares/response-middleware.js';
import {
  createFriendship,
  getFriendship,
  getFriendships,
} from '../../services/friendship-service.js';
const router = express.Router();

router.get(
  '/',
  asyncHandler(async (req, res) => {
    const friendships = await getFriendships();
    res.json(friendships);
  }),
  responseMiddleware
);

router.get(
  '/:id',
  asyncHandler(async (req, res) => {
    const friendshipId = req.params.id;

    const friendship = await getFriendship(friendshipId);
    res.json(friendship);
  }),
  responseMiddleware
```

```
);

router.post(
  '/',
  asyncHandler(async (req, res) => {
    res.friendship = req.body;

    const insertData = await createFriendship(res.friendship);
    const friendship = await getFriendship(insertData.insertId);
    res.json(friendship);
  }),
  responseMiddleware
);

export default router;
```

Messages-api.js

```
import express from 'express';
import asyncHandler from 'express-async-handler';
import { responseMiddleware } from '../../middlewares/response-middleware.js';
import {
  createMessage,
  getMessage,
  getMessages,
} from '../../services/message-service.js';
const router = express.Router();

router.get(
  '/',
  asyncHandler(async (req, res) => {
    const messages = await getMessages();
    res.json(messages);
  }),
  responseMiddleware
);

router.get(
  '/:id',
  asyncHandler(async (req, res) => {
    const messageId = req.params.id;

    const message = await getMessage(messageId);
    res.json(message);
  }),
  responseMiddleware
);

router.post(
  '/',
  asyncHandler(async (req, res) => {
    res.message = req.body;

    const insertData = await createMessage(res.message);
    const message = await getMessage(insertData.insertId);
    res.json(message);
  }),
  responseMiddleware
);
export default router;
```

Posts-api.js

```
import express from 'express';
import { createPost, getPost, getPosts } from '../../services/post-service.js';
import asyncHandler from 'express-async-handler';
import { responseMiddleware } from '../../middlewares/response-middleware.js';
import { getPostComments } from '../../services/comment-service.js';
const router = express.Router();

router.get(
  '/',
  asyncHandler(async (req, res) => {
    const posts = await getPosts();
    res.json(posts);
  }),
  responseMiddleware
);

router.get(
  '/:id',
  asyncHandler(async (req, res) => {
    const postId = req.params.id;

    const post = await getPost(postId);
    res.json(post);
  }),
  responseMiddleware
);

router.get(
  '/:id/comments',
  asyncHandler(async (req, res) => {
    const postId = req.params.id;

    const comments = await getPostComments(postId);
    res.json(comments);
  }),
  responseMiddleware
);

router.post(
  '/',
  asyncHandler(async (req, res) => {
    res.post = req.body;

    const insertData = await createPost(res.post);
    const post = await getPost(insertData.insertId);
    res.json(post);
  }),
  responseMiddleware
);

export default router;
```

Users-api.js

```
import express from 'express';
import {
  createUser,
```

```
  getUser,
  getUsers,
  updateUser,
} from '../../services/user-service.js';
import { validateUser } from '../../middlewares/user-validation.js';
import { getAvatarPath } from '../../middlewares/file-path.js';
import { hashPassword } from '../../middlewares/password-hash.js';
import { responseMiddleware } from '../../middlewares/response-middleware.js';
import asyncHandler from 'express-async-handler';
import { getUserFriends } from '../../services/friendship-service.js';
import { getUserPosts } from '../../services/post-service.js';
import { getUserComments } from '../../services/comment-service.js';
import { getUserChats } from '../../services/chat-service.js';
import { getUserChatMessages } from '../../services/message-service.js';
const router = express.Router();

router.get(
  '/',
  asyncHandler(async (req, res) => {
    const users = await getUsers();
    res.json(users);
  }),
  responseMiddleware
);

router.get(
  '/:id',
  asyncHandler(async (req, res) => {
    const userId = req.params.id;

    const user = await getUser(userId);
    res.json(user);
  }),
  responseMiddleware
);

router.get(
  '/:id/friends',
  asyncHandler(async (req, res) => {
    const userId = req.params.id;

    const userFriends = await getUserFriends(userId);
    res.json(userFriends);
  }),
  responseMiddleware
);

router.get(
  '/:id/chats',
  asyncHandler(async (req, res) => {
    const userId = req.params.id;

    const userChats = await getUserChats(userId);
    res.json(userChats);
  }),
  responseMiddleware
);

router.get(
  '/:userId/chats/:chatId/messages',
  asyncHandler(async (req, res) => {
    const userId = req.params.userId;
```

```
    const chatId = req.params.chatId;

    const userChats = await getUserChatMessages(userId,  chatId);
    res.json(userChats);
  }),
  responseMiddleware
);

router.get(
  '/:id/posts',
  asyncHandler(async (req, res) => {
    const userId = req.params.id;

    const userPosts = await getUserPosts(userId);
    res.json(userPosts);
  }),
  responseMiddleware
);

router.get(
  '/:id/comments',
  asyncHandler(async (req, res) => {
    const userId = req.params.id;

    const userComments = await getUserComments(userId);
    res.json(userComments);
  }),
  responseMiddleware
);

router.post(
  '/',
  validateUser,
  getAvatarPath,
  hashPassword,
  asyncHandler(async (req, res) => {
    const insertData = await createUser(res.user);
    const user = await getUser(insertData.insertId);
    res.json(user);
  }),
  responseMiddleware
);

router.put(
  '/:id',
  asyncHandler(async (req, res) => {
    const userId = req.params.id;
    const userData = req.body;
    debugger;

    const user = await updateUser(userId);
    res.json(user);
  }),
  responseMiddleware
);

export default router;
```

Src/Middleware/

Функції-посередники для обробки запитів на вебсервері.

Error-handler.js

```
export const handleServerError = (err, req, res, next) => {
  console.error(err.stack);
  res.status(500).render('errors/500');
};
```

File-path.js

```
import asyncHandler from 'express-async-handler';
import { uploadFile } from '../helpers/file-upload.js';

export const getAvatarPath = asyncHandler(async (req, res, next) => {
  const avatar = req.files?.avatar;

  if (!avatar) {
    res.user.avatar = 'default_user_avatar.jpeg';
  } else {
    res.user.avatar = await uploadFile(avatar, '/../static/avatars');
  }

  next();
});

export const getFilePath = asyncHandler(async (req, res, next) => {
  const file = req.params?.file;

  if (file) res.user.file = await uploadFile(file, '/../static/files');

  next();
});
```

Response-middleware.js

```
const responseMiddleware = (err, req, res, next) => {
  res.status(500).json({ error: true, message: err.message });
};

export { responseMiddleware };
```

User-validation.js

```
import asyncHandler from 'express-async-handler';

function hasExtraFields(user, maxFields) {
  const keys = Object.keys(user);

  if (keys.length > maxFields) {
```

```
    throw new Error('Extra fields are not allowed');
  }
}

function isEmailValid(email) {
  const emailRegExp = /^\S{3,}@\.com$/;
  if (!emailRegExp.test(email)) {
    throw new Error('Email is not valid');
  }
}

function isPasswordValid(password) {
  const passwordRegExp = /^\S{6,}/;
  if (!passwordRegExp.test(password)) {
    throw new Error('Password is not valid');
  }
}

function isEveryFieldValid({ email, pass }) {
  if (email) isEmailValid(email);
  if (pass) isPasswordValid(pass);
}

export const validateUser = asyncHandler(async (req, res, next) => {
  const user = req.body;

  res.user = user;
  next();
});
```

Src/Services

Сервіси, або ж функції доступу до бази даних, як-от: запит на дані, їх додавання, редагування та видалення

Chat-service.js

```
import { query } from '../helpers/database-query.js';

export const getChats = async () => {
  const sql = `SELECT * FROM chats`;

  const result = await query(sql);

  return result;
};

export const getUserChats = async (id) => {
  const sql = `
  SELECT c.chat_id, f.source_id AS 'friend_id'
  FROM chats c JOIN friendships f
  WHERE c.friendship_id = f.friendship_id AND f.target_id = ?
  UNION
  SELECT c.chat_id, f.target_id AS 'friend_id'
  FROM chats c JOIN friendships f
  WHERE c.friendship_id = f.friendship_id AND f.source_id = ?`;

  const result = await query(sql, [id, id]);

  return result;
};

export const getChat = async (id) => {
  const sql = `SELECT * FROM chats WHERE chat_id = ?`;

  const result = await query(sql, id);

  return result[0];
};

export const createChat = async (chat) => {
  const sql = `
  INSERT INTO chats (friendship_id)
  VALUES (?)`;

  const values = [chat.friendshipId];

  const result = await query(sql, values);

  return result;
};

export const updateChat = async (chat) => {
  const sql = `
  UPDATE chats
  SET friendship_id=?
  WHERE chat_id=?`;
```

```
  const values = [chat.friendshipId, chat.id];

  const result = await query(sql, values);

  return result;
};

export const deleteChat  = async (id) => {
  const sql = `DELETE FROM chats WHERE chat_id = ?`;

  const result = await query(sql, id);

  return result;
};
```

Comment-service.js

```
import { query } from '../helpers/database-query.js';

export const getComments  = async () => {
  const sql = `SELECT * FROM comments ORDER BY creation_time DESC`;

  const result = await query(sql);

  return result;
};

export const getPostComments  = async (id) => {
  const sql = `SELECT * FROM comments WHERE post_id = ?`;

  const result = await query(sql, id);

  return result;
};

export const getUserComments  = async (id) => {
  const sql = `SELECT * FROM comments WHERE author_id = ? ORDER BY creation_time DESC`;

  const result = await query(sql, id);

  return result;
};

export const getComment  = async (id) => {
  const sql = `SELECT * FROM comments WHERE comment_id = ?`;

  const result = await query(sql, id);

  return result[0];
};

export const createComment  = async (comment) => {
  const sql = `
  INSERT INTO comments (text, attachment, author_id, post_id)
  VALUES (?, ?, ?, ?)`;

  const values = [
    comment.text,
    comment.attachment,
    comment.authorId,
```

```
    comment.postId,
  ];

  const result = await query(sql, values);

  return result;
};

export const updateComment = async (comment) => {
  const sql = `
  UPDATE comments
  SET text=?,attachment=?,author_id=?,post_id=?
  WHERE comment_id=?`;

  const values = [
    comment.text,
    comment.attachment,
    comment.authorId,
    comment.postId,
    comment.id,
  ];

  const result = await query(sql, values);

  return result;
};

export const likeComment = async (id) => {
  const sql = `
  UPDATE comments
  SET likes_number = likes_number + 1
  WHERE comment_id=?`;

  const result = await query(sql, id);

  return result;
};

export const unlikeComment = async (id) => {
  const sql = `
  UPDATE comments
  SET likes_number = likes_number - 1
  WHERE comment_id=?`;

  const result = await query(sql, id);

  return result;
};

export const deleteComment = async (id) => {
  const sql = `DELETE FROM comments WHERE comment_id = ?`;

  const result = await query(sql, id);

  return result;
};
```

Form-service.js

```
import { query } from '../helpers/database-query.js';
```

```
export const getForms = async () => {
  const sql = `SELECT * FROM forms`;

  const result = await query(sql);

  return result;
};

export const getForm = async (id) => {
  const sql = `SELECT * FROM forms WHERE form_id = ?`;

  const result = await query(sql, id);

  return result[0];
};

export const createForm = async (form) => {
  const sql = `
  INSERT INTO forms (subject, text, attachment, author_id)
  VALUES (?, ?, ?, ?)`;

  const values = [form.subject, form.text, form.attachment, form.authorId];

  const result = await query(sql, values);

  return result;
};

export const updateForm = async (form) => {
  const sql = `
  UPDATE forms
  SET text=?,attachment=?,author_id=?
  WHERE form_id=?`;

  const values = [form.text, form.attachment, form.authorId, form.id];

  const result = await query(sql, values);

  return result;
};

export const deleteForm = async (id) => {
  const sql = `DELETE FROM forms WHERE form_id = ?`;

  const result = await query(sql, id);

  return result;
};
```

Friendship-service.js

```
import { query } from '../helpers/database-query.js';

export const getUserFriends = async (id) => {
  const sql = `
  SELECT source_id AS 'user_id' FROM friendships WHERE target_id = ?
  UNION
  SELECT target_id AS 'user_id' FROM friendships WHERE source_id = ? `;
```

```
  const result = await query(sql, [id, id]);

  return result;
};

export const getFriendships = async () => {
  const sql = `SELECT * FROM friendships`;

  const result = await query(sql);

  return result;
};

export const getFriendship = async (id) => {
  const sql = `SELECT * FROM friendships WHERE friendship_id = ?`;

  const result = await query(sql, id);

  return result[0];
};

export const createFriendship = async (friendship) => {
  const sql = `
  INSERT INTO friendships (source_id, target_id)
  VALUES (?, ?)`;

  const values = [friendship.sourceId, friendship.targetId];

  const result = await query(sql, values);

  return result;
};

export const updateFriendship = async (friendship) => {
  const sql = `
  UPDATE friendships
  SET source_id=?,target_id=?
  WHERE friendship_id=?`;

  const values = [friendship.sourceId, friendship.targetId, friendship.id];

  const result = await query(sql, values);

  return result;
};

export const deleteFriendship = async (id) => {
  const sql = `DELETE FROM friendships WHERE friendship_id = ?`;

  const result = await query(sql, id);

  return result;
};
```

Message-service.js

```
import { query } from '../helpers/database-query.js';

export const getMessages = async () => {
  const sql = `SELECT * FROM messages`;

  const result = await query(sql);
```

```
  return result;
};

export const getChatMessages = async (id) => {
  const sql = `SELECT * FROM messages WHERE chat_id = ?`;

  const result = await query(sql, id);

  return result;
};

export const getUserChatMessages = async (userId, chatId) => {
  const sql = `SELECT * FROM messages WHERE author_id = ? AND chat_id = ?`;

  const result = await query(sql, [userId, chatId]);

  return result;
};

export const getMessage = async (id) => {
  const sql = `SELECT * FROM messages WHERE message_id = ?`;

  const result = await query(sql, id);

  return result[0];
};

export const createMessage = async (message) => {
  const sql = `
  INSERT INTO messages (text, attachment, author_id, chat_id)
  VALUES (?, ?, ?, ?)`;

  const values = [
    message.text,
    message.attachment,
    message.authorId,
    message.chatId,
  ];

  const result = await query(sql, values);

  return result;
};

export const updateMessage = async (message) => {
  const sql = `
  UPDATE messages
  SET text=?,attachment=?,author_id=?,chat_id=?
  WHERE message_id=?`;

  const values = [
    message.text,
    message.attachment,
    message.authorId,
    message.chatId,
    message.id,
  ];

  const result = await query(sql, values);

  return result;
```

```
};

export const deleteMessage  = async (id) => {
  const sql = `DELETE FROM messages WHERE message_id = ?`;

  const result = await query(sql, id);

  return result;
};
```

Post-service.js

```
import { query } from '../helpers/database-query.js';

export const getPosts  = async () => {
  const sql = `SELECT * FROM posts ORDER BY creation_time DESC`;

  const result = await query(sql);

  return result;
};

export const getUserPosts  = async (id) => {
  const sql = `SELECT * FROM posts WHERE author_id = ? ORDER BY creation_time DESC`;

  const result = await query(sql, id);

  return result;
};

export const getPost  = async (id) => {
  const sql = `SELECT * FROM posts WHERE post_id = ?`;

  const result = await query(sql, id);

  return result[0];
};

export const createPost = async (post) => {
  const sql = `
  INSERT INTO posts (text, attachment, author_id)
  VALUES (?, ?, ?)`;

  const values = [post.text,  post.attachment,  post.authorId];

  const result = await query(sql, values);

  return result;
};

export const updatePost  = async (post) => {
  const sql = `
  UPDATE posts
  SET text=?,attachment=?,author_id=?
  WHERE post_id=?`;

  const values = [post.text,  post.attachment,  post.authorId, post.id];

  const result = await query(sql, values);
```

```
  return result;
};

export const likePost = async (id) => {
  const sql = `
  UPDATE posts
  SET likes_number = likes_number + 1
  WHERE post_id=?`;

  const result = await query(sql, id);

  return result;
};

export const unlikePost = async (id) => {
  const sql = `
  UPDATE posts
  SET likes_number = likes_number - 1
  WHERE post_id=?`;

  const result = await query(sql, id);

  return result;
};

export const deletePost = async (id) => {
  const sql = `DELETE FROM posts WHERE post_id = ?`;

  const result = await query(sql, id);

  return result;
};
```

User-service.js

```
import { query } from '../helpers/database-query.js';

export const getUsers = async () => {
  const sql = `SELECT user_id, email, nickname, avatar, registration_time, first_name, second_name, biography FROM users`;

  const result = await query(sql);

  return result;
};

export const getUser = async (id) => {
  const sql = `SELECT user_id, email, nickname, avatar, registration_time, first_name, second_name, biography FROM users WHERE user_id = ?`;

  const result = await query(sql, id);

  return result[0];
};

export const createUser = async (user) => {
  const sql = `
  INSERT INTO users (email, password, nickname, avatar, first_name, second_name)
  VALUES (?, ?, ?, ?, ?, ?)`;
```

```
  const values = [
    user.email,
    user.password,
    user.nickname,
    user.avatar,
    user.firstName,
    user.secondName,
  ];

  const result = await query(sql, values);

  return result;
};

export const updateUser  = async (user) => {
  const sql = `
  UPDATE users
  SET email=?,password=?,nickname=?,biography=?,avatar=?,first_name=?,second_name=?
  WHERE user_id=?`;

  const values = [
    user.email,
    user.password,
    user.nickname,
    user.biography,
    user.avatar,
    user.firstName,
    user.secondName,
    user.id,
  ];

  const result = await query(sql, values);

  return result;
};

export const deleteUser  = async (id) => {
  const sql = `DELETE FROM users WHERE user_id = ?`;

  const result = await query(sql, id);

  return result;
};
```

Src/Views

Вебсторінки, які генерує вебсервер.

Chats.pug

```
extends layouts/main-page-layout

block content
  .col.d-flex.flex-column.flex-lg-row.p-0(style={width:  "1px", height: "calc(100vh - 78px)"})
    nav#chats.nav.nav-pills.flex-lg-column.px-lg-3.overflow-auto.flex-nowrap.border-end.border-
bottom
    .d-flex.flex-column.flex-fill.h-100.p-3
      .flex-fill.fs-5
      form(method="post").row.justify-content-center.align-items-center.px-3
        .col-auto.d-none.d-md-inline
          img#userAvatar.rounded-circle(src='/avatars/default_user_avatar.jpeg'  width="40px")
        .col
          input.form-control(type='text',  placeholder='Type  a message')
        .col-auto.d-none.d-md-inline
          label(for="inputFile"  style={cursor: "pointer"})
            i.fs-5.bi-file-earmark-image
          input#inputFile.form-control(type='file'  hidden)
        .col-auto
          button.btn.p-0
            i.fs-5.bi-send
  script(src="/scripts/chats.js" type="module")
```

Contact.pug

```
extends layouts/main-page-layout

block content
  .col.p-3
    .col-lg-8.offset-lg-2.col-xxl-6.offset-xxl-3
      form.d-flex.flex-column.justify-content-center(action="http://localhost:3080/api/forms"
method='post'  novalidate enctype="multipart/form-data"  style={"min-height": "calc(100vh - 110px)"})
        .mb-3.mb-md-4
          label.form-label(for='inputSubject')  Subject
          input#inputSubject.form-control(type='text'  name='subject'  placeholder='Your subject')
        .mb-1
          label.form-label(for='inputMessage')  Message *
           textarea#inputMessage.form-control(name='text'  cols='30'  rows='4'  placeholder='Write  your
message')
        .mb-5
          label(for="inputFile")
          input#inputFile.form-control(type='file'  name="file")
        .row.justify-content-center
          .col-md-5.text-center
            button.btn.btn-block.btn-primary.py-2.px-4  Send Message
  script(src="/scripts/contact.js" type="module")
```

Home.pug

```
extends layouts/main-page-layout

block content
  .col.d-flex.flex-column.p-3
    .row
      .col-lg-8.offset-lg-2.col-xxl-6.offset-xxl-3
        form.row.justify-content-start.align-items-center.mb-3(action="http://localhost:3080/api/posts"
method="post" enctype="multipart/form-data")
          .col-12.mb-2
            textarea.form-control(name="text" rows="3" placeholder='Share your thoughts ...')
          .col-auto
            label(for="inputFile" style={cursor: "pointer"})
              i.fs-5.bi-file-earmark-image
              input#inputFile.form-control(type='file' name="attachment" accept=".jpg,.jpeg,.png,.gif"
hidden)
          .col-auto.ms-auto
            button.btn.btn-primary Share
        #posts.flex-fill.pt-3
          each post in posts
            .card.mb-3.mb-lg-4(data-post-id=`${post.post_id}`)
              .card-header
                .row.align-items-center
                  .col-auto
                    img.rounded-circle.me-3(width='40px' src=`/avatars/${post.author_avatar}` alt='')
                  .col
                    .h5.m-0 #{post.author_first_name} #{post.author_second_name}
                    .text-muted @#{post.author_nickname}
              .card-body.pb-0
                .card-text.mb-3 #{post.text}
                hr.m-0
                .row.align-items-center
                  .col
                    .nav
                      button.post-like.nav-link.d-flex.align-items-center.p-2.p-sm-3
                        i.fs-5.bi-heart.me-1
                        span.fs-5 #{post.likes_number}
                      button.comment-toggle.nav-link.d-flex.align-items-center.p-2.p-sm-3
                        i.fs-5.bi-chat-left-dots
                        span.d-none.d-sm-inline.ms-2 Comment
                  .col-auto
                    span #{new Date(Date.parse(post.creation_time)).getHours()}:#{new
Date(Date.parse(post.creation_time)).getMinutes()}:#{new
Date(Date.parse(post.creation_time)).getSeconds()} #{new Date(Date.parse(post.creation_time)).getDate()}-
#{new Date(Date.parse(post.creation_time)).getMonth() + 1}-#{new
Date(Date.parse(post.creation_time)).getFullYear()}
              .card-footer.d-none.p-3
                each comment in post.comments
                  .row.mb-3
                    .col-auto
                      img.rounded-circle.me-3(width='40px' src=`/avatars/${comment.author_avatar}` alt='')
                    .col.row
                      .col-12
                        span.h5.m-0 #{comment.author_first_name} #{comment.author_second_name}
                        span.text-muted @#{comment.author_nickname}
                      .col-12
                        div #{comment.text}
                  //- .col-auto
                    button.btn.btn-close
                  .w-100
```

```
                .col-auto
                  img.rounded-circle.opacity-0(width='40px'  src=`/avatars/${comment.author_avatar}`
alt='')
                .col
                  .nav
                    button.nav-link.d-flex.align-items-center.p-2.p-sm-3
                      i.fs-5.bi-heart.me-1
                      span.fs-5 #{comment.likes_number}
                .col-auto.align-content-center
                    span  #{new  Date(Date.parse(comment.creation_time)).getHours()}:#{new
Date(Date.parse(comment.creation_time)).getMinutes()}:#{new
Date(Date.parse(comment.creation_time)).getSeconds()}  #{new
Date(Date.parse(comment.creation_time)).getDate()}-#{new
Date(Date.parse(comment.creation_time)).getMonth()  +  1}-#{new
Date(Date.parse(comment.creation_time)).getFullYear()}
            .row
              .col-auto
                img.rounded-circle(src='https://i.imgur.com/F9Nf9Fx.jpeg'  width='40')
              .col
                form.row(action="http://localhost:3080/api/comments"  method="post"
enctype="multipart/form-data")
                  .col-12.mb-1
                    textarea.form-control(name="text")
                  .col-12
                    .mt-2.text-right
                      button.btn.btn-sm.btn-primary.me-3  Post comment
    script(src="/scripts/home.js" type="module")
```

Login.pug

```
    extends  layouts/layout

    block content
      .container.vh-100
        .row.h-100.justify-content-center.align-items-center.text-center
          .col-lg-8.col-xl-6
            .h1.fw-bold.mb-3.mb-lg-5  Log in to your account
            form#loginForm.my-3(method="post"  novalidate)
              .form-floating.mb-3
                input#inputLogin.form-control(type='text'  name='nickname'  minlength="2"
maxlength="30"  required placeholder='')
                label(for='inputLogin')  Nickname or Email
                .invalid-feedback * Please fill out this field!
              .form-floating.mb-3
                input#inputPassword.form-control(type='password'  minlength="6"  maxlength="30"
pattern="(?=.*[A-Z])(?=.*[a-z])(?=.*\\\d).{6,30}" required placeholder='')
                label(for='inputPassword')  Password
                .invalid-feedback * Please fill out this field!
              .my-3.d-block.text-center
                .col-12
                  button.btn.btn-primary.btn-block.w-100.fs-5.py-2  Log In
              .small.text-center
                p Don't have an account?
                  a(href="/signup") Sign Up
      script(src="/scripts/login.js" type="module")
```

Settings.pug

```
extends layouts/main-page-layout

block content
  .col.p-3
    .col-lg-8.offset-lg-2.col-xxl-6.offset-xxl-3
      form#settingsForm.d-flex.flex-column.justify-content-center.mb-3(method="put" style="min-height: calc(100vh - 94px)" enctype="multipart/form-data" novalidate)
        .row.justify-content-center.mb-3.mb-md-4.text-center
          .col-4
            .mb-3.mb-md-4
              img#imageAvatar.w-100.rounded-circle(src='/avatars/default_user_avatar.jpeg')
          .w-100
          .col-4
            label.text-muted(for="inputAvatar" style={cursor: "pointer"}) Choose an avatar
              input#inputAvatar.form-control(type='file' name="avatar" onchange="imageAvatar.src = window.URL.createObjectURL(this.files[0])" hidden)
        .mb-3.mb-md-4
          label.form-label(for='inputFirstName') First name
          input#inputFirstName.form-control(type='text' name="firstName" minlength="2" maxlength="50" pattern="^[A-Z][a-z]{1,39}" required)
          .invalid-feedback * Please fill out this field!
        .mb-3.mb-md-4
          label.form-label(for='inputSecondName') Second name
          input#inputSecondName.form-control(type='text' name="secondName" minlength="2" maxlength="50" pattern="^[A-Z][a-z]{1,39}" required)
          .invalid-feedback * Please fill out this field!
        .mb-3.mb-md-4
          label.form-label(for='inputNickname') Nickname
          input#inputNickname.form-control(type='text' name='nickname' minlength="2" maxlength="30" required)
          .invalid-feedback * Please fill out this field!
        .mb-3.mb-md-4
          label.form-label(for='inputEmail') Email
          input#inputEmail.form-control(type='email' name='email' minlength="5" maxlength="50" pattern=".{5,50}" required)
          .invalid-feedback * Please fill out this field!
        .mt-4.mb-2.d-block.text-center
          .col-12
            button.btn.btn-primary.btn-block.w-100.fs-5.py-2 Save
      h3 Danger zone
      form(method="delete").border.border-3.border-danger.p-3
        .row.align-items-center
          .col-12.col-sm.mb-2.mb-sm-0
            .lead Delete your account
            .muted Once you delete an account, there is no going back. Please be certain.
          .col-auto
            button.btn.btn-danger Delete the account

  script(src="/scripts/settings.js" type="module")
```

Signup.pug

```
extends layouts/layout

block content
  .container.vh-100
    .row.h-100.justify-content-center.align-items-center.text-center
      .col-lg-8.col-xl-6.my-3
        h1.fw-bold.mb-5 Create a new account
        form#signupForm(method="post" novalidate enctype="multipart/form-data")
          .row.justify-content-center.mb-3.mb-md-4
            .col-4
              .mb-3.mb-md-4
                img#imageAvatar.w-100.rounded-circle(src='/avatars/default_user_avatar.jpeg')
            .w-100
            .col-4
              input#inputAvatar.form-control(type='file' name="avatar" onchange="imageAvatar.src = window.URL.createObjectURL(this.files[0])" accept=".jpg,.jpeg,.png,.gif" hidden)
              label.text-muted(for="inputAvatar" style={cursor: "pointer"}) Choose an avatar
          .row.mb-3.mb-md-4
            .col-md-6
              .form-floating.mb-3.mb-md-0
                input#inputFirstName.form-control(type='text' name="firstName" minlength="2" maxlength="50" pattern="^[A-Z][a-z]{1,39}" placeholder='Enter your first name' required)
                label(for='inputFirstName') First name
                .invalid-feedback * Please fill out this field!
            .col-md-6
              .form-floating
                input#inputSecondName.form-control(type='text' name="secondName" minlength="2" maxlength="50" pattern="^[A-Z][a-z]{1,39}" placeholder='Enter your second name' required)
                label(for='inputSecondName') Second name
                .invalid-feedback * Please fill out this field!
          .form-floating.mb-3.mb-md-4
            input#inputNickname.form-control(type='text' name='nickname' placeholder='nickname' minlength="2" maxlength="30" required)
            label(for='inputNickname') Nickname
            .invalid-feedback * Please fill out this field!
          .form-floating.mb-3.mb-md-4
            input#inputEmail.form-control(type='email' name='email' placeholder='name@example.com' minlength="5" maxlength="50" pattern=".{5,50}" required)
            label(for='inputEmail') Email address
            .invalid-feedback * Please fill out this field!
          .row
            .col-md-6
              .form-floating.mb-3.mb-md-0
                input#inputPassword.form-control(type='password' name='password' placeholder='Create a password' minlength="6" maxlength="30" pattern="(?=.*[A-Z])(?=.*[a-z])(?=.*\\\d).{6,30}" required)
                label(for='inputPassword') Password
                .invalid-feedback * Please fill out this field!
            .col-md-6
              .form-floating.mb-3.mb-md-0
                input#inputPasswordConfirm.form-control(type='password' name='passwordConfirm' placeholder='Confirm password' minlength="6" maxlength="30" pattern="(?=.*[A-Z])(?=.*[a-z])(?=.*\\\d).{6,30}" required)
                label(for='inputPasswordConfirm') Confirm Password
                .invalid-feedback * Please fill out this field!
            .form-text Your password must be 6-30 characters long, contain at least one capital letter, one small letter and one number
          .mt-4.mb-2.d-block.text-center
```

```
                .col-12
                    button.btn.btn-primary.btn-block.w-100.fs-5.py-2(type='submit')  Create an account
                .small.text-center
                    p Have  an account?
                        a(href="/login") Login
        script(src="/scripts/signup.js" type="module")
```

User.pug

```
    extends  layouts/main-page-layout

    block content
      .col.d-flex.flex-column.p-3
        .row
          .col-lg-8.offset-lg-2.col-xxl-6.offset-xxl-3
            .row.align-items-center.mb-3
              .col.col-sm-4.col-xl-3.mb-2
                img.img-fluid.rounded-circle(src=`/avatars/${avatar}`)
              .col-auto.flex-column.align-content-center
                h5.fw-bold #{first_name} #{second_name}
                .muted @#{nickname}
              .col.text-center.text-sm-end
                button.btn.btn-primary  Add as friend
            .w-100
            .col.mt-2.px-2
              p #{biography}
            #userNavigation.nav.nav-pills.nav-fill.flex-nowrap.overflow-auto.text-center.gap-3.gap-md-5.mb-
3
              a.nav-link.rounded-5.active(data-table-name="Posts")  Posts
              a.nav-link.rounded-5(data-table-name="Friends")  Friends
              a.nav-link.rounded-5(data-table-name="Comments")  Comments
            #userContent.flex-fill.py-3
              #userPosts
                each post in posts
                  .card.mb-3.mb-lg-4
                    .card-header
                      .row.align-items-center
                        .col-auto
                          img.rounded-circle.me-3(width='40px'  src=`/avatars/${avatar}`  alt='')
                        .col
                          .h5.m-0 #{first_name} #{second_name}
                          .text-muted  @#{nickname}
                    .card-body.pb-0
                      .card-text.mb-3=  post.text
                      hr.m-0
                      .row.align-items-center
                        .col
                          .nav
                            a.post-like.nav-link.d-flex.align-items-center.p-2.p-sm-3
                              i.fs-5.bi-heart.me-1
                              span.fs-5= post.likes_number
                        .col-auto
                          span  #{new  Date(Date.parse(post.creation_time)).getHours()}:#{new
Date(Date.parse(post.creation_time)).getMinutes()}:#{new
Date(Date.parse(post.creation_time)).getSeconds()}  #{new  Date(Date.parse(post.creation_time)).getDate()}-
#{new  Date(Date.parse(post.creation_time)).getMonth()  +  1}-#{new
Date(Date.parse(post.creation_time)).getFullYear()}
              #userFriends.d-none
                each friend in friends
```

```
                a.row.align-items-center.text-decoration-none.mb-
3(href=`http://localhost:3080/user/${friend.user_id}`)
                  .col-auto
                    img.rounded-circle.me-3(width='40px' src=`/avatars/${friend.avatar}` alt='')
                  .col
                    .h5.m-0 #{friend.first_name} #{friend.second_name}
                    .text-muted @#{friend.nickname}
              #userComments.d-none
                each comment in comments
                  .row.mb-3
                    .col-auto
                      img.rounded-circle.me-3(width='40px' src=`/avatars/${avatar}` alt='')
                    .col.row
                      .col-12
                        span.h5.m-0 #{first_name} #{second_name}
                        span.text-muted @#{nickname}
                      .col-12
                        div #{comment.text}
                    .w-100
                    .col-auto
                      img.rounded-circle.opacity-0(width='40px' src=`/avatars/${avatar}` alt='')
                    .col
                      .nav
                        a.nav-link.d-flex.align-items-center.p-2.p-sm-3
                          i.fs-5.bi-heart-fill.me-1
                          span.fs-5 #{comment.likes_number}
                    .col-auto.align-content-center
                      span #{new Date(Date.parse(comment.creation_time)).getHours()}:#{new
Date(Date.parse(comment.creation_time)).getMinutes()}:#{new
Date(Date.parse(comment.creation_time)).getSeconds()} #{new
Date(Date.parse(comment.creation_time)).getDate()}-#{new
Date(Date.parse(comment.creation_time)).getMonth() + 1}-#{new
Date(Date.parse(comment.creation_time)).getFullYear()}

        script(src='/scripts/user.js' type='module')
```

Src/Views/Errors

Сторінки для генерації інформації про помилки робоби вебдодатку.

401.pug

```
extends ../layouts/error

block content
    .container
        .row.align-items-center.min-vh-100
            .text-center
                h1.display-1 404
                p.lead The page you requested was not found!
                a(href='/home') Return to Home
```

404.pug

```
extends ../layouts/error

block content
    .container
        .row.align-items-center.min-vh-100
            .text-center
                h1.display-1 404
                p.lead The page you requested was not found!
                a(href='/home') Return to Home
```

500.pug

```
extends ../layouts/error

block content
    .container
        .row.align-items-center.min-vh-100
            .text-center
                h1.display-1 500
                p.lead Internal Server Error
                a(href='/home') Return to Home
```

Src/Views/Layouts/

Шаблони для генерації сторінок з вебсерверу.

Error.pug

```
doctype html
html(lang="en")
  head
    meta(charset="UTF-8")
    meta(name="viewport", content="width=device-width, initial-scale=1.0")
    title ConnectHub
    link(rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css")
    link(rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-
icons.min.css")
  body
    block content
```

Main-page-layout.pug

```
doctype html
html(lang="en")
  head
    meta(charset="UTF-8")
    meta(name="viewport", content="width=device-width, initial-scale=1.0")
    title ConnectHub
    link(rel="stylesheet" type="text/css" href="/styles/general.css")
    link(rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css")
    link(rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-
icons.min.css")
    script(src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" defer)
    script(src="/scripts/general.js" type="module" defer)
  body
    nav.navbar.navbar-expand.sticky-top.bg-body-tertiary
      .container-fluid
        a.navbar-brand.p-3(href='/home')
          span.fs-5.d-none.d-sm-inline ConnectHub
          span.fs-5.d-sm-none CH
        navbar-nav.flex-row.ms-auto
          a.nav-link(href='/login')
            button.btn.btn-primary.btn-block.py-2 Log In
          a.nav-link(href='/signup')
            button.btn.btn-primary.btn-block.bg-transparent.text-body.py-2 Sign Up
    .container-fluid
      .row(style={"min-height": "calc(100vh - 78px)"})
        .col-auto.col-md-3.col-xl-2.bg-body-tertiary.p-3
          nav.nav.nav-pills.flex-column.gap-1.sticky-top(style={top: "94px"})
            a#homeNav.nav-link.d-flex.align-items-center.text-black(href='/home')
              i.fs-4.bi-house.me-md-1
              span.ms-1.d-none.d-md-inline Home
            a#chatsNav.nav-link.d-flex.align-items-center.text-black(href='/chats')
              i.fs-4.bi-chat.me-md-1
              span.ms-1.d-none.d-md-inline Chats
            a#settingsNav.nav-link.d-flex.align-items-center.text-black(href='/settings')
```

```
                i.fs-4.bi-gear.me-md-1
                span.ms-1.d-none.d-md-inline Settings
              a#profileNav.nav-link.d-flex.align-items-center.text-black(href='/user/3')
                i.fs-4.bi-person.me-md-1
                span.ms-1.d-none.d-md-inline Profile
              a#contactNav.nav-link.d-flex.align-items-center.text-black(href='/contact')
                i.fs-4.bi-question-circle.me-md-1
                span.ms-1.d-none.d-md-inline Contact Us
              a.nav-link.d-flex.align-items-center.text-black(href="" data-bs-toggle='modal' data-bs-target='#adminPanel')
                i.fs-4.bi-pencil-square.me-md-1
                span.ms-1.d-none.d-md-inline Admin panel
          block content
        #adminPanel.modal.fade(tabindex='-1')
          .modal-dialog.modal-fullscreen.modal-dialog-scrollable
            .modal-content
              .modal-header
                h1#adminPanelLabel.modal-title.fs-5 Admin panel
                button.btn-close(type='button' data-bs-dismiss='modal')
              .modal-body
                .row.h-100
                  .col-auto.col-lg-2
                    nav#adminNav.nav.nav-pills.flex-column.gap-1.sticky-top
                      a.nav-link.active.d-flex.align-items-center.text-black.text-white(data-table-name = "users")
                        i.fs-4.bi-person.me-lg-1
                        span.ms-1.d-none.d-lg-inline Users
                      a.nav-link.d-flex.align-items-center.text-black(data-table-name = "posts")
                        i.fs-4.bi-postcard-heart.me-lg-1
                        span.ms-1.d-none.d-lg-inline Posts
                      a.nav-link.d-flex.align-items-center.text-black(data-table-name = "forms")
                        i.fs-4.bi-info-circle.me-lg-1
                        span.ms-1.d-none.d-lg-inline Contact forms
                  .col.pe-0
                    #adminTable
                      table#usersTable.table.table-hover.align-middle.caption-top.m-0
                        caption Number of users:
                          span#userNumber 44
                        thead.sticky-top.text-center
                          tr
                            th(scope='col') Avatar
                            th(scope='col') First name
                            th(scope='col') Second name
                            th(scope='col') Nickname
                            th(scope='col') Biography
                            th(scope='col') Email
                            th(scope='col') Controls
                        tbody.text-center
                          tr
                            th(scope='row') 1
                            td
                              img.rounded-circle(src="/avatars/default_user_avatar.jpeg" style={width: "64px"})
                            td Mark
                            td Otto
                            td @mdo
                              td Lorem ipsum dolor sit amet consectetur adipisicing elit. Laborum perspiciatis modi, atque similique porro numquam totam quaerat, iusto ex inventore unde itaque soluta, nemo illo quis fuga. Voluptates, delectus ipsa.
                            td markto@mail.com
                            td
                              nav.nav.flex-nowrap.justify-content-center
                                a.nav-link(href="/user/3")
```

```
                    i.fs-5.bi-eye
                  a.nav-link(href="" data-bs-toggle='modal'  data-bs-target='#userPanel')
                    i.fs-5.bi-pencil
                  a.nav-link(href="")
                    i.fs-5.bi-trash
            table#postsTable.table.table-hover.align-middle.caption-top.m-0.d-none
              caption Number of posts:
                span#postNumber  13
              thead.sticky-top.text-center
                tr
                  th(scope='col') Author ID
                  th(scope='col') Text
                  th(scope='col') Attachment
                  th(scope='col') Likes
                  th(scope='col') Controls
              tbody.text-center
                tr
                  th(scope='row') 1
                  td 3
                    td Lorem ipsum dolor sit, amet consectetur adipisicing elit. Quas in expedita omnis nobis non beatae, sint veniam laudantium quia nostrum quasi tempora fugit officia repellendus voluptates, rerum, corrupti esse dignissimos!
                  td
                    img(src="/avatars/default_user_avatar.jpeg"  style={width: "64px"})
                  td 12
                  td 2
                  td
                    nav.nav.flex-nowrap.justify-content-center
                      //- a.nav-link(href="" )
                        i.fs-5.bi-eye
                      a.nav-link(href="" data-bs-toggle='modal'  data-bs-target='#postPanel')
                        i.fs-5.bi-pencil
                      a.nav-link(href="")
                        i.fs-5.bi-trash
            table#formsTable.table.table-hover.align-middle.caption-top.m-0.d-none
              caption Number of forms:
                span#formNumber  2
              thead.sticky-top.text-center
                tr
                  th(scope='col') Author ID
                  th(scope='col') Subject
                  th(scope='col') Text
                  th(scope='col') Attachment
                  th(scope='col') Time
                  th(scope='col') Controls
              tbody.text-center
                tr
                  th(scope='row') 1
                  td 3
                  td New feature
                  td Please add some cool features in future
                  td
                    img(src="/avatars/default_user_avatar.jpeg"  style={width: "64px"})
                  td 2024-05-24 14:50:34
                  td
                    nav.nav.flex-nowrap.justify-content-center
                      a.nav-link(href="" data-bs-toggle='modal'  data-bs-target='#formPanel')
                        i.fs-5.bi-eye
                      a.nav-link(href="")
                        i.fs-5.bi-trash
        #userPanel.modal.fade(tabindex='-1')
          .modal-dialog.modal-dialog-centered.modal-dialog-scrollable
```

```
                .modal-content
                  .modal-header
                    h1#adminPanelLabel.modal-title.fs-5  User editor panel
                     button.btn.btn-primary.ms-auto(data-bs-target='#adminPanel'    data-bs-toggle='modal')    Back
to Admin panel
                  .modal-body
                    form#userPanelForm.d-flex.flex-column.justify-content-center.mb-
3(action="http://localhost:3080/api/users/"  method="put"  enctype="multipart/form-data")
                     .row.justify-content-center.mb-3.mb-md-4.text-center
                      .col-4
                        .mb-3.mb-md-4
                          img#imageAvatar.w-100.rounded-circle(src='/avatars/default_user_avatar.jpeg')
                      .w-100
                      .col-4
                        label.text-muted(for="userInputAvatar"  style={cursor: "pointer"}) Choose an avatar
                          input#userInputAvatar.form-control(type='file'           onchange="imageAvatar.src           =
window.URL.createObjectURL(this.files[0])"  hidden)
                     .mb-3.mb-md-4
                      label.form-label(for='userInputFirstName')  First name
                      input#userInputFirstName.form-control(type='text')
                     .mb-3.mb-md-4
                      label.form-label(for='userInputSecondName')  Second name
                      input#userInputSecondName.form-control(type='text')
                     .mb-3.mb-md-4
                      label.form-label(for='userInputNickname')  Nickname
                      input#userInputNickname.form-control(type='text')
                     .mb-3.mb-md-4
                      label.form-label(for='userInputBiography')  Biography
                      textarea#userInputBiography.form-control(type='text')
                     .mb-3.mb-md-4
                      label.form-label(for='userInputEmail')  Email
                      input#userInputEmail.form-control(type='email')
                     .mt-4.mb-2.d-block.text-center
                      .col-12
                        button.btn.btn-primary.btn-block.w-100.fs-5.py-2  Save
            #postPanel.modal.fade(tabindex='-1')
              .modal-dialog.modal-dialog-centered.modal-dialog-scrollable
                .modal-content
                  .modal-header
                    h1#adminPanelLabel.modal-title.fs-5  Post editor panel
                     button.btn.btn-primary.ms-auto(data-bs-target='#adminPanel'    data-bs-toggle='modal')    Back
to Admin panel
                  .modal-body
                    form.d-flex.flex-column.justify-content-center.mb-3(method="put"       enctype="multipart/form-
data")
                     .row.justify-content-center.mb-3.mb-md-4.text-center
                      .col-4
                        label.text-muted(for="postInputAttachment"  style={cursor: "pointer"}) Attachment
                          input#postInputAttachment.form-control(type='file'       onchange="imageAvatar.src       =
window.URL.createObjectURL(this.files[0])"  hidden)
                      .w-100
                      .col-4
                        .mb-3.mb-md-4
                          img#postImageAttachment.w-100(src='/avatars/default_user_avatar.jpeg')
                     .mb-3.mb-md-4
                      label.form-label(for='postInputMessage')  Text
                      textarea#postInputMessage.form-control(cols='30'  rows='4')
                     .mb-3.mb-md-4
                      label.form-label(for='postInputNickname')  Likes
                      input#postInputNickname.form-control(type='text')
                     .mt-4.mb-2.d-block.text-center
                      .col-12
```

```
                button.btn.btn-primary.btn-block.w-100.fs-5.py-2 Save
            #formPanel.modal.fade(tabindex='-1')
              .modal-dialog.modal-dialog-centered.modal-dialog-scrollable
                .modal-content
                  .modal-header
                    h1#adminPanelLabel.modal-title.fs-5 Form reply panel
                    button.btn.btn-primary.ms-auto(data-bs-target='#adminPanel' data-bs-toggle='modal') Back to Admin panel
                  .modal-body
                    form.d-flex.flex-column.justify-content-center.mb-3(method="put" enctype="multipart/form-data")
                      .row.justify-content-center.mb-3.mb-md-4.text-center
                        .col-4
                          label.text-muted(for="formAttachment") Attachment
                        .w-100
                        .col-4
                          .mb-3.mb-md-4
                            img#formAttachment.w-100(src='/avatars/default_user_avatar.jpeg')
                      .mb-3.mb-md-4
                        label.form-label(for='formSubject') Subject
                        input#formSubject.form-control(type="text" disabled)
                      .mb-3.mb-md-4
                        label.form-label(for='formMessage') Text
                        textarea#formMessage.form-control(cols='30' rows='4' disabled)
                      .mb-3.mb-md-4
                        label.form-label(for='formInputReply') Reply text
                        textarea#formInputReply.form-control(cols='30' rows='4')
                      .mt-4.mb-2.d-block.text-center
                        .col-12
                          button.btn.btn-primary.btn-block.w-100.fs-5.py-2 Save
```

Src/Static/

Статичні файли для обслуговування серверу. Тут зберігаються скрипти вебсторінок, їх стилі, а також фотографії профілів користувачів.

Src/Static/Avatars/

Світлини профілів користувачів.

Src/Static/Scripts/

Скрипти для статичних сторінок.

Chats. js

```
document.addEventListener('DOMContentLoaded', () => {
  chatsNav.classList.add('active', 'text-white');
});
```

Contact.js

```
document.addEventListener('DOMContentLoaded', () => {
  contactNav.classList.add('active', 'text-white');
});
```

General.js

```
import { makeRequest } from './helpers.js';

function createUserElement(user) {
  return `<tr data-user-id=${user.user_id}>
    <td><img class="rounded-circle" src="/avatars/${user.avatar}" style="width:64px;"></td>
    <td>${user.first_name}</td>
    <td>${user.second_name}</td>
    <td>@${user.nickname}</td>
    <td>${user.biography}</td>
    <td>${user.email}</td>
    <td>
      <nav class="nav flex-nowrap justify-content-center controls">
      <a class="nav-link" href="/user/${user.user_id}">
        <i class="fs-5 bi-eye"></i>
      </a>
       <a class="nav-link" href="" data-bs-toggle="modal" data-bs-target="#userPanel" data-
action="edit">
        <i class="fs-5 bi-pencil"></i>
      </a>
      <a class="nav-link" href="" data-action="view">
        <i class="fs-5 bi-trash"></i>
      </a>
```

```
            </nav>
          </td>
        </tr>`;
      }

      function createPostElement(post) {
        return `<tr data-post-id=${post.post_id}>
          <td>${post.author_id}</td>
          <td>${post.text}</td>
          <td><img src="/files/${post.attachment}" style="width:64px;"</td>
          <td>${post.likes_number}</td>
          <td>
            <nav class="nav flex-nowrap justify-content-center controls">
              <a class="nav-link" href="" data-bs-toggle="modal" data-bs-target="#postPanel" data-
action="edit">
                <i class="fs-5 bi-pencil"></i>
              </a>
              <a class="nav-link" href="" data-action="delete">
                <i class="fs-5 bi-trash"></i>
              </a>
            </nav>
          </td>
        </tr>`;
      }

      function createFormElement(form) {
        return `<tr data-form-id=${form.form_id}>
          <td>${form.author_id}</td>
          <td>${form.subject}</td>
          <td>${form.text}</td>
          <td><img src="/files/${form.attachment}" style="width:64px;" /></td>
          <td>${form.creation_time}</td>
          <td>
            <nav class="nav flex-nowrap justify-content-center controls">
              <a class="nav-link" href="" data-bs-toggle="modal" data-bs-target="#formPanel" data-
action="edit">
                <i class="fs-5 bi-eye"></i>
              </a>
              <a class="nav-link" href="" data-action="delete">
                <i class="fs-5 bi-trash"></i>
              </a>
            </nav>
          </td>
        </tr>`;
      }

      function createEntityElements(type, entities) {
        return entities.map((entity) => {
          switch (type) {
            case 'users':
              return createUserElement(entity);
            case 'posts':
              return createPostElement(entity);
            case 'forms':
              return createFormElement(entity);
          }
        });
      }

      document.addEventListener('DOMContentLoaded', async () => {
        const [users, posts, forms] = await Promise.all([
          makeRequest('http://localhost:3080/api/users'),
```

```
    makeRequest('http://localhost:3080/api/posts'),
    makeRequest('http://localhost:3080/api/forms'),
  ]);

  const userElements = createEntityElements('users', users);
  const postElements = createEntityElements('posts', posts);
  const formElements = createEntityElements('forms', forms);

  document.querySelector('#usersTable tbody').innerHTML = userElements.join('');
  document.querySelector('#userNumber').innerHTML = userElements.length;

  document.querySelector('#postsTable tbody').innerHTML = postElements.join('');
  document.querySelector('#postNumber').innerHTML = postElements.length;

  document.querySelector('#formsTable tbody').innerHTML = formElements.join('');
  document.querySelector('#formNumber').innerHTML = formElements.length;
});

function removeActiveLinks() {
  const navLinks = adminNav.querySelectorAll('a');
  navLinks.forEach((navLink) => {
    navLink.classList.remove('active', 'text-white');
  });
}

function hideAdminTables() {
  const tables = adminTable.querySelectorAll('table');
  tables.forEach((table) => {
    table.classList.add('d-none');
  });
}

function showProperAdminTable(tableName) {
  document.querySelector(`#${tableName}Table`).classList.remove('d-none');
}

adminNav.addEventListener('click', (event) => {
  const navLink = event.target.closest('a');
  if (!navLink) return;

  removeActiveLinks();
  navLink.classList.add('active', 'text-white');

  hideAdminTables();
  showProperAdminTable(navLink.dataset.tableName);
});

usersTable.addEventListener('click', async (event) => {
  const target = event.target;

  if (!target.closest('.controls')) return;

  if (target.closest('[data-action="edit"]')) {
    const userId = target.closest('tr').dataset.userId;
    const endpoint = `http://localhost:3080/api/users/${userId}`;

    const response = await fetch(endpoint);
    const user = await response.json();

    userPanelForm.action = endpoint;
    imageAvatar.src = `/avatars/${user.avatar}`;
    userInputFirstName.value = user.first_name;
```

```
    userInputSecondName.value  = user.second_name;
    userInputNickname.value  = user.nickname;
    userInputBiography.value  = user.biography;
    userInputEmail.value  = user.email;
  }

  if (target.closest('[data-action="delete"]')) {
  }
});
```

Helpers.js

```
export async function makeRequest(url) {
  const response = await fetch(url);
  const result = await response.json();

  return result;
}
```

Home.pug

```
document.addEventListener('DOMContentLoaded', async () => {
  homeNav.classList.add('active', 'text-white');
});

function toggleLikeIcon(likeIcon) {
  if (likeIcon.classList.contains('bi-heart')) {
    likeIcon.classList.add('bi-heart-fill');
    likeIcon.classList.remove('bi-heart');
  } else {
    likeIcon.classList.add('bi-heart');
    likeIcon.classList.remove('bi-heart-fill');
  }
}

function increaseLikesNumber(likeSpan) {}

posts.addEventListener('click', (event) => {
  const target = event.target;
  const post = target.closest('.card');
  const postId = post.dataset.postId;

  if (target.closest('.post-like')) {
    likePost(target.closest('.post-like'), postId);
  }

  if (target.closest('.comment-toggle')) {
    const commentSection = post.querySelector('.card-footer');

    if (commentSection.classList.contains('d-none')) {
      commentSection.classList.remove('d-none');
    } else {
      commentSection.classList.add('d-none');
    }
  }
});
```

Login.js

```
function checkFormValidity(event) {
  if (!loginForm.checkValidity()) {
    event.preventDefault();
    event.stopPropagation();
  }

  loginForm.classList.add('was-validated');
}

function showLoginErrorMessage() {
  const validityState = inputLogin.validity;
  const parentElement = inputLogin.parentElement;
  const invalidFeedbackElement = inputLogin.parentElement.lastElementChild;

  parentElement.classList.add('was-validated');
  invalidFeedbackElement.textContent = '';

  if (validityState.valueMissing) {
    invalidFeedbackElement.textContent = '* Please fill out this field!';
  }

  if (validityState.tooShort) {
    invalidFeedbackElement.textContent = '* Login is too short!';
  }
}

function showPasswordErrorMessage() {
  const validityState = inputPassword.validity;
  const parentElement = inputPassword.parentElement;
  const invalidFeedbackElement = inputPassword.parentElement.lastElementChild;

  const passValue = inputPassword.value;

  const capitalLetterRegExp = /(?=.*[A-Z])./;
  const letterRegExp = /^(?=.*[a-z])/;
  const digitRegExp = /^(?=.*\d)/;

  const capitalError = !capitalLetterRegExp.test(passValue);
  const letterError = !letterRegExp.test(passValue);
  const digitError = !digitRegExp.test(passValue);
  const lengthError = validityState.tooShort;

  parentElement.classList.add('was-validated');
  invalidFeedbackElement.textContent = '';

  if (validityState.valueMissing) {
    invalidFeedbackElement.textContent = '* Please fill out this field!';
  }

  if (validityState.patternMismatch) {
    if (capitalError) {
      invalidFeedbackElement.textContent =
        '* Should contain at least one capital letter!';
    }
    if (letterError) {
      invalidFeedbackElement.textContent =
        '* Should contain at least one small letter!';
    }
    if (digitError) {
```

```
    invalidFeedbackElement.textContent =
      '* Should contain at least one digit!';
   }
  }

  if (lengthError) {
   invalidFeedbackElement.textContent = '* Password is too short!';
  }
}

loginForm.addEventListener('submit', checkFormValidity);
inputLogin.addEventListener('change', showLoginErrorMessage);
inputPassword.addEventListener('change', showPasswordErrorMessage);
```

Settings. js

```
document.addEventListener('DOMContentLoaded', () => {
  settingsNav.classList.add('active', 'text-white');
});

function checkFormValidity(event) {
  if (!settingsForm.checkValidity()) {
   event.preventDefault();
   event.stopPropagation();
  }

  settingsForm.classList.add('was-validated');
}

function showNameErrorMessage(event) {
  const nameInput = event.target;
  const validityState = nameInput.validity;
  const parentElement = nameInput.parentElement;
  const invalidFeedbackElement = nameInput.parentElement.lastElementChild;

  parentElement.classList.add('was-validated');
  invalidFeedbackElement.textContent = '';

  if (validityState.valueMissing) {
   invalidFeedbackElement.textContent = '* Please fill out this field!';
  }

  if (validityState.patternMismatch) {
   invalidFeedbackElement.textContent =
    '* Should start with a capital letter!';
  }

  if (validityState.tooShort) {
   invalidFeedbackElement.textContent = '* Name is too short!';
  }
}

function showNicknameErrorMessage() {
  const validityState = inputNickname.validity;
  const parentElement = inputNickname.parentElement;
  const invalidFeedbackElement = inputNickname.parentElement.lastElementChild;

  parentElement.classList.add('was-validated');
  invalidFeedbackElement.textContent = '';
```

```
  if (validityState.valueMissing) {
    invalidFeedbackElement.textContent = '* Please fill out this field!';
  }

  if (validityState.tooShort) {
    invalidFeedbackElement.textContent = '* Nickname is too short!';
  }
}

function showEmailErrorMessage() {
  const validityState = inputEmail.validity;
  const parentElement = inputEmail.parentElement;
  const invalidFeedbackElement = inputEmail.parentElement.lastElementChild;

  parentElement.classList.add('was-validated');
  invalidFeedbackElement.textContent = '';

  if (validityState.valueMissing) {
    invalidFeedbackElement.textContent = '* Please fill out this field!';
  }

  if (validityState.typeMismatch) {
    invalidFeedbackElement.textContent = '* Should be an email!';
  }

  if (validityState.tooShort) {
    invalidFeedbackElement.textContent = '* Email is too short!';
  }
}

settingsForm.addEventListener('submit', checkFormValidity);
inputFirstName.addEventListener('change', showNameErrorMessage);
inputSecondName.addEventListener('change', showNameErrorMessage);
inputNickname.addEventListener('change', showNicknameErrorMessage);
inputEmail.addEventListener('change', showEmailErrorMessage);
```

Signup. js

```
function checkFormValidity(event) {
  if (!signupForm.checkValidity()) {
    event.preventDefault();
    event.stopPropagation();
  }

  signupForm.classList.add('was-validated');
}

function showNameErrorMessage(event) {
  const nameInput = event.target;
  const validityState = nameInput.validity;
  const parentElement = nameInput.parentElement;
  const invalidFeedbackElement = nameInput.parentElement.lastElementChild;

  parentElement.classList.add('was-validated');
  invalidFeedbackElement.textContent = '';

  if (validityState.valueMissing) {
    invalidFeedbackElement.textContent = '* Please fill out this field!';
```

```
  }

  if (validityState.patternMismatch) {
    invalidFeedbackElement.textContent =
      '* Should start with a capital letter!';
  }

  if (validityState.tooShort) {
    invalidFeedbackElement.textContent = '* Name is too short!';
  }
}

function showNicknameErrorMessage() {
  const validityState = inputNickname.validity;
  const parentElement = inputNickname.parentElement;
  const invalidFeedbackElement = inputNickname.parentElement.lastElementChild;

  parentElement.classList.add('was-validated');
  invalidFeedbackElement.textContent = '';

  if (validityState.valueMissing) {
    invalidFeedbackElement.textContent = '* Please fill out this field!';
  }

  if (validityState.tooShort) {
    invalidFeedbackElement.textContent = '* Nickname is too short!';
  }
}

function showEmailErrorMessage() {
  const validityState = inputEmail.validity;
  const parentElement = inputEmail.parentElement;
  const invalidFeedbackElement = inputEmail.parentElement.lastElementChild;

  parentElement.classList.add('was-validated');
  invalidFeedbackElement.textContent = '';

  if (validityState.valueMissing) {
    invalidFeedbackElement.textContent = '* Please fill out this field!';
  }

  if (validityState.typeMismatch) {
    invalidFeedbackElement.textContent = '* Should be an email!';
  }

  if (validityState.tooShort) {
    invalidFeedbackElement.textContent = '* Email is too short!';
  }
}

function showPasswordErrorMessage() {
  const validityState = inputPassword.validity;
  const parentElement = inputPassword.parentElement;
  const invalidFeedbackElement = inputPassword.parentElement.lastElementChild;

  const passValue = inputPassword.value;

  const capitalLetterRegExp = /(?=.*[A-Z])./;
  const letterRegExp = /^(?=.*[a-z])/;
  const digitRegExp = /^(?=.*\d)/;

  const capitalError = !capitalLetterRegExp.test(passValue);
```

```
    const letterError = !letterRegExp.test(passValue);
    const digitError = !digitRegExp.test(passValue);
    const lengthError = validityState.tooShort;

    parentElement.classList.add('was-validated');
    invalidFeedbackElement.textContent = '';

    if (validityState.valueMissing) {
      invalidFeedbackElement.textContent = '* Please fill out this field!';
    }

    if (validityState.patternMismatch) {
      if (capitalError) {
        invalidFeedbackElement.textContent =
          '* Should contain at least one capital letter!';
      }
      if (letterError) {
        invalidFeedbackElement.textContent =
          '* Should contain at least one small letter!';
      }
      if (digitError) {
        invalidFeedbackElement.textContent =
          '* Should contain at least one digit!';
      }
    }

    if (lengthError) {
      invalidFeedbackElement.textContent = '* Password is too short!';
    }
  }

  function showPasswordConfirmErrorMessage() {
    const validityState = inputPasswordConfirm.validity;
    const parentElement = inputPasswordConfirm.parentElement;
    const invalidFeedbackElement =
      inputPasswordConfirm.parentElement.lastElementChild;

    parentElement.classList.add('was-validated');
    invalidFeedbackElement.textContent = '';

    if (validityState.valueMissing) {
      invalidFeedbackElement.textContent = '* Please fill out this field!';
    }

    if (validityState.tooShort) {
      invalidFeedbackElement.textContent = '* Password is too short!';
    }

    if (inputPasswordConfirm.value !== inputPassword.value) {
      inputPasswordConfirm.style.cssText = `
      border-color: var(--bs-form-invalid-border-color);
      padding-right: calc(1.5em + .75rem);
      background-image: url("data:image/svg+xml,%3csvg xmlns='http://www.w3.org/2000/svg'
viewBox='0 0 12 12' width='12' height='12' fill='none' stroke='%23dc3545'%3e%3ccircle cx='6' cy='6'
r='4.5'/%3e%3cpath stroke-linejoin='round' d='M5.8 3.6h.4L6 6.5z'/%3e%3ccircle cx='6' cy='8.2' r='.6'
fill='%23dc3545' stroke='none'/%3e%3c/svg%3e");
      background-repeat: no-repeat;
      background-position: right calc(.375em + .1875rem) center;
      background-size: calc(.75em + .375rem) calc(.75em + .375rem);
      `;

      invalidFeedbackElement.textContent = '* Passwords should match!';
```

```
    invalidFeedbackElement.style.display = 'block';
  }
}

signupForm.addEventListener('submit', checkFormValidity);
inputFirstName.addEventListener('change', showNameErrorMessage);
inputSecondName.addEventListener('change', showNameErrorMessage);
inputNickname.addEventListener('change', showNicknameErrorMessage);
inputEmail.addEventListener('change', showEmailErrorMessage);
inputPassword.addEventListener('change', showPasswordErrorMessage);
inputPasswordConfirm.addEventListener(
  'change',
  showPasswordConfirmErrorMessage
);
```

User. js

```
document.addEventListener('DOMContentLoaded', () => {
  profileNav.classList.add('active', 'text-white');
});

function removeActiveLinks() {
  const navLinks = userNavigation.querySelectorAll('a');
  navLinks.forEach((navLink) => {
    navLink.classList.remove('active');
  });
}

function hideAdminTables() {
  const userPosts = userContent.querySelector('#userPosts');
  const userFriends = userContent.querySelector('#userFriends');
  const userComments = userContent.querySelector('#userComments');

  userPosts.classList.add('d-none');
  userFriends.classList.add('d-none');
  userComments.classList.add('d-none');
}

function showProperAdminTable(tableName) {
  document.querySelector(`#user${tableName}`).classList.remove('d-none');
}

userNavigation.addEventListener('click', (event) => {
  const navLink = event.target.closest('a');
  if (!navLink) return;

  removeActiveLinks();
  navLink.classList.add('active');
  hideAdminTables();
  showProperAdminTable(navLink.dataset.tableName);
});
```

Src/Static/Styles

Стилі для статичних вебсторінок

General.css

```
@media only screen and (max-width: 425px) {
  #userNavigation {
    width: calc(100vw - 118px);
  }
}

@media only screen and (max-width: 991.98px) {
  #adminTable {
    width: calc(100vw - 100px);
  }
}
```

Src/Models

Моделі визначають структуру сутностей, які будуть надходити до БД.

User.js

```
export default {
  email,
  pass,
  nickName,
  avatarPath,
  firstName,
  secondName,
};
```

Src/Helpers

Фунції загального призначення, що виконують допоміжну роботу для основних модулів вебдодатку, як-от: підключення та запит даних до бази даних, завантаження файлу на сервер.

Database-query.js

```
import mysql from 'mysql2/promise';
import config from './db-config.js';

export async function query(sql, params) {
  const conn = await mysql.createConnection(config.db);
  const [results] = await conn.query(sql, params);

  await conn.end();

  return results;
}
```

Db-config.js

```
export default {
  db: {
    host: 'localhost',
    user: 'root',
    // password: 'root',
    database: 'social_media',
  },
};
```

File-upload.js

```
import { fileURLToPath } from 'node:url';
import * as path from 'path';

const __dirname = path.dirname(fileURLToPath(import.meta.url));

function getFileName(file) {
  const imgExt = path.extname(file.name);
  const imgName = path.basename(file.name, imgExt);

  return `${imgName}_${Date.now()}${imgExt}`;
}
export async function uploadFile(file, filePath) {
  const fileName = getFileName(file);
  const uploadPath = path.join(__dirname, filePath, fileName);

  file.mv(uploadPath, (err) => {
    if (err) throw err;
  });
  return fileName;
```

ДОДАТОК Г

Рекомендаційна система пошуку друзів у соціальній мережі. Розробка й аналіз класифікаторів для підтримки прийняття рішень

Денисенко Ф. М., *студент ІТ-01*; Лавров Є. А., *професор*

Сумський державний університет, м. Суми, Україна

Вступ. Зручність використання соціальної мережі багато в чому залежить від наявності у ній системи підтримки прийняття рішень (СППР) для запровадження гнучких та унікальних алгоритмів надання користувачам актуальної інформації. Функціональний аналіз СППР дозволив виявити, що однією з базових підсистем СППР повинна бути підсистема "Пошук друзів". Такий аналіз має проводитись з урахуванням сукупності факторів, що впливають на зручність використання соціальної мережі. Одним з таких факторів є пріоритизація рекомендаційної системи пошуку можливих друзів.

Постановка задачі. Метою роботи є побудова моделі пріоритизації рекомендаційної системи пошуку можливих друзів та реалізація демонстраційного прикладу з використанням різноманітних класифікаторів для створення алгоритму автоматичної пріоритизації.

Результати. Введемо поняття нормованих показників узгодженості інтересів користувачів:

$$\alpha 1 = \frac{\text{спільна кількість підписок к1 та к2}}{\text{загальна кількість підписок к1}} \qquad (1.1),$$

$$\alpha 2 = \frac{\text{спільна кількість підписок к1 та к2}}{\text{загальна кількість підписок к2}} \qquad (1.2),$$

де к1 – користувач 1, а к2 – користувач 2.

Експертним методом сформовано навчальну матрицю для рекомендаційної системи, невеликий фрагмент якої наведено в таблиці 1.

Для оптимізації процесу вибору друзів проведено аналіз деяких класифікаторів (алгоритмів дискримінантного лінійного і квадратичного аналізу, наївного баєсівського класифікатора та методу дерев рішень). Кожен з них приймає на вхід показники α1 та α2. На виході отримуємо клас пріоритизації. Демонстрацію результатів роботи класифікаторів з позначкою значень помилки наведено на рисунку 1.

Таблиця 1. Демонстраційна навчальна матриця для класифікатора (фрагмент).

α1	α2	Пріоритет рекомендації
0,65	0,9	Високий
0,8	0,7	Високий
0,6	0,7	Середній
0,55	0,5	Середній
0,4	0,3	Низький
0,1	0,1	Низький

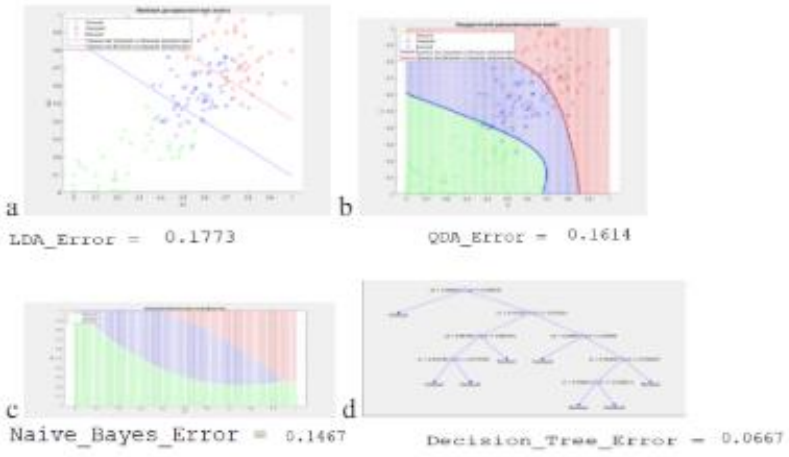


Рисунок 1 – Приклад роботи класифікаторів для рекомендаційної системи пошуку можливих друзів у соціальній мережі (a – дискримінантний лінійний аналіз, b – дискримінантний квадратичний аналіз, c – наївний баєсівський класифікатор, d – метод дерев рішень)

Висновок. Створено модель пріоритизації рекомендаційної системи пошуку можливих друзів. У ході розробки класифікаторів для створення алгоритму автоматичної пріоритизації використано накопичену базу пріоритетів. Для задачі пошуку можливих друзів класифікатори дискримінантного лінійного і квадратичного аналізу, наївного баєсівського класифікатора та методу дерев рішень дають достатньо точні результати, але найбільш точним в умовах використаної навчальної матриці виявився класифікатор, що працює за методом дерева рішень.