НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри інформаційних систем та технологій
_____ М. З. Швиденко
«___»_____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему «Впровадження інтерактивної візуалізації в інтерфейс платформи для демонстрації портфоліо дизайнерів»

Спеціальність 126 «Інформаційні системи та технології»

Гарант освітньої програми

д.т.н, проф. Мокрієв М. В.

Керівник бакалаврської кваліфікаційної роботи

к.т.н., доц. Назаренко В. А.
науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Козленко А.В.
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЛЬТЕТ ІНФОРМАЦІЙНИХ ТНХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри інформаційні системи та технології
_____ М. З. Швиженко
«__» _____ 2025 р.

ЗАВДАННЯ
на виконання бакалаврської кваліфікаційної роботи студенту
Козленко Анастасія Вікторівна
(прізвище, ім'я, по батькові)

Спеціальність 126 «Інформаційні системи та технології»
(код і назва)

1. Тема роботи: **«Впровадження інтерактивної візуалізації в інтерфейс платформи для демонстрації портфоліо дизайнерів»**

Затверджена наказом ректора №2245"С" від 16.12.2024

2. Термін подання завершеної роботи на кафедру – ___06.2025р.

3. Вихідні дані Розробити Веб систему управління замовленнями для підприємства з виробництва металу

4. Перелік питань, що розглядаються:

1. Теоретико-методологічні засади дослідження систем управління виробничими замовленнями

2. Архітектурно-технологічні аспекти проєктування веб-системи управління замовленнями

3. Програмна реалізація веб-системи управління замовленнями

5. Календарний план

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Вибір теми, планування та підготовка	25.02.2025	Завдання виконано
2	Дослідження та збір матеріалів	30.03.2025	Завдання виконано
3	Аналіз та написання	20.05.2025	Завдання виконано
4	Попередній перегляд та оцінка	27.05.2025	Завдання виконано
5	Захист бакалаврської роботи	06.2025	Завдання виконано

Керівник кваліфікаційної роботи _____ / __Назаренко В.А., доктор філософії____
підпис ПІБ, вчене звання та ступінь

Завдання прийняв до виконання _____ / _____Козленко А.В.
підпис

Дата отримання завдання 15.02.2025

РЕФЕРАТ

Тема бакалаврської кваліфікаційної роботи: *“*Інформаційна система для публікації та демонстрації творчих робіт користувачів на платформі BeRaw*”*

Автор роботи: Козленко Анастасія Вікторівна

Керівник роботи: Назаренко Володимир Анатолійович

Пояснювальна записка: 50 с., 23 рис., 2 табл., 15 джерел

Графічна частина: 20 презентаційних слайдів.

ВЕБ-ПЛАТФОРМА, КЛІЄНТСЬКА ЧАСТИНА, FIREBASE, FIRESTORE, REACT, TAILWINDCSS, ІНТЕРАКТИВНИЙ ІНТЕРФЕЙС, ІНФОРМАЦІЙНА СИСТЕМА.

Метою роботи є створення сучасної веб-платформи для митців, що дозволяє публікувати власні роботи, ділитися ними, взаємодіяти з іншими користувачами, а також шукати контент за тегами.

У межах дипломної роботи розроблено та впроваджено **інформаційну систему BeRaw**, призначену для збереження, фільтрації та демонстрації творчих матеріалів. Функціонал реалізовано за допомогою технологій **React** (клієнтська частина), **Firebase Firestore** (база даних), **Firebase Storage** (сховище зображень) та **Firebase Authentication** (авторизація). В системі передбачено зручний та адаптивний інтерфейс, створений за допомогою **TailwindCSS** і **Framer Motion**, що забезпечує приємний користувацький досвід.

Розроблена система підтримує **створення акаунта**, **публікацію постів**, **залишення коментарів**, **вподобайки**, а також **фільтрацію контенту за тегами**. У рамках тестування система показала стабільну роботу, високу швидкість завантаження та позитивний відгук від цільової аудиторії

. ABSTRACT

Bachelor’s Thesis Topic: “Information System for Publishing and Showcasing Users’ Creative Works on the BeRaw Platform”

Author: Anastasiia Kozlenko

Supervisor: Volodymyr Anatoliiovych Nazarenko

Explanatory note: 50 pages, 23 figures, 2 tables, 15 references

Graphic part: 20 presentation slides

Keywords: Web platform, client-side, Firebase, Firestore, React, TailwindCSS, interactive interface, information system

The goal of this thesis is to develop a modern web platform for artists, enabling them to publish their creative works, share them, interact with other users, and explore content based on tags.

As part of the bachelor’s qualification project, an information system called BeRaw was designed and implemented to store, filter, and display creative content. The functionality was developed using React (client-side), Firebase Firestore (database), Firebase Storage (image storage), and Firebase Authentication (authorization). The system features a user-friendly and responsive interface created with TailwindCSS and Framer Motion, providing a smooth user experience.

The implemented system supports account creation, post publication, commenting, likes, and content filtering by tags. During testing, the system demonstrated stable performance, fast loading times, and received positive feedback from the target audience.

Зміст

Перелік умовних позначень

UI — User Interface (інтерфейс користувача)

UX — User Experience (досвід користувача)

SDK — Software Development Kit (набір засобів розробки програмного забезпечення)

CRUD — Create, Read, Update, Delete (базові операції з даними)

ID — Identifier (унікальний ідентифікатор)

Firebase Firestore — хмарна документно-орієнтована база даних

Firebase Storage — хмарне сховище зображень та файлів

Firebase Authentication — служба автентифікації користувачів

NoSQL — нереляційна база даних без фіксованої схеми

React — JavaScript-бібліотека для створення інтерфейсів користувача

TailwindCSS — утилітарна бібліотека CSS для стилізації

Framer Motion — бібліотека для анімації в React

ER-діаграма — Entity-Relationship діаграма, що показує структуру бази даних

Rules-based security — система правил безпеки, яка визначає права доступу

CDN — Content Delivery Network (мережа доставки контенту)

HTTPS — захищений протокол передачі гіпертексту

Вступ

В сучасному світі де цифрові технології дійшли майже в кожний куточок світу та доторкнулися до кожної професії і є невід'ємною її частиною, внесли свої корективи та нові умови праці. Більшість сучасних компаній та користувачів надають вагоме значення цифровій обробці, з багатьох причин: швидко, надійно, зручно.

Ця участь також не оминула і творців, коли музика стала доступна в лічені секунди, щоб переглянути виставу не потрібно вставати навіть з дивану, а картини стали оцифровані.

Але є інша сторона оцифрування творчості, зросла конкуренція, а з ростом конкуренції почали зростати і вимоги до кандидатів. Також через хвилю популярності, легкістю навчання та високою заробітною платою. Люди почали масово створювати великий потік контенту. Що призвело до масового перенасичення ринку праці. Відомі та великі митці витісняють своїх менш відомих конкурентів не даючи можливість для зросту, заполонивши увагу людей на всіх платформах.

Тому для кожного хто хоче знайти свою аудиторію та побудувати міцне уявлення про себе. Була розроблена та створена платформа BeRaw. Кожен хто хоча б раз захоплювався мистецтвом як хобі, бажали продемонструвати свою роботу іншим, але не зробив це через брак можливостей, тому ця платформа саме для того щоб показати себе з певної сторони. А для людей які цим займаються професійно BeRaw стане невід'ємною частиною рутини, настільним атласом. Метою даної роботи є створення веб-застосунку BeRaw - онлайн-платформи для демонстрації портфоліо дизайнерів, що забезпечує простоту в користуванні.

Для того щоб розробити веб-застосунок BeRaw, нами було проаналізовано потреби митців щодо платформи, а також зведено плюси та мінуси самих платформ. Після аналізу було виявлено явні недоліки. У більшості випадків платформи не є зрозумілими у використанні і в результаті чого у користувачів,

особливо в новачків, виникає потреба у вивченні великої кількості відповідної літератури. Алгоритмічне просування контенту залишає багатьох без уваги, а недостатній захист авторства пропускає плагіат. Тому, звернувши на це увагу, я розробила зручний інтерфейс з приємною і звичною кольоровою гамою, були підібрані технології для динаміки веб-застосунку та бази даних.

Для того щоб реалізувати дану платформу, мені потрібно забезпечити всім необхідним, можливістю Реєстрації/Авторизації, створення аккаунта та поширення робіт, також можливістю взаємодіяти та проявляти зацікавленістю учасників до інших.

Об’єктом дослідження є процес створення веб-платформи для взаємодії з візуальним контентом.

Предметом дослідження є використання технологій хмарної інфраструктури Firebase у поєднанні з React для реалізації інтерфейсу та бекенду динамічного веб-застосунку.

У розробці було використано такі технології: React — для інтерфейсу, Firebase Authentication — для автентифікації користувачів, Firestore — для зберігання даних, Firebase Storage — для завантаження та зберігання зображень.

BeRaw створена та реалізована у рамках дипломної роботи, але у майбутньому може бути реалізована і вдосконалена для більшої кількісті користувачів.

Ідея створити BeRaw виникла не просто на рівному місці. Під час власного пошуку платформ для публікації своїх робіт я неодноразово стикалась з тим, що більшість з них або перевантажені, або вимагають спеціального запрошення, або й взагалі мають складну архітектуру, яка відштовхує новачків. Особливо відчутною була відсутність простого місця, де можна просто викласти свій малюнок чи пост без зайвих кроків, де все працює "інтуїтивно", як кажуть — "відкрив, завантажив, поділився".

Також постійно виникала проблема з тим, що твою роботу ніхто не бачить, бо вона “не потрапила в алгоритм” або не набрала “лайків” за перші 15 хвилин. Саме

через такі речі народився задум створити платформу, де всі матимуть рівні шанси, а головна увага буде не на популярності, а на самому змісті роботи. Це і стало ключовою ідеєю BeRaw.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВА ЗАДАЧІ

1.1. Аналіз існуючих програм-аналогів

В цифровому просторі існує безліч платформ для демонстрації дизайну та робіт, взаємодію з аудиторією, та створення особистих портфоліо. Найвідоміші і всім звичні можна виокремити, це три платформи: **Behance**, **Dribbble** та **Pinterest**. Кожна з них має свої переваги та недоліки, що можна враховувати при створенні платформи

Behance

Behance – це веб-застосунок від Adobe, професійна платформа для дизайнерів, ілюстраторів, 3Д-художників, художників та web-дизайнерів, для всих тих хто хоче продемонструвати свої роботи та зібрати професійне портфоліо. Платформа орієнтована на створення повноцінних проектів – вона надає користувачам можливість вигрузки фото, відео, опис. Дизайн самостійної роботи виглядає як презентація розмір якої можна задати в ручну, тобто це дозволяє створювати презентацію робіт як односторінкових сайтів. Behance розрахована на людей які в пошуку роботи або окремих проектів.

Також в платформи є і недоліки: складний інтерфейс перевантажений функціоналом, що потребує додаткових курсів по користуванні платформою. Додаток обмежує в спілкуванні та в коментарях, а особисті повідомлення можливі при умові бути учасником спільноти Adobe. Контент на платформі важко просувати без зовнішніх посилань та втручань.

Dribbble

Dribbble – платформа орієнтована на демонстрацію візуальних концепцій таких як UI-елементів, логотипів та web-дизайн. Аудиторія на яку націлена

платформа - це web-дизайнерів, UI/UX – дизайнерів, ілюстраторів, web-розробників, фрілансерів. Але у Dribbble є недолік, нові користувачі можуть публікувати свої роботи лише після запрошення.

Pinterest

Pinterest - це універсальна платформа для демонстрації робіт яка орієнтована на широку аудиторію, починаючи від ілюстраторів чи графічних художників закінчуючи домогосподарками та DIY-ентузіастами. Платформа ввібрала в себе всі напрямки контенту, рецепти, музика, ремонт авто або будинку.

Pinterest виконує більше роль візуальної соціальної мережі, яка дозволяє користувачам ширити, зберігати, ділитися ідеї у вигляді «пінів». В порівняно з попередніми двома вона має простий інтерфейс з мінімальним набором інструменту. Але із-за того що вона соціальна мережа і в ній великий обсяг контенту, а інколи це сміття, знайти або поширювати професійний контент достатньо важко.

Тож аналізуючи всі платформи я склала табличку в якій виявлено головні недоліки та переваги базових функцій

Таблиця 1.1 Зведені переваги та недоліки відомих платформ

Платформа	**Behance**	**Dribbble**	**Pinterest**
Ціль	Професійне портфоліо	Демонстрація концептів та ідей	Колекціонування візуального контенту
Цільова аудиторія	Дизайнери, художники, UX/UI фахівці	UI/UX дизайнери, графіки, фрілансери	Масова аудиторія, DIY-ентузіасти, блогери

1.2. Постанова задачі

Проаналізувавши вже існуючі веб-застосунки було виявлено недоліки, які пояснюють потребу створити нову платформу. Вона б вирішила наявні претензії великої аудиторії, наприклад: обмежена взаємодія, не контролюється авторство, високий поріг входу новачків та загалом складний інтерфейс. Тому це свідчить про необхідність створення інноваційного та простого веб-застосунку.

Метою даної роботи стоїть розробка веб-платформи BeRaw, що вирішуватиме актуальні проблеми, такі як:

- Захист авторських прав
- Зручний інтерфейс
- Можливість публікації робіт
- Фільтрація контенту
- Оновлення стрічки за алгоритмом
- Можливість взаємодіяти

Також платформа повинна бути технічно стабільною, з приємним «user-friendly» інтерфейсом який відповідає нормам та сучасним стандартам.

1.3. Вимоги до системи

1.3.1 Функціональні вимоги

В нашій системі користувачу передбачено багато сценаріїв дій. Він зможе створювати аккаунт, і в подальшому його редагувати, публікувати пости, залишати коментарі або слідкувати за іншими авторами. Після входу пропонуватиметься реєстрація або вхід. Створення та редагування свого профілю спроектоване за зрозумілими правилами, що забезпечить єдиний стиль та зручність навігації.

Для публікації постів з особистими роботами, потрібно буде додати назву, опис, теги або ключові слова за якими вони в майбутньому будуть відслідковуватись та фільтруватись при пошуку. Також в системі буде передбачена комунікація спільноти в вигляді коментарів і вподобайок для підтримки, прояву поваги до автора та його роботи-витвору.

1.3.2 Вимоги до інтерфейсу

Інтерфейс орієнтований на простоту, доступність і комфорт користувача, незалежно від того, який він має досвід з подібними системами

Головний критерій при розробці - інтуїтивно зрозумілий інтерфейс без перевантаження зайвими елементами з повним фокусом на візуальний контент. Розроблений в монохромній палітрі, щоб не конкурувати з роботами

користувачів. Сірий колір платформи буде гармонійно взаємодіяти з роботами та не створювати дискомфорт при перегляді, акцентуючи увагу глядача саме на постах, без відволікаючих факторів на фоні. Шрифти підбиратимуться достатнього розміру, з акцентом на зручність читання, без надмірної декоративності, щоб не ускладнювати сприйняття інформації.

1.3.3 Технічні вимоги

Після аналізу технічних потреб нами було прийнято рішення розроблять платформу BeRaw за допомогою фреймворку React, для забезпечення високої швидкості роботи, компонентної структури та гнучкості при розробці інтерфейсу. Бібліотеки TailwindCSS для стилізації та Framer Motion для виконання анімацій без відчутної втрати продуктивності.

Також в системі повинні бути застосовані технології для зберігання зображень-постів та користувачів, для цієї реалізації використаємо Firebase Firestore – для зберігання даних, постів, користувачів, коментарів тощо. Та Firebase Storage – для зберігання медіафайлів, в основному зображення, що прикріплюються до постів. Ці інструменти забезпечать масштабне зберігання та швидкий доступ до даних

Безпека системи підтримується та обмежується Firebase Authentication, де можна реалізувати різні сценарії авторизації.

У платформу впровадимо можливість реєстрація та авторизація через Google-аккаунт, ввійти в систему можна буде за допомогою пошти та паролю, що робить систему безпечною та універсальною для користувачів із різними потребами.

Права доступу налаштуються за допомогою механізму rules-based security, цим самим забезпечимо захист персональних даних і авторського контенту від несанкціонованих доступів в залежності від прав користувача.

Для розгортання платформи було обрано Firebase Hosting, тут ми забезпечимо стабільну роботу веб-застосунку, швидке завантаження сторінок завдяки CDN, також впровадимо підтримку HTTPS-протоколу для захищеного з’єднання.

Таблиця 1.2 Технології, які використаються для виконання завдання

№	Технологія	Призначення
1	React	Фреймворк для побудови інтерфейсу користувача (UI), компонентний підхід
2	TailwindCSS	CSS-бібліотека для швидкої та зручної стилізації елементів інтерфейсу
3	Framer Motion	Бібліотека для реалізації плавної анімації та переходів у React
4	Firebase Firestore	Хмарна база даних для зберігання текстових даних: профілі, пости, коментарі
5	Firebase Storage	Сховище для зображень та інших медіафайлів, прикріплених до постів
6	Firebase Authentication	Авторизація та реєстрація користувачів (Google, пошта + пароль)
7	Rules-based security	Механізм для обмеження доступу до даних за ролями/правами користувачів
8	Firebase Hosting	Розгортання застосунку, CDN-доставка контенту, підтримка HTTPS

1.4. Опис предметної області

Веб-платформа BeRaw застосовується в візуальному мистецтві, графічному дизайні, творчих ілюстраціях та професійних веб-дизайнах. У сучасному цифровому просторі митці, як професіонали так і аматори потребують ефективних потужних інструментів для роботи та її демонстрації.

Платформою можуть користуватися не лише ті, чия робота є прямо пов’язана з створенням картин або професійної графіки, а й маленькі та великі ентузіасти, студенти та всі охочі, які прагнуть поділитись своїм талантом але не знайшли місце для свого хисту у перевантажених стороннім контентом та рекламою соціальних мережах чи у занадто складних, не зрозумілих і часто закритих для новачків платформах.

Тому ця система вирішить багато проблем та задовольнить основну масу потреб творчої спільноти. Одне з ключових рішень і є веб-платформа, де кожен знайде своє місце. Цю роль забезпечить головна сторінка – стрічка на якій будуть розміщуватись пости з завантаженими роботами іншими користувачами незалежних від ваших вподобань чи алгоритмів.

Оновлюватиметься стрічка за датою поста, а не за популярністю, забезпечуючи для всіх рівні можливості. Також проблем в використанні нашого

продукту не буде, все буде легко та інтуїтивно, не доведеться вивчати купу додаткових ресурсів і проводити години вивчаючи будову інтерфейсу заради того, щоб просто розмістити свою роботу.

Платформа створена для людей які хочуть ділитися прекрасним та чудесним з оточуючими, картини, графічні малюнки, ілюстрації та дизайн сайтів. Системою, для підтримки якісного творчого простору, буде заборонятись розміщувати нецензурний та різний низькоякісний контент, для цього ми запровадимо ШІ, який перед цим пройде навчання на великій кількості дата-сетів.

Основними об’єктами системи є: користувач, пости, теги, коментарі

Користувач матиме особистий кабінет с основною інформацією про себе. До кожного прив’язуватиметься завантажений ним пост, який містить зображення та повноцінний опис. Взаємодія з постами проходить через фільтрацію за тегами, система шукатиме відповідні теги і виводитиме пов’язаний з ними результат. Важливою частиною нашого проекту - передбачена змога коментувати публікації інших людей, що і виконуватиме функцію соціальної взаємодії для ведення діалогів і висловлювання думок.

І не менш важливою є функція як підпис на користувача, тобто достатньо лише перейти в розділ «Підписки» та знайти роботи улюбленого автора в один клік, не шукаючи їх кожного разу вручну.

Розділ 2: ПРОЄКТУВАННЯ СИТЕМИ

2.1 Бізнес-процеси та сценарії використання

Веб-платформа BeRaw будується на простих та логічних діях і саме це робить її зручною у користуванні та не вимогливою до рівня підготовки новачків.

Взаємодія користувача з платформою BeRaw розпочинається зі сторінки «вхід». В загальному взаємодія користувача з системою та сама робота системи побудована навколо простого послідовного ланцюга дій, що включає роботу системи: оновлення стрічки, фільтрація постів, збереження даних. Наприклад, уявімо сценарій для нового користувача. Він відкриває платформу і бачить два варіанти: кнопка «Увійти» та кнопка «Зареєструватись». Після натискання на «Зареєструватись» з’являється сторінка з полями: пошта, пароль, підтвердження паролю. Пароль перевіряється на складність — має бути не менше 8 символів, з великою, малою буквою та цифрою. Якщо пароль не відповідає — поле одразу підсвічується червоним і з’являється підказка.

Коли користувач успішно проходить реєстрацію, система автоматично перекидає його на сторінку **створення акаунта**. Там є 4 основних поля: завантажити аватар (зображення), ввести псевдонім (нікнейм), коротко написати про себе (до 200 символів), і додати теги (до 5 штук, через кому). Біля кожного поля — пояснення маленьким сірим текстом, щоб не плутатися. Якщо псевдонім уже зайнятий, система не дозволяє продовжити — одразу пише «такий нік уже існує». Після заповнення всіх полів кнопка «Опублікувати» стає активною, і користувач потрапляє на головну сторінку.

Головна сторінка — це стрічка постів інших користувачів, яка завантажується за датою публікації. Ніякого алгоритмічного сортування. Новий користувач може одразу натиснути на іконку плюса в боковій панелі — відкриється сторінка «Створення поста». Там принцип такий самий як при створенні акаунта: додати фото, придумати назву, написати опис, додати теги. Знову ж — всі поля перевіряються, і якщо щось не так, система підказує, що саме виправити. Наприклад, якщо фото не обрано або розмір перевищує 5MB — буде повідомлення: «Файл надто великий або має неправильний формат».

Таким чином, з моменту відкриття сайту до першого опублікованого поста проходить буквально кілька хвилин. Усі кроки логічно пов’язані між собою, система не дає зробити помилку або загубитися в інтерфейсі. Це все було спеціально продумано і протестовано на прикладах справжніх людей, які вперше відкривали платформу. Наша задача — зробити так, щоб користувач відчував себе впевнено на кожному етапі. Взаємодія між користувачами: підписи, коментування постів, вподобайки. Та користувачів з системою: створення та поширення робіт, реєстрація/авторизація, створення профілю. Функції системи: зберігає данні про користувача , оновлює стрічку за датою поширення робіт, фільтрація робіт за тегами.

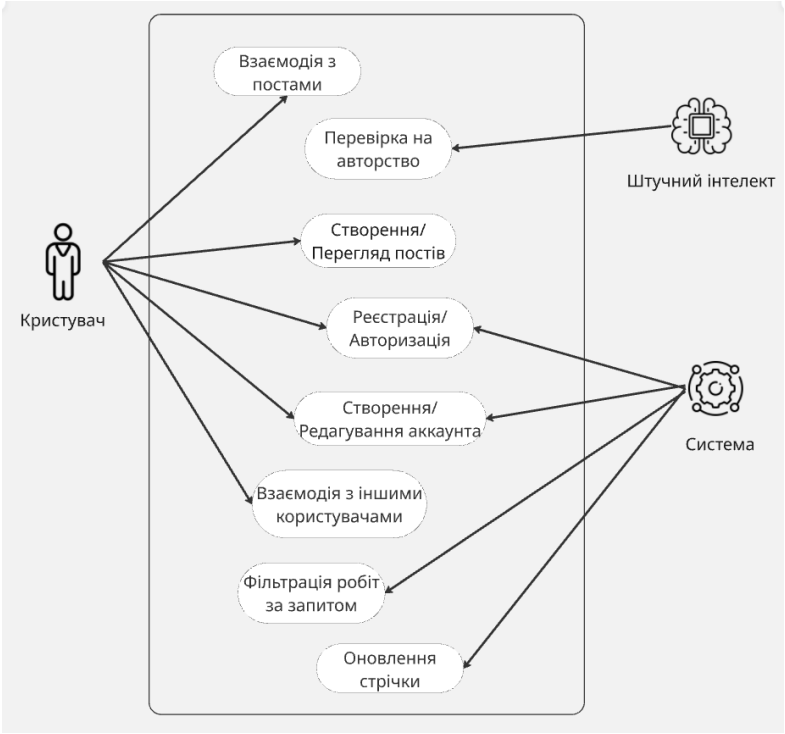


Рис 2.1 – Діаграма сценаріїв взаємодії користувача та платформи BeRaw

З побудованої діаграми сценаріїв видно, наша платформа є лінійною і орієнтованою но простоту, що в свою чергу забезпечить ефективну роботу без перевантажень інтерфейсу.

2.1.1 Реєстрація та створення публікації

Перед тим щоб повноцінно взаємодіяти з платформою користувачу потрібно зареєструватися та створити аккаунт у випадку, якщо це новий користувач для системи.

Відкривши платформу, користувач обирає «Реєстрація», далі система створює запит на введення пошти для ідентифікації користувача (її ніхто не бачитиме крім нього самого) та на введення паролю два рази, перший необхідно ввести свій пароль а другий це його підтвердження, критерії при написанні паролю: латиниця верхнього та нижнього регістру, цифри від 0 до 9, а також набір спеціальних символів для кращого захисту. Коли система підтверджує, що користувач пройшов успішну реєстрацію його перенаправляють на сторінку

«створення акаунта», якщо ж при перевірці введених даних система вибила помилку, то причиною цьому буде некоректне введення пошти або паролю, продовжити користувач зможе лиш тоді як виправить помилки і повторно пройде реєстрацію.

Нами також передбачено помилку при реєстрації в тому випадку, коли введена пошта вже створена та існує в базі, щоб уникнути дублів та лишніх взаємодій в системі. Після появи відповідного повідомлення користувач зможе просто зайти в свій уже створений профіль та продовжити роботу.

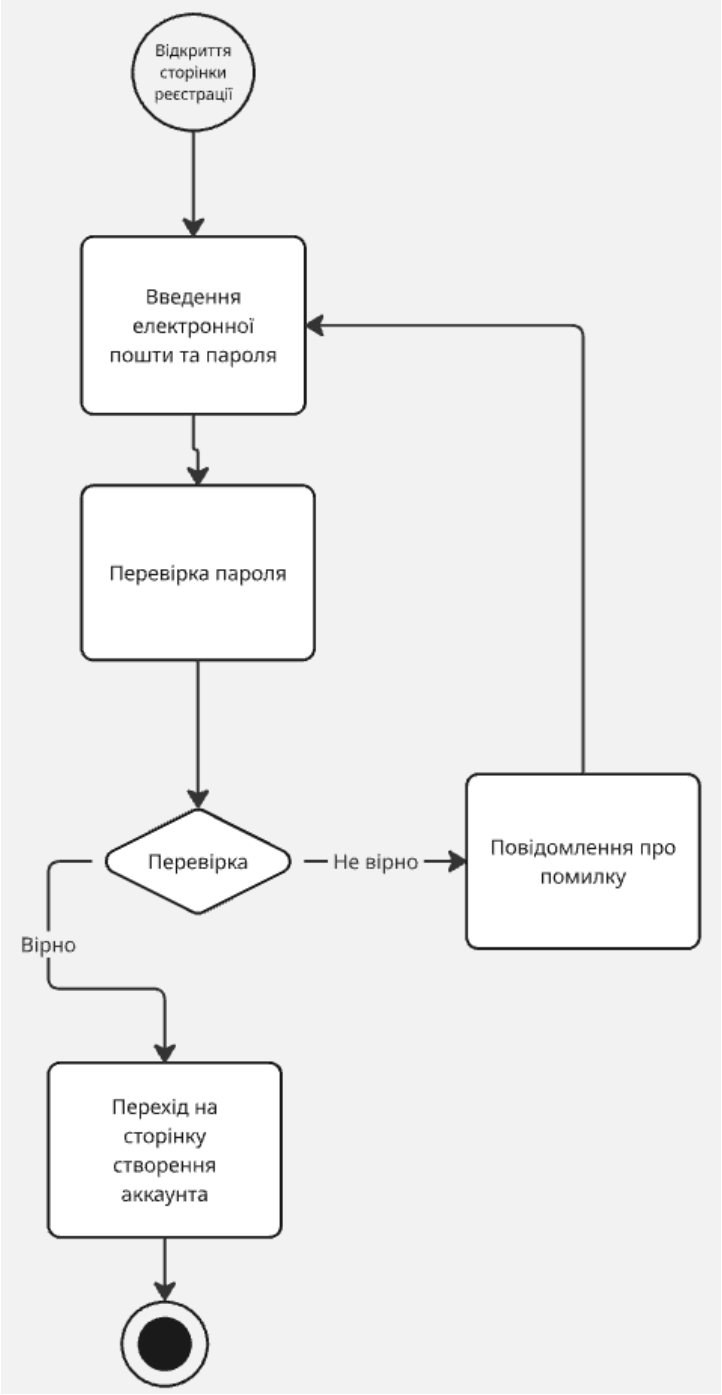


Рис 2.2 – діаграма діяльності «Реєстрація»

2.1.2 Створення акаунта

Коли користувач зареєструвався він потрапляє на сторінку «створення акаунта» де йому потрібно заповнити всі необхідні дані, згідно підказок, що будуть біля кожного поля введення.

На цьому етапі завантажується зображення для ролі аватару, грубого обмеження не буде, лиш, щоб аватар не порушував цензуру або інші морально-суспільні правила.

При введенні свого імені-псевдоніму людина повинна ввести унікальне слово чи словосполучення, за яким і здійснюватиметься пошук в системі, умова щодо написання це латиниця та існування тільки одного єдиного такого варіанту псевдоніму аби уникнути «однакових» користувачів. Введення опису – коротко розповісти про себе , де кожен може написати, необхідну на його думку інформацію.

Системою забороняється вписувати більше 5 тегів, вони виступають як опис але більше про напрямок в якому люди займається. Після того як всі необхідні поля будуть заповнені, передбачено натискання «опублікувати».

При створенні акаунта всі поля повинні бути заповненні, якщо система перевірила на заповнення всі поля успішно, то вона створює аккаунт та направляє користувача на головну сторінку, якщо ж не всі дані будуть внесені або псевдонім уже існуючий, всі проблемні поля вводу будуть підсвічуватись червоним, таким чином повідомляючи про помилку.

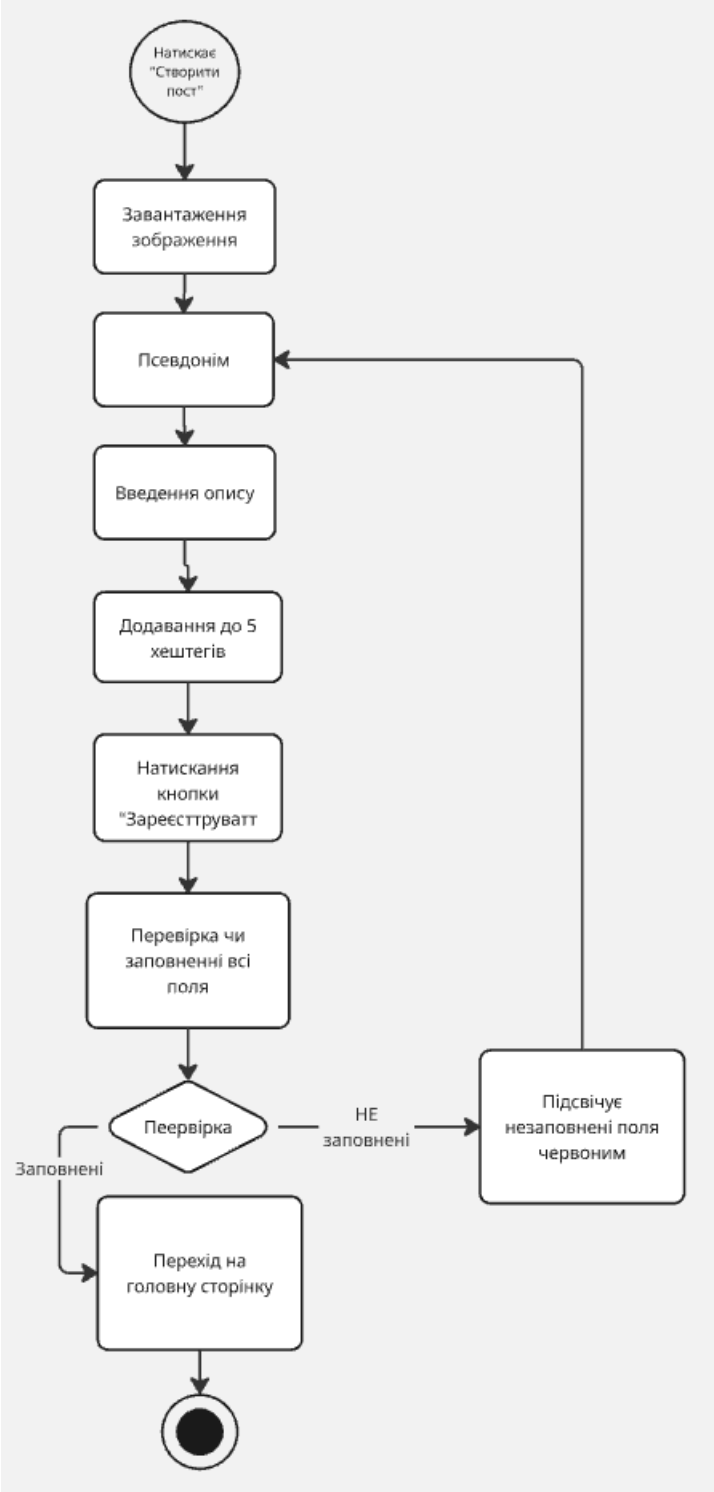


Рис 2.3 – Діаграма діяльності «Створення аккаунта»

2.1.3 Створення поста

Функція «створення поста» дуже схожа до «створення акаунта», різниця лише в тому, що коли створюється акаунт, то описується користувач в системі, коли ж

створюється публікація то описується саме додана робота. Додаєм фото, описуємо суть її створення, задум, а теги за принципом відповідності теми і жанру, щоб пост можна було знайти за ними при фільтрації.

Після перевірки в системи можна буде активувати «поширити», але при виявлені некоректного заповнення чи порушення правил, людина побачить про це повідомлення, виправити чи доповнити необхідно буде в підсвічених червоним полях, аби поширити в простір.

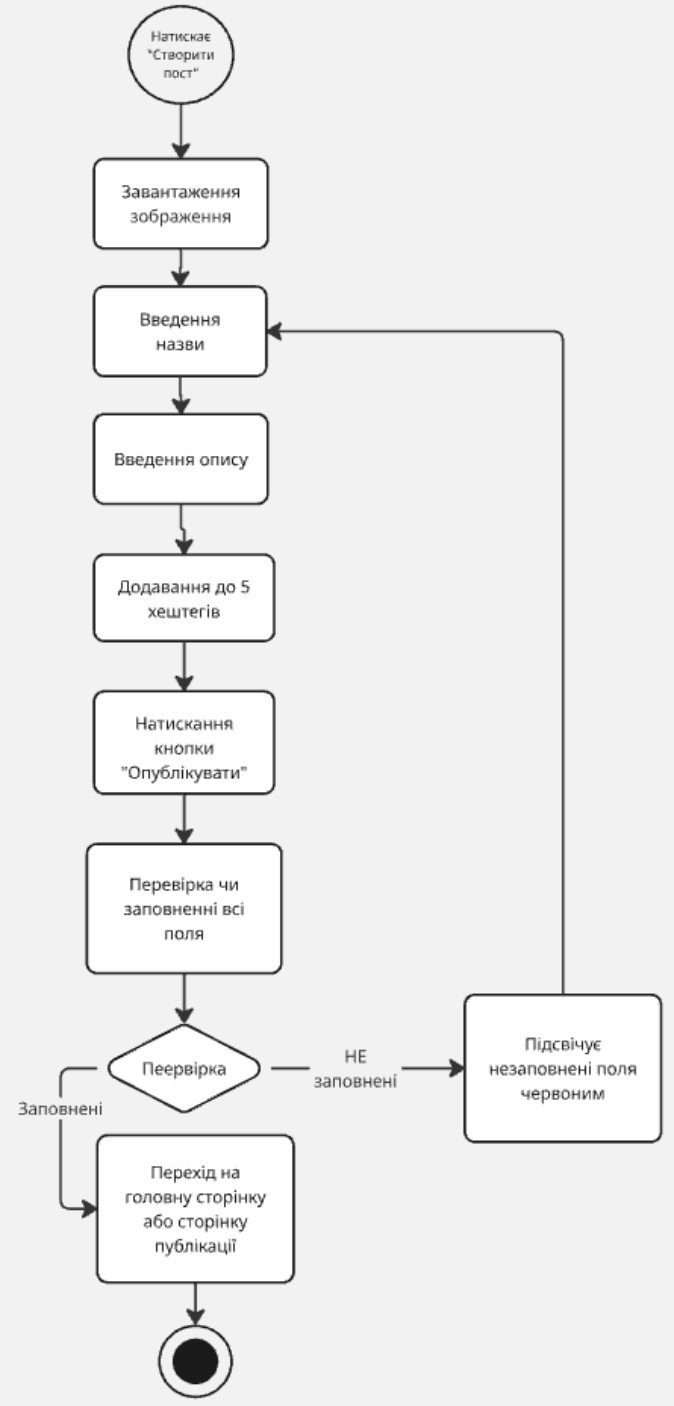


Рис 2.4 – Діаграма діяльності «Створення поста»

Задля економії сторінок записки в додатку Б можна знайти діаграми діяльності «Пошук користувача» «Взаємодія з постом» та «Фільтрація постів»

В діаграмі «Пошук користувача»: у нас є особистий акаунт де є підписки – слідкування за іншими людьми. Зайшовши туди гортаємо наші підписки, знаходячи потрібне. Перейшовши на ту необхідну сторінку перед нами розгортається профіль іншого користувача, де ми зможемо побачити

інформацію про нього. Тут можна буде ознайомитись з усією актуальною інформацією і його твореннями .

Рис 2.5 – Діаграма діяльності «Пошук користувача»

Також відвідати сторінку іншого акаунту можна через пост, якщо під час пошуку знаходимо той який сподобався, вгорі над постом є псевдонім та аватар користувача, натиснувши на аватара, нас перенаправить на сторінку цього користувача. Якщо сподобався користувач можна підписатися, або якщо користувач згодом перестав подобатись можливо буде відписатися, повторно тицьнувши на відповідну кнопку.

На нашій платформі як уже раніше згадувалось, реалізовуватиметься користувацька взаємодія з публікаціями, а саме коментарі та система вподобайок. Відкривши «коментарі», збоку з’явиться сторінка для коментування, де є поле вводу особистих думок, кнопка надіслати, та в горі якщо вже хтось прокоментував, залишені повідомлення інших учасників спільноти. Для відмітки поста вподобайкою достатньо буде натиснути на відповідну іконку поруч з основним контентом, і вона змінить колір на інший

Рис 2.6 – Діаграма діяльності «Взаємодія з постом»

2.2 Проєктування інтерфейсу користувача

Платформа має багато сторінок, але основних небагато. Перша сторінка – сторінка «вхід в систему», яка пропонує два варіанта входу

Дві наступних відповідають за «реєстрацію» та «авторизацію», потім якщо користувач не зареєстрований то сторінка «створення аккаунта». Після процесу входу, нас перекидає на «головна сторінка» на якій є функції : «фільтр постів», «головна», «створення поста», «перехід на особистий аккаунт», та логотип самої платформи.

Якщо ми використовуємо фільтр, в нас з’являється бокова панель де ми вводим дані для відфільтровування постів за тематикою. Функція створити новий пост, переносе нас на сторінку «створення пост» де ми вводим все необхідне та натискаєм «створити». Ми опиняємось на сторінці створеного поста. Далі у нас є декілька дій: вподобати, прокоментувати, редагувати. Якщо редагувати то сторінка «редагування посту» схожа до сторінки створення посту, але додаткова функція це видалення. Якщо вподобати то серце змінить колір, а якщо прокоментувати перед нами з’явиться бокова панель з коментами та полем вводу для коментування. Також внизу з’являються схожі роботи за тегами які ми ввели

Над постом є аватар (круг з фото) та псевдонім якщо натиснути на псевдонім нас перенесе на сторінку особистого профілю або профілю іншої людини, в залежності від того хто створив пост.

На сторінці особистого профілю видно наші дані які ми вводили при реєстрації, підписки – на кого ми підписані, та підписники – той хто на нас. Ми можем перейти на ці сторінки, погортати та перейти на необхідний профіль. Поряд з ними є роботи які ми створювали раніше, натиснувши на них система перекидає на інші сторінки де вже можем передивитися всі які були нами створені. Поряд

Також у нас є функція «редагування профілю» сама сторінка ідентична до сторінки реєстрації, але з функціями «видалити» ти «зберегти».

На сторінці іншого користувача функціональність профіля не відрізняється від нашої окрім функції редагування, заміна на функцію «підписатися»

Багато сторінок створені схожі один на одного з мінімальною різницею у функціональності, це створено заради того щоб було легко та інтуїтивно розібратися з функціоналом.

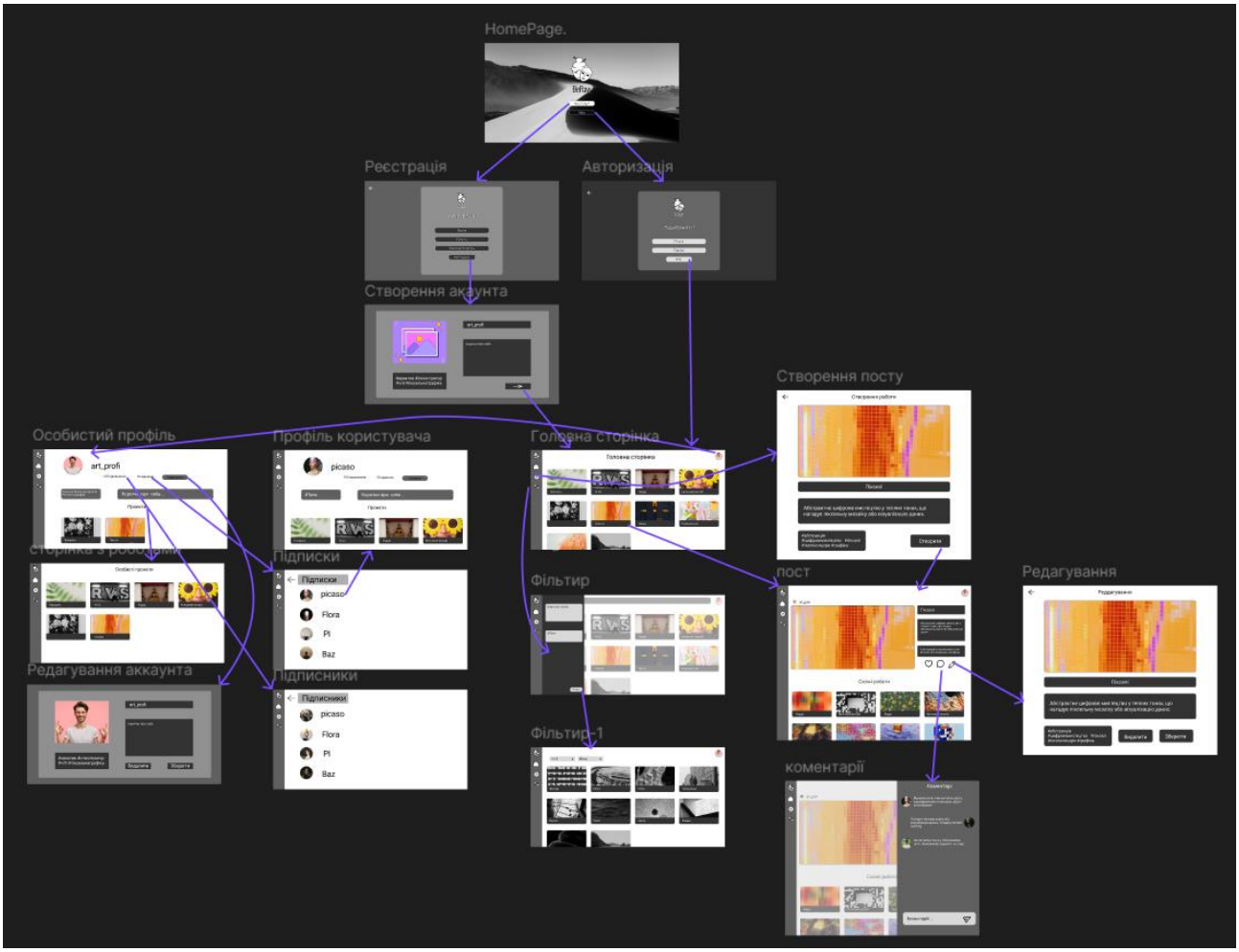


Рис. 2.7 – Структура інтерфейсу платформи BeRaw та переходи між сторінками

Додатково в інтуїтивному використанні сайту допомогли UI\UX навички, кольорова палітра в чорно-білому кольорі, всі кнопки знаходяться послідовно або на тих місцях де дуже ймовірно знаходились б на інших сайтах якими найчастіше користуються люди. Наприклад, кнопка для створення нового поста завжди знаходиться в одному місці — в боковій панелі з іконкою плюса. Ця іконка не просто стоїть як картинка — вона оживає при наведенні: злегка збільшується і змінює відтінок, що підказує користувачу, що тут можна натиснути. Це зроблено через анімації Framer Motion, які додають плавності й не створюють відчуття «рваного» переходу.

Фільтр постів реалізовано у вигляді бокової панелі, яка з’являється з плавним зсувом зліва — користувачу не потрібно шукати, де саме вводити теги. При

натисканні на іконку фільтра, панель розсувається з анімацією, а хрестик для закриття — завжди помітний і знаходиться зверху праворуч. Це важливо, бо користувач відчуває контроль і може в будь-який момент повернутись до стрічки без стресу.

Інтерфейс побудований навколо логіки “мінімум відволікань” — наприклад, кольорова палітра платформи витримана в сірому, неяскравому тоні. Це зроблено для того, щоб самі роботи (зображення, пости) були у фокусі, а фон не «забивав» увагу. Коли людина гортала сторінку, її погляд автоматично прямує до контенту, а не до кнопок.

Навіть у формі створення акаунта всі поля мають однаковий розмір, однакові відступи, однакові підказки — тому користувач не втрачає орієнтації навіть на мобільному. Переходи між сторінками виконуються без перезавантаження, все працює “на місці”, і це створює відчуття, ніби сайт — це додаток.

Там де потрібно записувати або вставляти зображення то є надписи або привичні картинки – іконки в мінімальному стилі. Якщо ж нема полів вводу, то є кнопки у вигляді іконок що нагадують слова які б писались замість них.

2.3 Вибір та моделювання бази даних

Для реалізації зберігання даних платформи BeRaw було обрано Firebase Firestore – хмарна документно-орієнтована база даних від google

Ця база даних відноситься до NoSQL-структур, що дозволяє гнучко зберігати структуровану інформацію у вигляді колекцій і не потребує у визначенні таблиць.

У СУБД яку я обрала є переваги: не потребує власного сервера або хостингу, підходить для маленьких так і великих проектів, велика швидкість та встановлена безпека Secury Rules – це правила які визначають у якого користувача який рівень доступу, а також аунтифікація.

Сутності та поля моєї бази даних

- Сутність users – користувачі
- Зберігає інформацію про зареєстрованих користувачів
- Поля:
- usersID – ідентифікаційний номер
- nickname – псевдонім користувача
- email – пошта
- avatarURL – фото користувача
- bio – короткий опис «про себе»
- tags – теги
- followersCount – кількість підписників
- followingCount - кількість підписок
- createdAT - дата створення акаунта

Сутність posts – пости

- Збереження даних про пости
- Поля:
- postID – індифікаційний номер помста
- userID – індифікаційний номер користувача
- title – назва поста
- descripyion – опис роботи

- imageURL – посилання на зображення
- tages – теги(не більше 5)
- createdAt – дата створення

Сутність comments – коментарі

- Зберігає дані про коменти під постами
- Поля:
- commentID – ІД коментаря
- postID – ІД поста
- userID – ІД користувача
- text - текст
- createdAt – дата

Сутність likes – вподопбайки

- Взаємодія з постами (вподпобайка)
- Поля:
- likeID – унікальний ІД
- postID – Ід поста
- userID – ІД користувача

Сутність followers/following – підписники/підписки

- Зберігає інформацію про того хто підписався і на кого

Сутність tags – теги

- Зберігає список тегів для аналітики, швидкого пошуку, фільтрації
- tagID – ІД тега
- name – ім’я тега
- usageCount – кількість використаних у постах
- createdAt – дата створення

- lastUsed – дата останнього використання

Зв’язки між сутностями

- Користувач → пости: один користувач може створювати багато постів
- Користувач → коментарі: один користувач може написати багато коментарів
- Пост → коментарі: один пост може мати багато коментарів
- Користувач → лайки: один користувач може вподобати багато постів
- Пост → лайки: один пост може отримати багато вподобайок
- Користувач → підписки/підписники: один користувач може підписатися на багатьох інших користувачів
- Пост → Теги: один пост може мати багато тегів; один тег може бути в багатьох постах

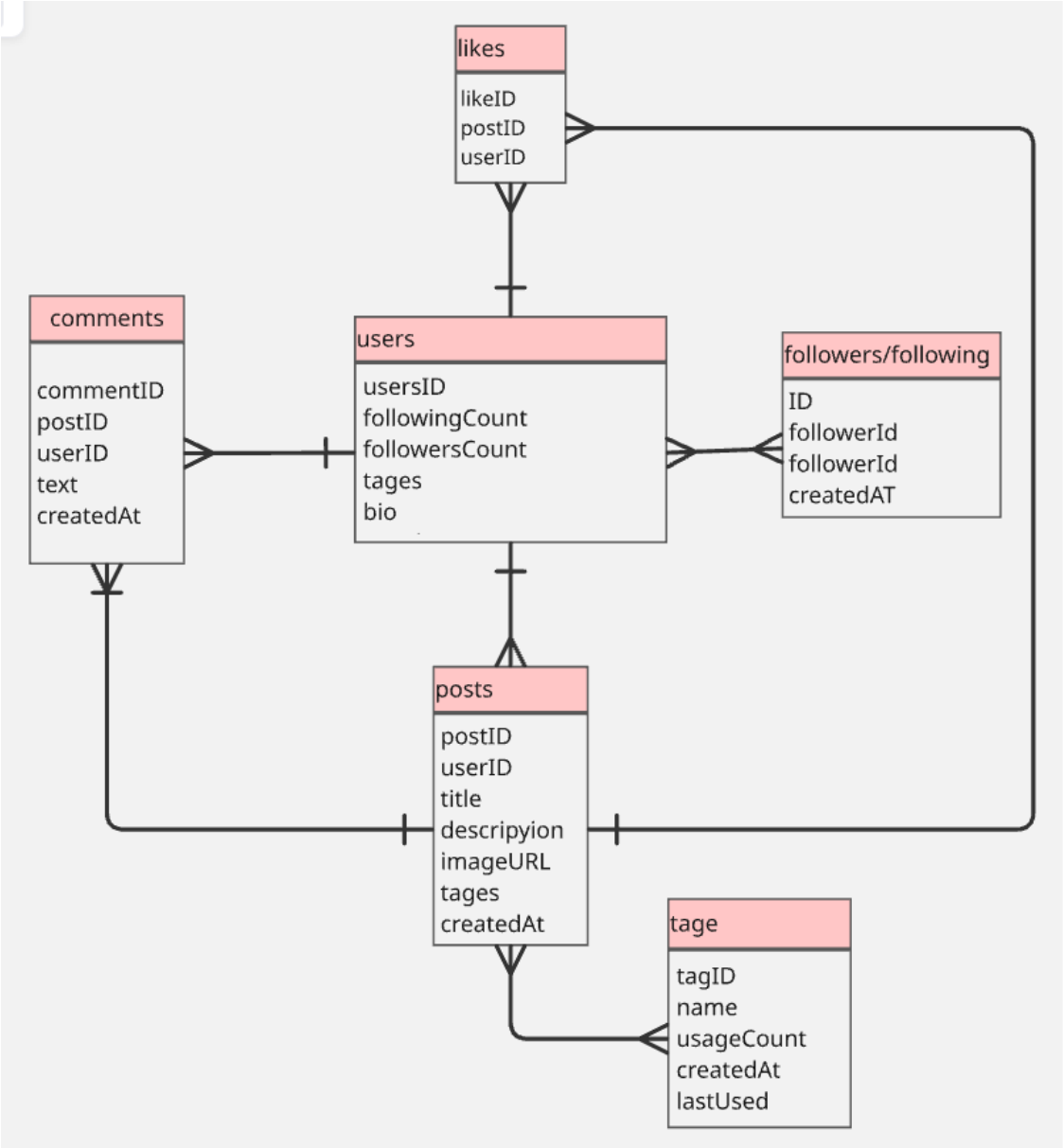


Рис 2.8 База даних системи BeRaw

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

3.1. Вибір технологій та стеку реалізації

Для реалізації веб-застосунку було обрано декілька технологій які разом стикувались. Для клієнтської частини було обрано React, який дозволяє створювати динамічні застосунки на основі компонентів.

У мене було декілька варіантів для реалізації клієнтської частини. Вибір був між React та Angular, обидва достатньо популярні та широко використовувані, але їх підхід до реалізації, структура та концепції суттєво відрізняються

Angular- високорівнева мова програмування, великий і повноцінний фреймворк на основі TypeScript, а головне типізована. Цей інструмент чудово підходить до масштабних проектів, для великих компаній, де потрібна строга типізація. Angular надає всіх необхідні інструменти для створення проекту, підключення до сервера, швидкого оновлення даних, модуляцію. Але як для реалізації на цей час не великого проекту Angular виявився навпаки громістким і зайвим.

Отже вибрано було React - в свою чергу він прости, швидкий та гнучкий що дозволяє розробляти користувацький інтерфейс, а велика еко система дозволяє вирости до повноцінного фреймворку, із-за своєї простоти він зручним у використанні та не перевантажує сторінку. Також в силу своєї гнучкості він не нав’язує типізацію, а також можна самостійно обирати бібліотеки для маршрутизації, легко інтегрується з іншими середовищами.

У контексті розробки веб-застосунку, де головними критеріями є швидкість, простота та інтеграція з Firebase React стає фаворитом, він допоможе підключити сторінку, щоб вона оновлювалась в реальному часі, не потрібно додаткових файлів для стилізації сторінок, а достатньо лише одного для кожної сторінки з стилізацією TailwindCSS. Розроблена система зможе нормально функціонувати з підключеними бібліотеками TailwindCSS – стилізації інтерфейсу.

TailwindCSS – утилітарний CSS-фреймворк, який дозволяє створювати та стилізувати платформу без окремих файлів. Грубо кажучи стилізація проходитиме прямо в класах.

Приклади рядків коду з бібліотекою

```
<button className="px-4 py-2 bg-black text-white rounded hover:bg-gray-800 transition">
  Створити пост
</button>
```

Рис 3.1 Кнопка з адаптивним стилем

Також це допомагає в швидкому створенні адаптивної верстки, та ідеально підходить для мінімального дизайну.

TailwindCSS набагато краще чим звичні нам стилі чи бібліотеки.

Framer Motion – анімація

Framer Motion – це інструмет для створення анімації в React, вона надає потужний API. За допомогою бібліотеки можна створити плавний перехід між сторінками, появу/зникнення об’єктів, перетягування. Зазвичай бібліотека використовується для анімації карток, постів, для відкриття та закриття бокових панелей.

```
import { motion } from "framer-motion";

<motion.div
  initial={{ opacity: 0 }}
  animate={{ opacity: 1 }}
  transition={{ duration: 0.5 }}
>
  <h2>Новий пост</h2>
</motion.div>
```

Рис 3.2 Елемент з плавною появою

React Router - маршрутизація

Бібліотека для створення маршрутизаціями між сторінками, працює без перевантаження сторінок. В веб-застосунку використовується маршрутизація

між головною сторінкою, особистим профілем, сторінкою з постом та фільтрацією

```
import { useNavigate } from "react-router-dom";

const navigate = useNavigate();

<button onClick={() => navigate("/profile")}>
  Перейти в профіль
</button>
```

Рис 3.3 Приклад маршрутизації

Firebase SDK – робота з базою даних

Firebase SDK вона включає всі модулі для роботи з Firebase. А саме для бази даних, реєстрації/аутентифікації, зображень, розгортання додатку. Також допомагає інтегрувати клієнтську частину з бекендом без додаткового серверного коду

React Hook Form бібліотека використовується для валідації полів, їх довжини. Створення форм та показу помилок. На платформі використовується до акаунтів, а точніше їх створення або редагування.

3.2. Реалізація клієнтської частини

Веб-застосунок BeRaw має чітку логічну структуру, що включає такі основні сторінки: він побудований на основі React , із використанням функціональних компонентів. Ось основні компоненти:

HomePage.jsx – головний екран, стартова сторінка

Login.jsx – форма входу в систему

Register.jsx – форма реєстрації нового користувача

MainPage.jsx – головна сторінка стрічки постів

CreatePost.jsx – створення нового поста

Project.jsx – перегляд окремого поста

Profile.jsx – сторінка власного профілю

OtherProfile.jsx – перегляд профілю іншого користувача

Кожна з цих сторінок виконує свою конкретну функцію, і всі вони працюють разом як єдина логічна система. Наприклад, **Login.jsx** — це сторінка входу до системи. Вона проста, без зайвих полів: пошта + пароль, кнопка «Увійти» та стрілочка «назад». Усі поля перевіряються на коректність — якщо щось не так, одразу підсвічується червоним і з’являється пояснення. Таке ж саме стосується і **Register.jsx**, але з додатковим полем для повторення пароля. Тут також враховані правила безпеки: пароль повинен бути не надто коротким, мати цифри і букви — інакше не пропустить.

Далі — **AccountCreate.jsx** (у файлі може мати іншу назву, але логіка одна): користувач вводить свій нікнейм, про себе, додає аватарку, і вибирає до 5 тегів. Тут важлива деталь: всі обмеження працюють одразу — якщо написав більше 5 тегів, поле стає червоним і показує, що треба прибрати зайві. А якщо не завантажено фото, кнопка «Опублікувати» неактивна. Такі дрібниці значно зменшують кількість помилок.

Компонент **MainPage.jsx** — це стрічка постів. Реалізована сіткою 4 в ряд, яка автоматично адаптується під ширину екрану. Пост — це окрема картка, яка має

картинку, назву, і при натисканні відкривається в повному вигляді. Важливо, що ніякі сторінки не перезавантажуються — все зроблено через **React Router**, тому перехід між компонентами миттєвий. Якщо ми натискаємо на пост — відкривається **Project.jsx**, де видно зображення, опис, коментарі, теги.

Компоненти **Profile.jsx** та **OtherProfile.jsx** дуже схожі. Один показує твої дані, інший — профіль будь-якого користувача. Важливо, що інтерфейс у них майже ідентичний — щоб користувач не губився. Різниця лише в тому, що у себе ти бачиш кнопку “Редагувати профіль”, а в іншого — “Підписатись”. Це створює відчуття послідовності та стабільності всередині платформи.

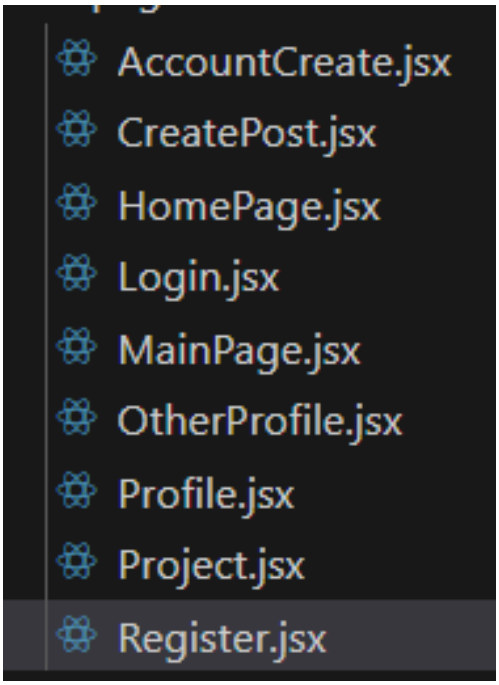


Рис. 3.4 Вигляд основних компонентів

Всі сторінки стилізовані за допомогою стилізації, яку я вивила у окремі файли для кращої читабельності та чистоти коду. Окремий файл має окрему стилістику до сторінки.

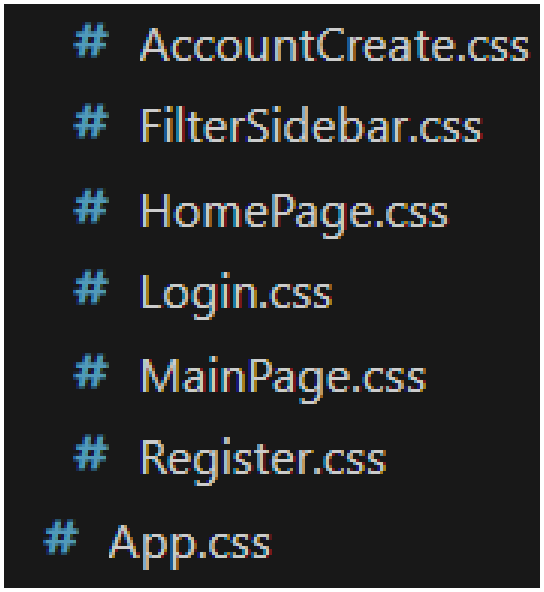


Рис. 3.5 Файли стилізації

Фактично, щоб не заплутатись файли зі стилями мають ідентичні назви до основних файлів зі сторінками.

Також дуже важлива навігація між сторінками, реалізація переходів React Router v6 він використовує компоненти <Routes>. Та навігація яка виконується через хук useNavigate():

Це забезпечує зручну навігацію по сайту без перезавантаження сайту. Також це забезпечує економність, бо не перевантажує сторінки. Можна гнучко керувати маршрутами та створює відчуття цілісності платформи без ривків та очікувати.

Якщо ми не будем використовувати цю технологію, то користувачі будуть страждати від незручності використання. Платформа яка передбачає під собою зручне користування з плавним переходом між сторінками.

Також, щоб система не конфліктувала з користувачем підключається бібліотека React Hook Form, тому в подальшому коли користувач буде взаємодіяти з платформою йому потрібно буде заповнення певних полів які в подальшому піду в сховище баз даних.

React Hook Form – бібліотека допомагає запобігти введення некоректних даних або порожніх даних, зменшення кількості помило і звернення до бази

даних, покращує інтерфейс миттєвим виводом інформації про помилки. Також що дуже важливо для платформи, яка повинна працювати та реагувати миттєво.

3.3. Реалізація серверної частини та бази даних

За серверну частину платформи BeRAW відповідає екосистема Firebase. Завдяки таким технологіям не потрібно створювати окремий сервер, обробка даних, автентифікація та зберігання в базі даних відбувається за рахунок SDK Firebase.

Клієнтський інтерфейс React на пряму взаємодіє з Firebas через модуля

- Authentication – для створення акаунтів, їх редагування та входу/виходу
- Firestore – для збереження текстових даних
- Storage – для зберігання зображень
- Firebase Rules – для контролю доступів до ресурсів

Підключення до Firebase реалізується через конфігураційний файл firebase.js, який ініціалізує додаток та надає доступ до всіх сервісів через функції на рис. 3.3.1

```
1  import { initializeApp } from "firebase/app";
2  import { getFirestore } from "firebase/firestore";
3  import { getAuth } from "firebase/auth";
4  import { getStorage } from "firebase/storage";
5
6  const firebaseApp = initializeApp(firebaseConfig);
7  export const db = getFirestore(firebaseApp);
8  export const auth = getAuth(firebaseApp);
9  export const storage = getStorage(firebaseApp);
10
```

Рис. 3.6 Ініціалізація доступів до всіх сервісів

Для зберігання даних було використано Firebase Firestore – вбудована база даних, яка є NoSQL-документною базою даних. Дані зберігаються у вигляді колекцій та документів, що дозволяє масштабовано зберігати структуровано дані без жорсткої схеми.

Основні колекції

- Users – інформація про користувачів
- Posts – поблікації постів
- Comments – коментарі під постами
- Likes - вподобайки
- Followers/Following – система підписок/підписників
- Tags – список унікальних тегів

Щоб краще зрозуміти, як працює серверна частина, розглянемо конкретний приклад. Уявімо, що користувач створює новий пост. Спочатку він вводить назву, опис, додає зображення та теги. При натисканні на кнопку «Створити», ці дані передаються у вигляді JSON-об’єкта до Firebase Firestore.

Цей об’єкт зберігається у колекції posts. Окремо обробляється завантаження зображення - файл зберігається у Firebase Storage, а в документ додається imageURL, який вказує шлях до цього файлу.

Важливо, що система одразу перевіряє права доступу: чи дійсно ця людина авторизована, чи має право створювати пости. Це реалізовано через Firebase Rules. Якщо користувач не увійшов або спробує створити пост від чужого імені - система не допустить запит.

Після успішного створення поста, інтерфейс React автоматично оновлює стрічку - нова робота з’являється без перезавантаження. Цей ефект досягається завдяки «onSnapshot» з Firestore, який слухає зміни в базі в реальному часі.

Такий підхід не тільки зручний, а й дуже швидкий. Жодного “оновити сторінку”, жодних затримок - усе працює “на льоту”. Користувач одразу бачить результат своєї дії, що дуже мотивує публікувати нові роботи.

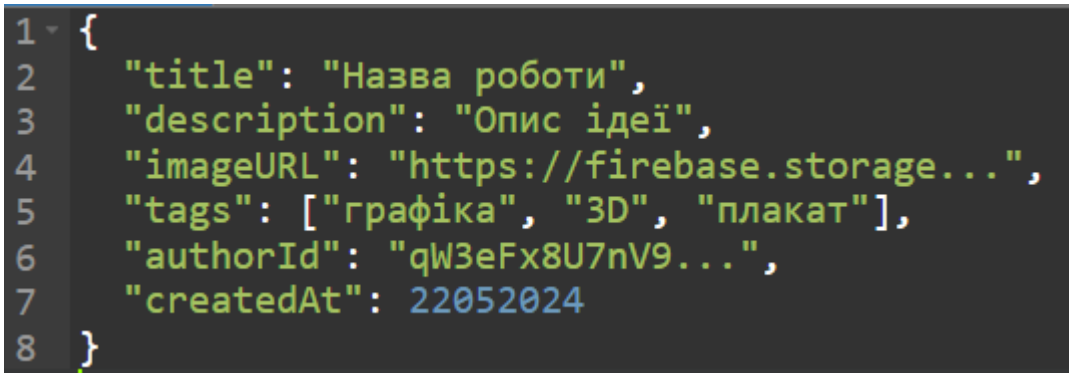


Рис. 3.7 Структура документа Posts

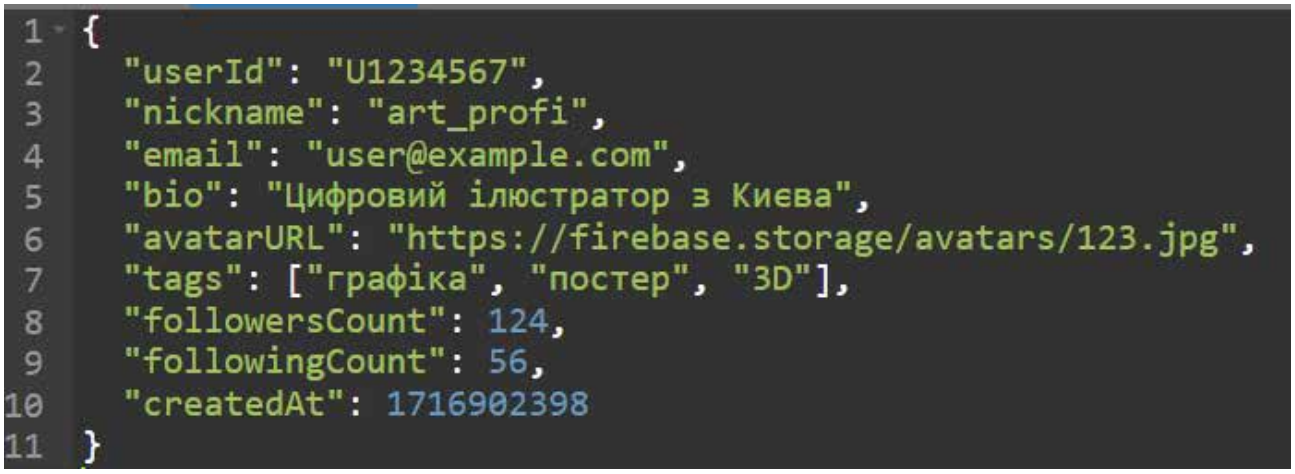


Рис. 3.8 Структура документа users

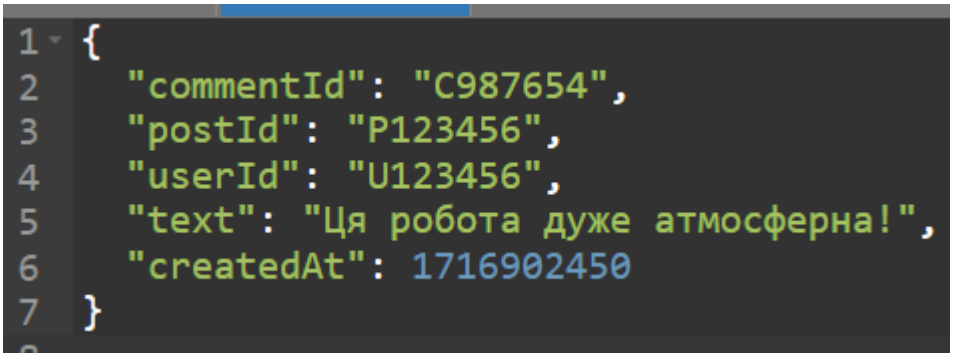


Рис. 3.9 Структура документа Comments

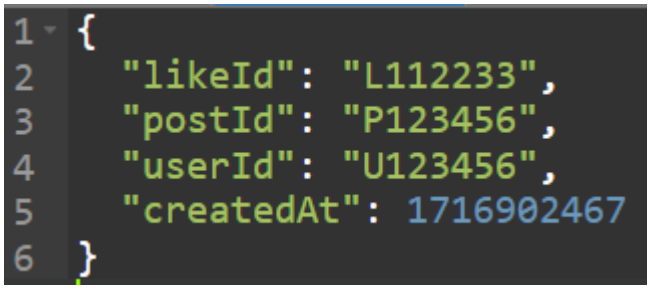


Рис. 3.10 Структура документа Likes

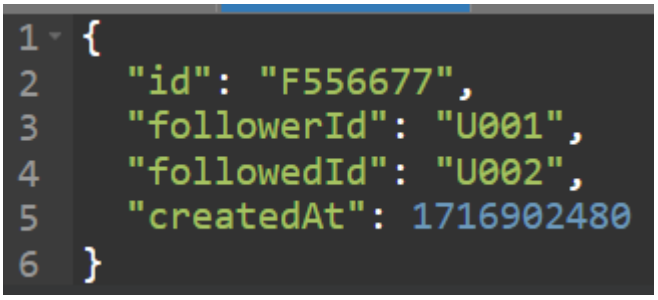


Рис. 3.11 Структура документа Followers/Following

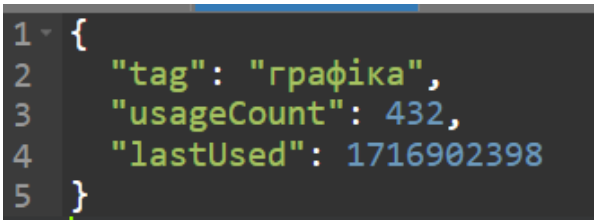


Рис. 3.12 Структура документа Tags

Також ми можем не лише підключити до бази даних та зберігати в ній. У нас є CRUD-операції (Create, Read, Update, Delete) це все виконується через Firebase SDK. Наприклад створення нового поста реалізується так:

```
import { addDoc, collection } from "firebase/firestore";

await addDoc(collection(db, "posts"), {
  title,
  description,
  imageURL,
  tags,
  authorId: currentUser.uid,
  createdAt: Date.now(),
});
```

Рис. 3.13 Приклад створення нового поста

Як говорилось раніше за зберігання зображень відповідає Firebase Storage.

```
const imageRef = ref(storage, `posts/${file.name}`);
await uploadBytes(imageRef, file);
const imageURL = await getDownloadURL(imageRef);
```

Рис. 3.14 Приклад завантаження файлу

За захист та авторизацію відповідає та перевіряє Firebase Authentication. Є правила які визначають рівень доступу користувача.

```
1 match /posts/{postId} {
2   allow read: if true;
3   allow write: if request.auth != null && request.auth.uid == resource.data.authorId;
4 }
```

Рис. 3.15 Визначення рівня доступу користувача

Це забезпечує створення/редагування лише автором, обмежує доступ до чутливої інформації

3.4 Забезпечення зручності, безпеки та роботи з контентом

Щоб користувачу було приємно користуватись платформою BeRaw, а також щоб система працювала швидко, без збоїв і гарантувала захист особистих даних — було реалізовано кілька важливих рішень. Основні акценти зроблені на валідацію введених даних, зручність роботи з медіаконтентом та повний контроль над безпекою і правами доступу до елементів.

Валідація форм та UX-помилки: Форму заповнення акаунта чи створення поста не можна залишити напівпорожньою або заповнити як попало — система не дасть цього зробити. Весь процес контролюється бібліотекою React Hook Form, яка дозволяє відслідковувати кожне поле в реальному часі. Це значить, що якщо користувач забув ввести опис або вказав псевдонім, який уже зайнятий, — система миттєво виведе повідомлення з поясненням, що не так.

Всі помилки видно одразу під полем — вони не ховаються, не пишуться десь у кутку дрібним шрифтом. Наприклад: Якщо пароль занадто короткий або без спецсимволів — повідомлення типу: «Пароль має містити щонайменше 8 символів, цифри і великі букви».Якщо псевдонім уже є — червона підсвітка і напис: «Цей нік уже зайнятий».Якщо теги перевищують ліміт — система видає: «Можна ввести не більше 5 тегів»

Це дуже зручно, бо не треба здогадуватись, чому кнопка «опублікувати» не працює. Людина одразу розуміє, що саме потрібно виправити.Ще один приємний момент — якщо заповнити все правильно, поле змінює колір на зелений, і це додає впевненості, що все зроблено правильно. Весь процес спрямований на те, щоб людина почувалась комфортно, навіть якщо користується платформою вперше.

Завантаження медіафайлів та попередній перегляд. Медіаконтент — це основа платформи. Без зображень — жодного сенсу. Тому робота із

завантаженням файлів повинна бути простою, але чітко контрольованою. Коли користувач додає аватар чи зображення до поста — зразу ж бачить попередній перегляд. Це працює через механізм URL.createObjectURL(), а сам файл при підтвердженні йде у Firebase Storage.

Платформа дозволяє лише безпечні формати:

- .jpg, .jpeg, .png
- Максимальний розмір — до 5 мегабайт

Якщо людина випадково вибрала PDF чи інший тип файлу — побачить повідомлення: «Непідтримуваний формат файлу». І, звісно, система не дасть продовжити публікацію без виправлення. Для збереження порядку файли зберігаються по папках:

- /users/avatars/ — для профільних фото
- /posts/images/ — для постів

Це дає змогу потім легко розділити зображення за типом і зменшити ризик помилок під час обробки. У майбутньому передбачено також впровадження попередньої перевірки на неприйнятний контент за допомогою AI — щоб нецензурні зображення навіть не доходили до етапу публікації.

Контроль доступу та захист даних. Ще один критично важливий момент — це безпека. Ніхто не повинен мати змогу змінити чужий пост, переглянути персональну інформацію або завантажити щось без дозволу. Усе це регулюється через Firebase Security Rules. Система побудована так, що кожен користувач бачить лише те, що має бачити. Наприклад:

- Редагування чи видалення постів можливе лише тим, хто їх створив
- Профілі інших людей можна переглядати, але не змінювати
- Доступ до персональних даних (email, налаштування) закритий

Приклад правила на рівні бази даних:

js

Копировать

Редактировать

```
match /posts/{postId} {
  allow update, delete: if request.auth.uid == resource.data.userID;
}
```

Тобто якщо ID поточного користувача не збігається з ID автора поста — він не може його змінити. Також всі дані передаються по HTTPS-з’єднанню, що гарантує захищене з’єднання між браузером і сервером.

Адаптивність та швидкість: Щоб сторінки швидко завантажувались і не висли навіть на слабких пристроях, використано React.lazy() та Suspense. Це означає, що важкі частини інтерфейсу завантажуються тільки тоді, коли вони реально потрібні. Наприклад, профіль іншого користувача не вантажиться одразу при вході в систему — тільки тоді, коли на нього натискають.

Також платформа повністю адаптивна. Завдяки TailwindCSS елементи автоматично змінюють свої розміри і виглядають красиво і на телефоні, і на великому моніторі. Анімації від Framer Motion додають плавності при переходах, відкритті меню, появі постів. Це створює ефект сучасного, динамічного застосунку, але без перевантаження.

3.5 Адаптація та швидкість роботи

Завдяки TailwindCSS додаток повністю адаптивний — всі сторінки працюють на телефонах, планшетах і великих екранах. Анімації через Framer Motion не навантажують систему, бо бібліотека легка і працює тільки при взаємодії.

Також я використала React.lazy() і Suspense для лінивого завантаження важких компонентів — це дозволяє швидше відкривати сторінки і не гальмує інтерфейс. Наприклад, профіль або пост завантажуються окремо, тільки коли треба.

РОЗДІЛ 4. ВПРОВАДЖЕННЯ ТА ПЕРСПЕКТИВИ РОЗВИТКУ

4.1. Апробація системи

На етапі, що є наступним після створенні функціонала BeRAw та підключення її до сервера, нами було проведено попередній тест кількома учасниками з різних творчих напрямків, від традиційних робіт на папері та з глини до 3D-графіки, дизайнів та ілюстрацій. Підбір піддослідних також враховував їхні навички працювати з ПК, це люди з базовим розумінням комп'ютера і додатків та ті хто мають досить просунутий досвід повноцінної роботи.

Для перевірки системи нами було надіслано посилання на тестову варіацію веб-застосунку та інструкцію з експлуатації платформи. А саме: зареєструватися, створити пост, провзаємодіяти з іншими постами, спробувати фільтрацію постів, створити акаунт, підписатися на інших людей.

І от що ми маємо за результатами дослідження:

90% людей справились до кінця з завданням, 10% - люди які мінімально розуміються на роботі з комп'ютером, не змогли до кінця виконати весь ланцюжок дій.

Наприклад, одна з учасниць тестування — студентка-першокурсниця з художнього напряму — вперше реєструвалась на платформі самостійно. Вона відзначила, що особливо сподобалась логіка платформи: «Мені спочатку було страшно, що не розберуся, але коли я натиснула кнопку зі знаком плюса — все пішло як по кроках. І підказки, і червоні підсвітки одразу пояснювали, що робити. Після публікації я навіть подумала: а чого так просто?»

Тест також показав, що користувачі швидко розуміють принцип фільтрації — і це навіть без жодної інструкції. Люди вводили тематику (наприклад, «анімація»

або «портрет»), і система одразу повертала релевантні пости. Важливо, що навіть ті, хто ніколи не користувався подібними сайтами, змогли самостійно пройти повний шлях: від реєстрації — до створення акаунта — до першої публікації.

У деяких користувачів викликали складнощі поля з тегами (особливо момент обмеження до 5 штук). Тому після тесту я оновила підказку біля цього поля — тепер там чітко написано: «можна ввести не більше 5 тегів, через кому». Після цього повторні користувачі більше не допускали помилок. Це показує, наскільки навіть маленькі уточнення можуть покращити досвід користування.

Середній час створення акаунта – 2-3 хвилини, може збільшуватися в залежності від різних причин: комп'ютера, швидкості інтернету та людського фактору, мається на увазі, що комусь досить легко написати про себе кілька слів та придумати собі псевдонім, іншому ж потрібно трохи часу на прийняття цих рішень.

Оцінку 4.5/5 отримав інтерфейс за зрозумілість, легкість в користуванні, швидкості реагування на запити людини, приємний дизайн та лаконічність форм з об'єктами. Багатьом сподобалась швидкість завантаження сторінок – <5 сек.

4.2. Оцінка ефективності платформи

Щоб сформувати адекватно і об’єктивно бал ми створили три показники: стабільність роботи, швидкість взаємодії, простота в використанні.

Стабільність роботи.

У жодному з тестів не було зафіксовано критичних збоїв. Навіть у ситуації, коли користувачі одночасно завантажували зображення, система працювала без затримок. Наприклад, під час тестування на слабкому ноутбуці з повільним Wi-Fi створення поста відбувалось на тому ж рівні стабільності, що й на сучасному ПК. Це підтверджує, що хмарна інфраструктура Firebase працює не лише швидко, а й стабільно у різних умовах.

Підчас використання платформи система реагувала стабільно, без аварійних вимкнень чи зависання сторінок. Через підключення бібліотек та хмарній архітектурі Firebase наш продукт є комфортним для його використання великій кількості людей.

Швидкість взаємодії

ористувачі помітили, що переходи між сторінками відбуваються без «мерехтіння» або повторного завантаження — це досягається завдяки React Router. Наприклад, після натискання на аватар іншого користувача, сторінка профілю відкривається практично миттєво. А під час коментування поста, після натискання кнопки «надіслати», коментар одразу з’являється в списку — без оновлення сторінки. Це створює ефект “живої системи”, яка не гальмує й не дратує.

Завантаження контенту та даних з Firestore та Storage забезпечує систему швидкістю завантаження та плавність в роботі. Завдяки стабільності технічної

частини система коректно та швидко реагувала на запити людини, зберігаючи або оновлюючи дані сторінки чи постів.

Простота в використанні.

Всі учасники тесту відзначили, що платформа схожа за логікою до мобільного додатку — все на своїх місцях, нічого зайвого, а головне — нічого не ламається, навіть якщо натискати “не туди”. Один із респондентів сказав: “Я просто дивився, що можна натиснути, і воно або спрацьовувало, або пояснювало, чому ні. Це дуже приємно, бо в багатьох сайтах треба спочатку почитати інструкцію”. Такі враження підтверджують, що акцент на UX був обраний правильно.

Платформа інтуїтивно зрозуміла, стандартні символи, об’єкти та поля вводу/виводу створювались на основі аналізу багатьох подібних додатків, звідки і перейняло лиш позитивні моменти для нашого веб-застосунку, та були на звичних місцях. Це забезпечило користувачам не зациклюватись на інтерфейсі та його архітектурі. Також кнопки та поля для вводу були підписані коротко та достатньо ясно, щоб користувач зміг просто ввести дані та «рухатись» далі по системі.

4.3 Взаємодія з користувачами та підготовка до майбутнього розвитку

Після запуску пробної версії BeRaw платформа одразу отримала зворотний зв’язок від користувачів. Це були як досвідчені ілюстратори й дизайнери, так і ті, хто просто хотів викласти свої малюнки чи колажі. Більшість добре впоралась із завданнями, проте деякі ситуації допомогли краще зрозуміти, що саме ще можна покращити.

Наприклад, частина користувачів не одразу зрозуміла, як працює система тегів. Тому біля поля вводу тегів було додано просте пояснення своїми словами, щоб усі могли швидко зорієнтуватись. Кнопка для створення поста виявилася не зовсім помітною, тож я додала м’яку анімацію при наведенні, щоб вона «оживала» в потрібний момент. Також довелось вдосконалити адаптацію для мобільних пристроїв — на дуже малих екранах деякі елементи перекривались або відображались некоректно. Всі ці дрібниці були швидко усунені, що зробило платформу ще зручнішою.

Окрім цього, були продумані різні ситуації, які можуть виникати під час користування. Якщо, наприклад, користувач випадково втрачає інтернет, система не дозволяє виконувати жодних дій і виводить повідомлення про втрату з’єднання. При натисканні на кнопку «видалити пост» обов’язково з’являється підтвердження, щоб уникнути випадкового видалення. Якщо ж користувач намагається відкрити розділ, не увійшовши в систему, він одразу перенаправляється на сторінку входу.

Технічна частина теж була продумана до деталей. Кожен користувач бачить лише свою інформацію і не може редагувати чужі пости чи профілі. Це реалізовано завдяки спеціальним правилам доступу в Firebase. Платформа не падає і не гальмує навіть тоді, коли нею користується багато людей одночасно

— це забезпечується хмарною інфраструктурою, яка підлаштована під навантаження.

Платформа не просто була протестована — вона отримала життя. Багато хто з учасників тестування почали активно ділитись роботами, навіть без запрошення чи додаткової мотивації. Коли люди самі пишуть: «А буде нічна тема?», «А можна більше 5 тегів?» — це ознака того, що продукт працює, він їм цікавий, і вони хочуть залишитися.

Саме ці реакції стали головним показником того, що BeRaw — це не черговий шаблонний сайт, а система, яку хочеться використовувати. Один із користувачів надіслав скриншот своєї стрічки, підписавши: «Це перший раз, коли мої роботи не загубились серед реклами». Подібні повідомлення стали підтвердженням того, що шлях, яким розвивається платформа — правильний.

Також було цікаво спостерігати, як різні користувачі взаємодіють з фільтром тегів. Деякі спеціально підбирали нові хештеги, щоб протестувати, чи буде знаходити їхній пост. І після того, як це спрацьовувало, люди залишали позитивні емоції у відгуках. Це показало, що навіть найменші функції можуть мати великий вплив на залученість користувачів.

Після перших тестів платформи я вже визначила напрямки, в яких BeRaw буде розвиватися надалі. Одним із головних кроків стане впровадження штучного інтелекту для автоматичної перевірки зображень — це допоможе захищати платформу від забороненого або скопійованого контенту. Також планується створення нічного режиму для тих, хто працює в темний час доби. Щоб користувачам було зручніше поширювати свої роботи, з’явиться функція імпорту зображень з інших платформ — таких як Behance або Instagram.

Окрім того, буде реалізована система внутрішніх повідомлень, щоб користувачі могли спілкуватися безпосередньо один з одним, обговорюючи свої роботи або обмінюючись порадами. І ще однією важливою функцією стане віджет портфоліо — можливість створити короткий блок із роботами, який можна вставити у власний сайт або електронне резюме.

Усі ці зміни покликані зробити платформу ще ближчою до її користувачів — відкритою, живою та такою, що підтримує кожного творця.

ВИСНОВКИ

У ході виконання цієї бакалаврської кваліфікаційної роботи було пройдено повний і послідовний шлях — від задуму створення сучасної веб-платформи до її повноцінного втілення у вигляді інтерактивного програмного продукту. Основною метою проєкту було створення інформаційної системи, яка дозволяє користувачам ділитися своїми творчими роботами у зручному, безпечному та зрозумілому середовищі. Ця мета була реалізована через розробку та запуск платформи BeRaw.

На етапі підготовки та аналізу було досліджено ринок існуючих рішень, зокрема платформ Behance, Dribbble та Pinterest. Було виявлено низку суттєвих обмежень, таких як складність інтерфейсу, надмірне орієнтування на популярність чи алгоритми, а також відсутність гнучкості для новачків. Ці недоліки стали поштовхом до створення власного бачення системи, яка поєднує простоту використання з повноцінною функціональністю.

Розробка платформи BeRaw здійснювалась з використанням сучасних веб-технологій. React забезпечив гнучку побудову інтерфейсу користувача, а також швидку та реактивну взаємодію між компонентами без перезавантаження сторінок. Firebase надав засоби для зберігання інформації, автентифікації користувачів та організації хмарної бази даних. Стилізація платформи була виконана за допомогою TailwindCSS, що дозволило зробити інтерфейс чистим, лаконічним і адаптивним до різних типів пристроїв. Для додавання плавних анімацій застосовано Framer Motion, що підсилило візуальне сприйняття інтерфейсу та зробило платформу більш “живою”.

Особливу увагу було приділено деталям взаємодії з користувачем. Було реалізовано систему підказок, перевірку полів введення, обмеження на кількість тегів, динамічну валідацію паролів, миттєве відображення попереднього

перегляду зображень, захист даних через правила доступу, а також систему авторизації через пошту або Google-акаунт. Усі ці механізми не тільки спрощують роботу, а й створюють відчуття довіри до платформи.

Після реалізації функціоналу була проведена апробація системи на реальних користувачах. У тестуванні взяли участь представники різних творчих напрямів: художники, графіки, студенти дизайну, а також користувачі без технічного досвіду. Більшість з них самостійно пройшли увесь маршрут: від реєстрації до публікації власного поста. Було відзначено зручність навігації, швидке завантаження сторінок, інтуїтивну логіку побудови інтерфейсу, а також загальну естетичність візуального середовища. Позитивні відгуки підтвердили ефективність обраних рішень. За результатами тестів було внесено додаткові покращення — зокрема, уточнено підказки до полів, адаптовано компоненти для мобільних пристроїв та оптимізовано роботу фільтрації.

Окремо було досліджено потенціал платформи до подальшого розвитку. Запити користувачів на нові функції — такі як нічний режим, розширення системи тегів, можливість комунікації між авторами, імпорт зображень з інших платформ — свідчать про інтерес до BeRaw як до живої системи. Платформа сприймається не як тимчасовий проєкт, а як основа для подальшого зростання.

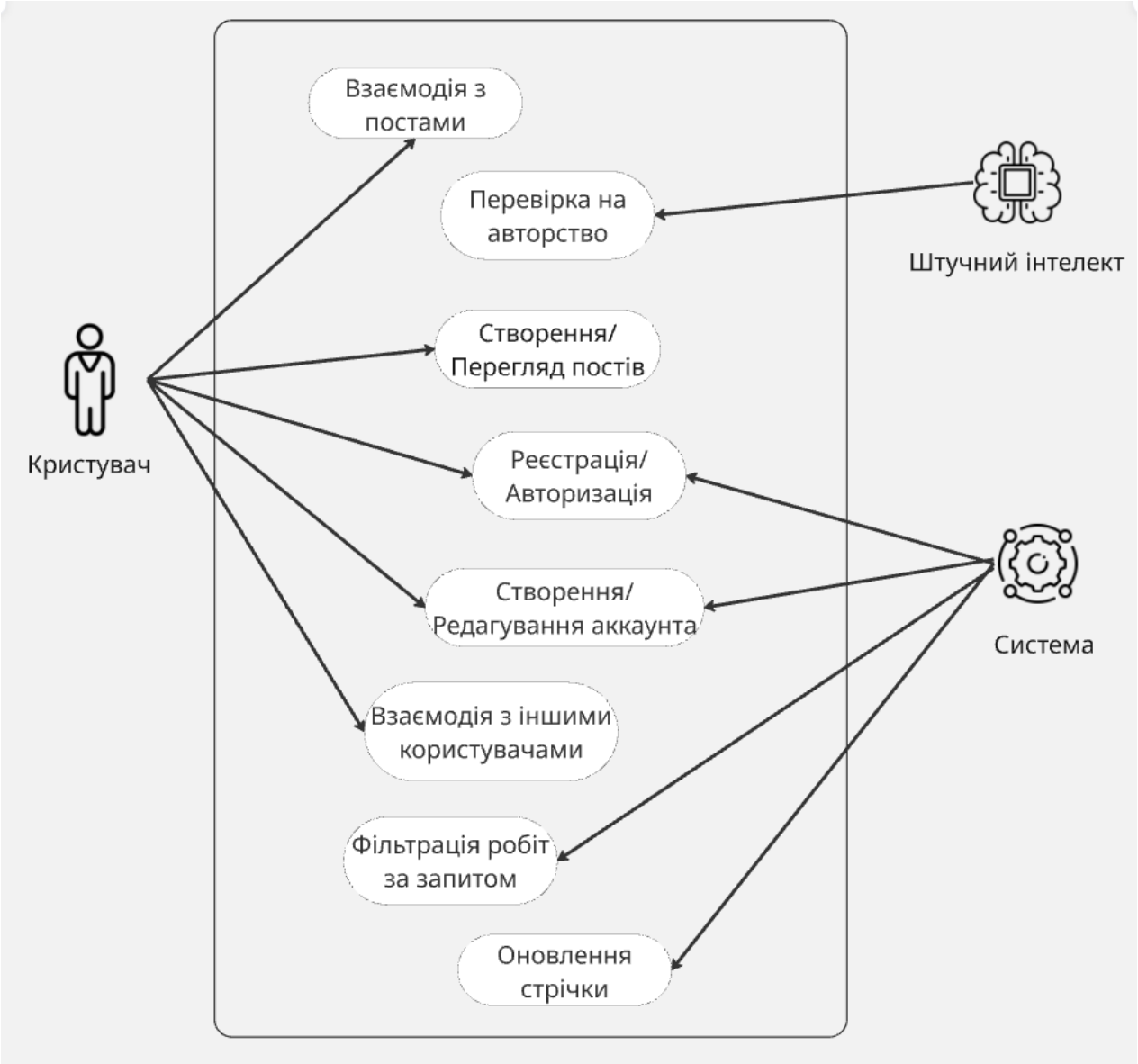
Таким чином, дипломна робота охопила повний цикл — від постановки задачі до перевірки готового продукту в реальних умовах. Усі цілі були досягнуті: створено стабільний, зручний і сучасний веб-застосунок для креативної спільноти. BeRaw вже зараз демонструє свою життєздатність і потенціал до масштабування. Вона забезпечує чесні умови для демонстрації творчості, не ставлячи користувачів у нерівні умови. Платформа створена з урахуванням потреб людей і з повагою до контенту, який вони створюють. Саме в цьому полягає її головна цінність.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

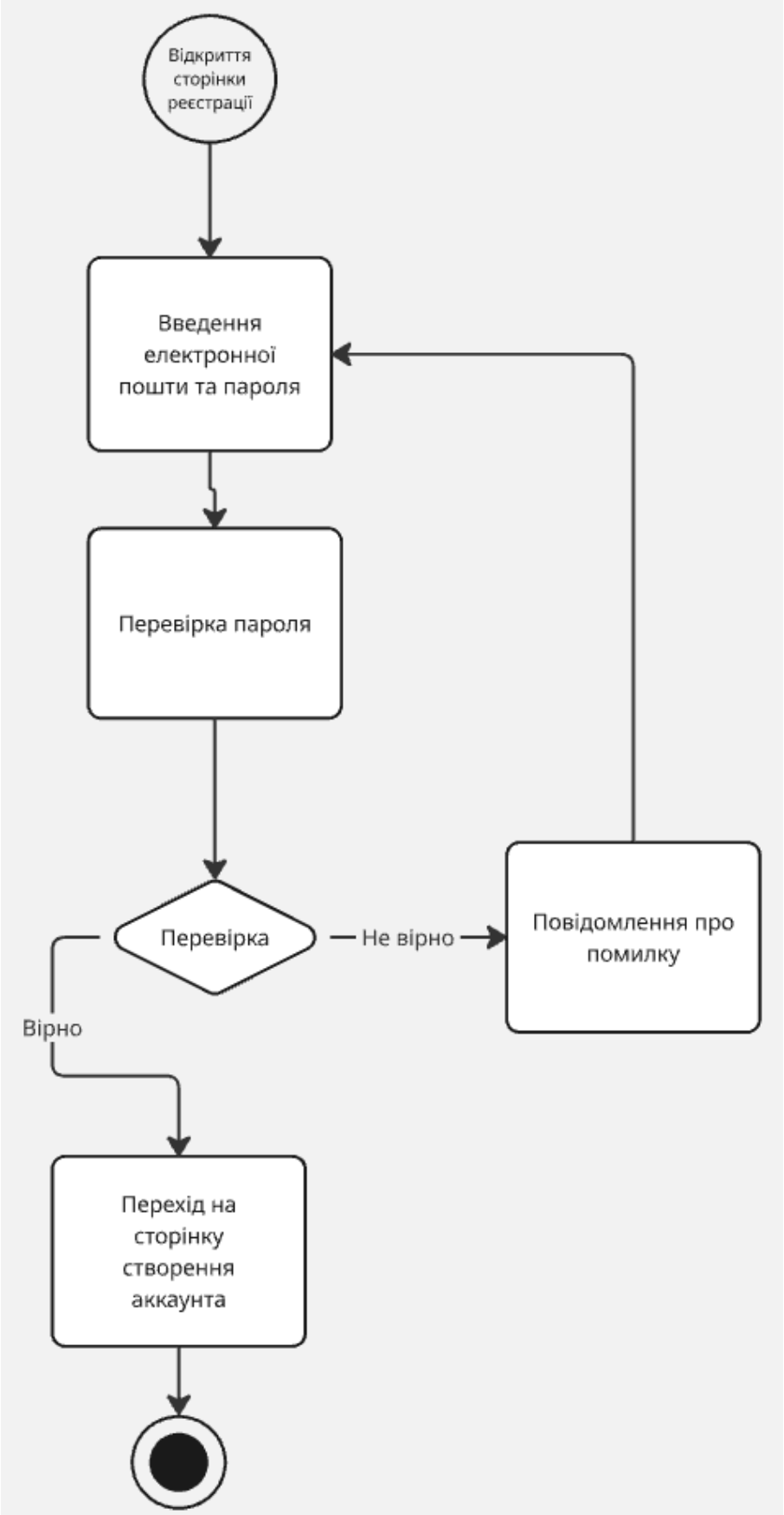
1. Бутенко, Н. В. (2022). Використання технологій хмарних обчислень у розробці веб-застосунків. Інформаційні технології і засоби навчання, 90(2), 15–28.
2. Чаплинський, М. О. (2021). Проєктування користувацького інтерфейсу з використанням бібліотеки React. Наукові записки НУ "Львівська політехніка", 15(3), 64–72.
3. Олійник, І. Ю. (2020). Використання Firebase у мобільній та веб-розробці. Комп’ютерні науки та інформаційні технології, 18(1), 77–85.
4. Дьяків, Р. В. (2021). Інтерактивна візуалізація в UI/UX-дизайні: Framer Motion і TailwindCSS у комплексі. Інформаційні технології в дизайні, 3(4), 41–47.
5. Нгуєн, Д. Х., & Пархоменко, О. В. (2023). Хмарні бази даних у веб-застосунках: аналіз переваг NoSQL Firestore. Сучасні інформаційні системи, 26(2), 51–60.
6. Google. (2023). Firebase Documentation. Офіційна документація. Отримано з https://firebase.google.com/docs
7. Meta. (2023). React Documentation. Офіційна документація. Отримано з https://react.dev
8. Tailwind Labs. (2023). TailwindCSS Documentation. Отримано з https://tailwindcss.com/docs
9. Framer. (2023). Framer Motion Documentation. Отримано з https://www.framer.com/motion

10. Adobe Systems. (2022). Behance Platform Overview. Отримано з https://www.behance.net
11. Dribbble Holdings Ltd. (2022). Dribbble Community Guidelines. Отримано з https://dribbble.com
12. Pinterest Inc. (2022). Pinterest Product Features. Отримано з https://www.pinterest.com
13. W3C. (2023). Content Security and HTTPS Protocols in Web Development. Отримано з https://www.w3.org
14. OpenAI. (2024). Technical Integration of AI in Web Platforms. Внутрішня наукова робота.

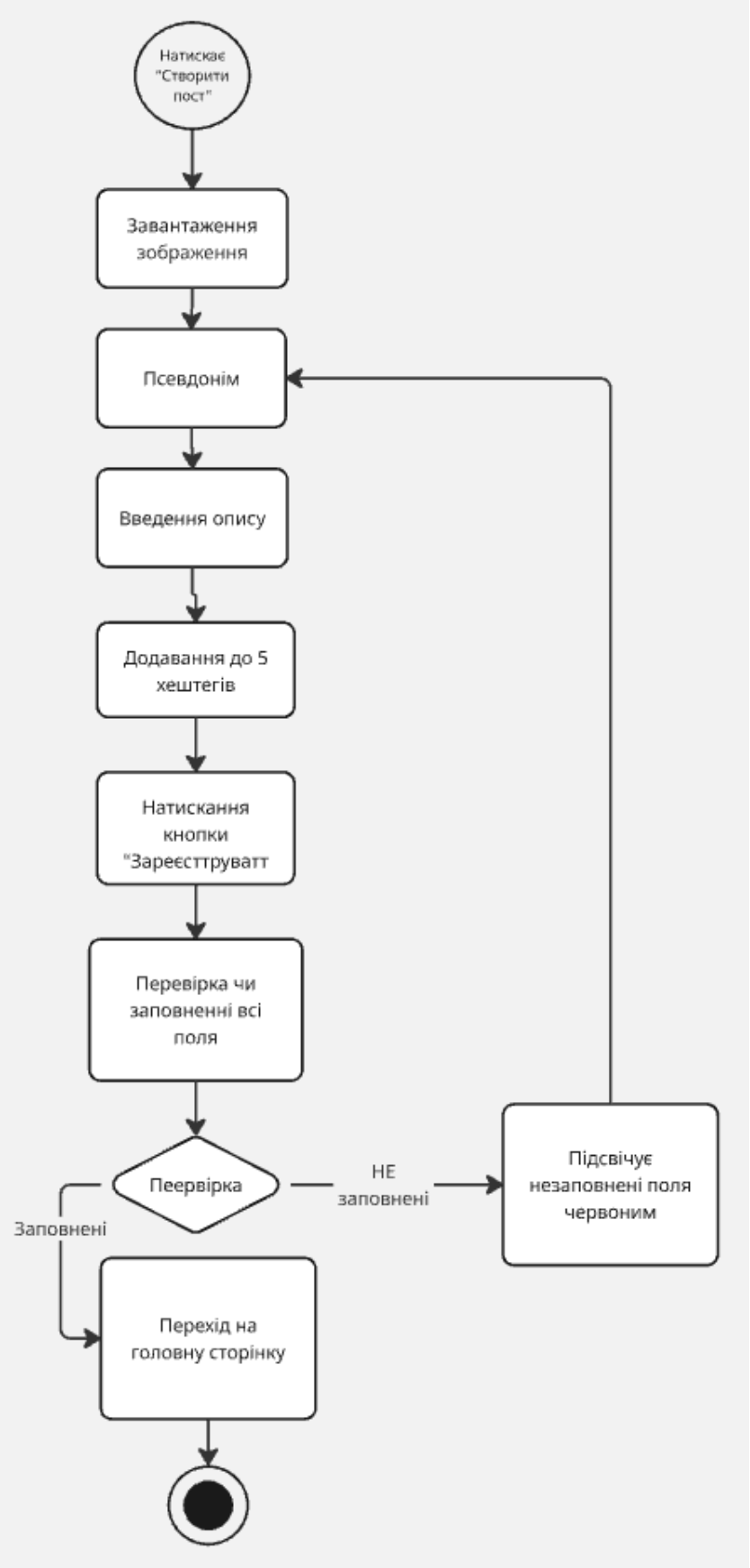
ДОДАТОК А



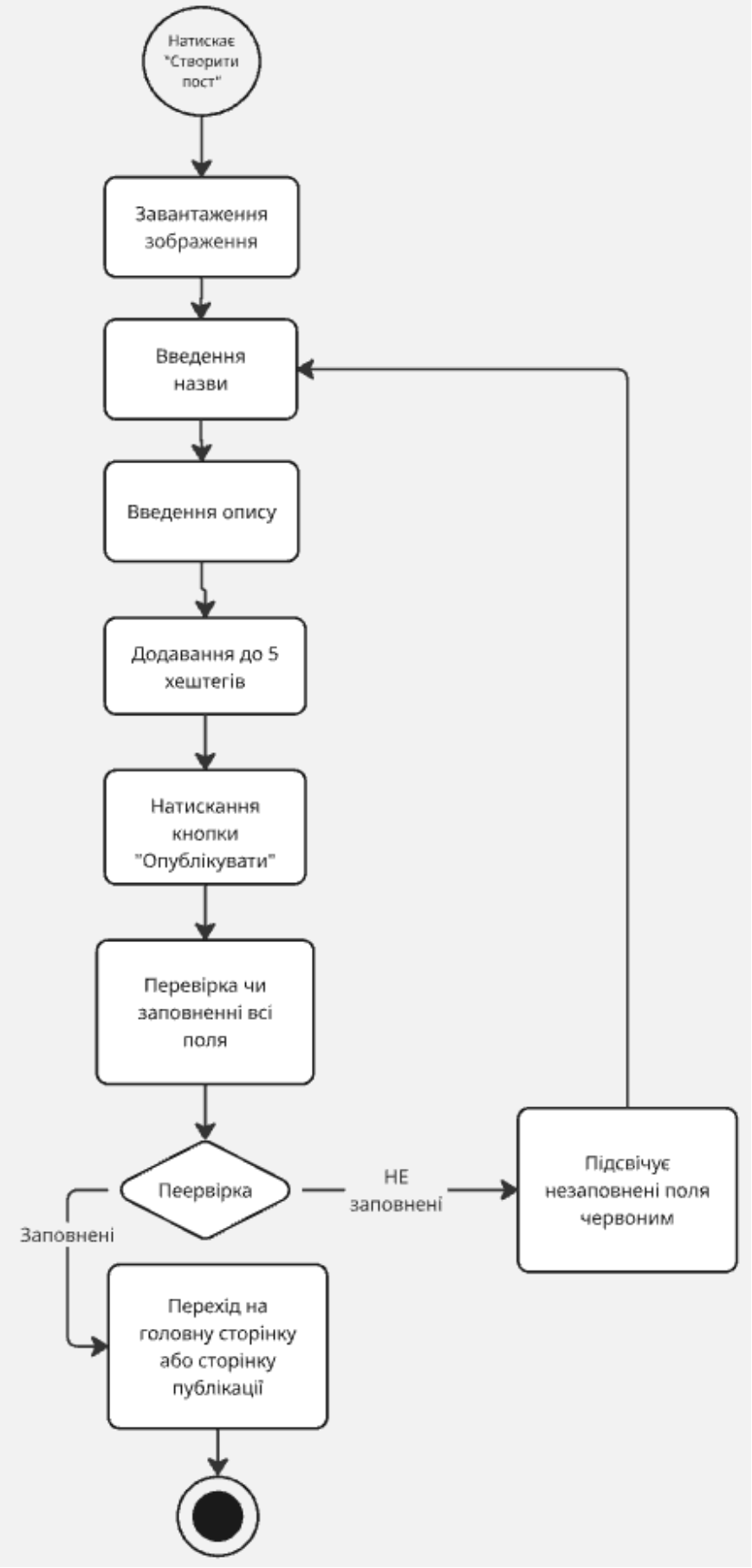
Use case -діаграма



Діаграма діяльності «Реєстрааація»



Діаграма діяльності «Створення аккаунта»

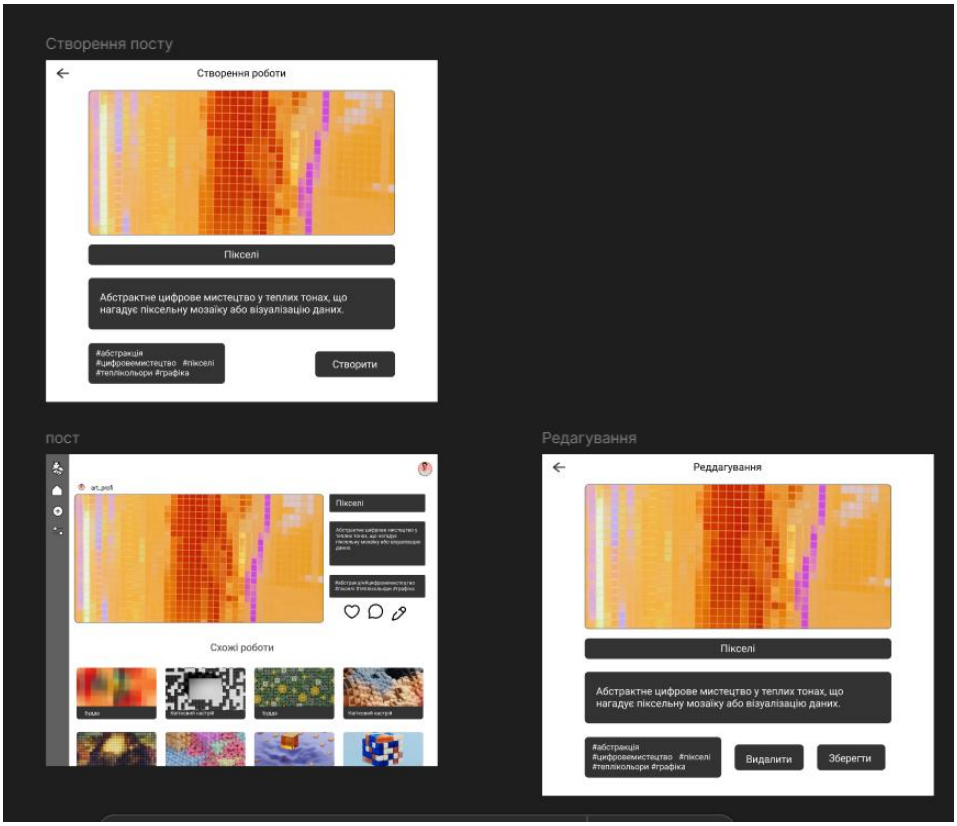

Натискає "Створити пост"
Завантаження зображення
Введення назви
Введення опису
Додавання до 5 хештегів
Натискання кнопки "Опублікувати"
Перевірка чи заповненні всі поля
Пеервірка
НЕ заповнені
Підсвічує незаповнені поля червоним
Заповнені
Перехід на головну сторінку або сторінку публікації

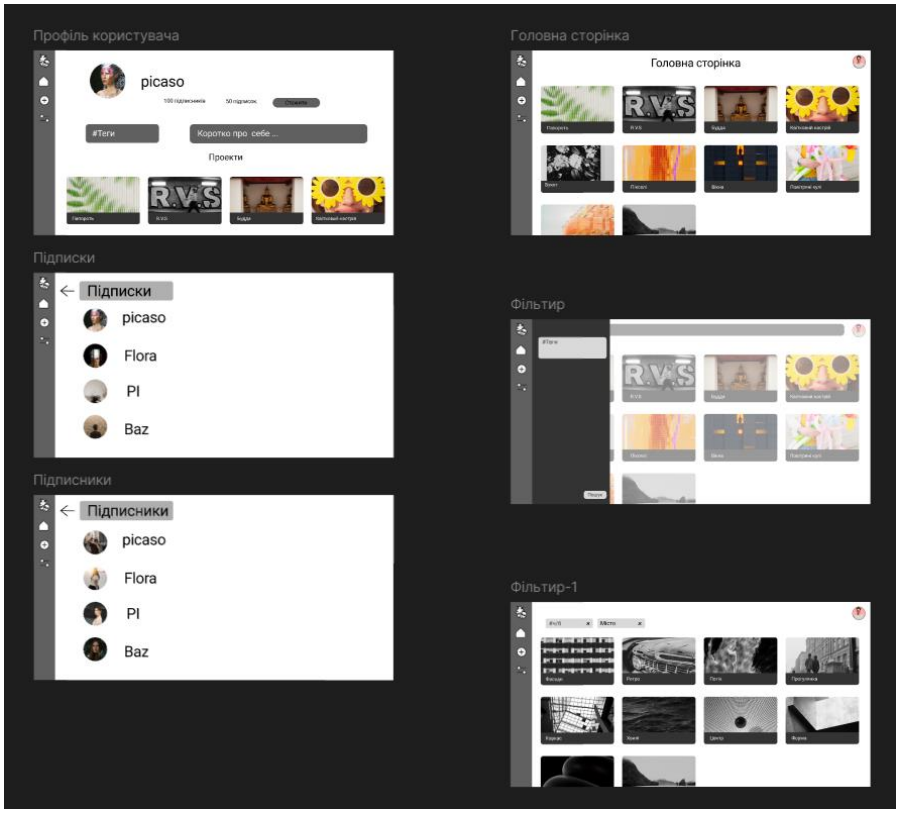
Діаграма діяльності «Створення поста»

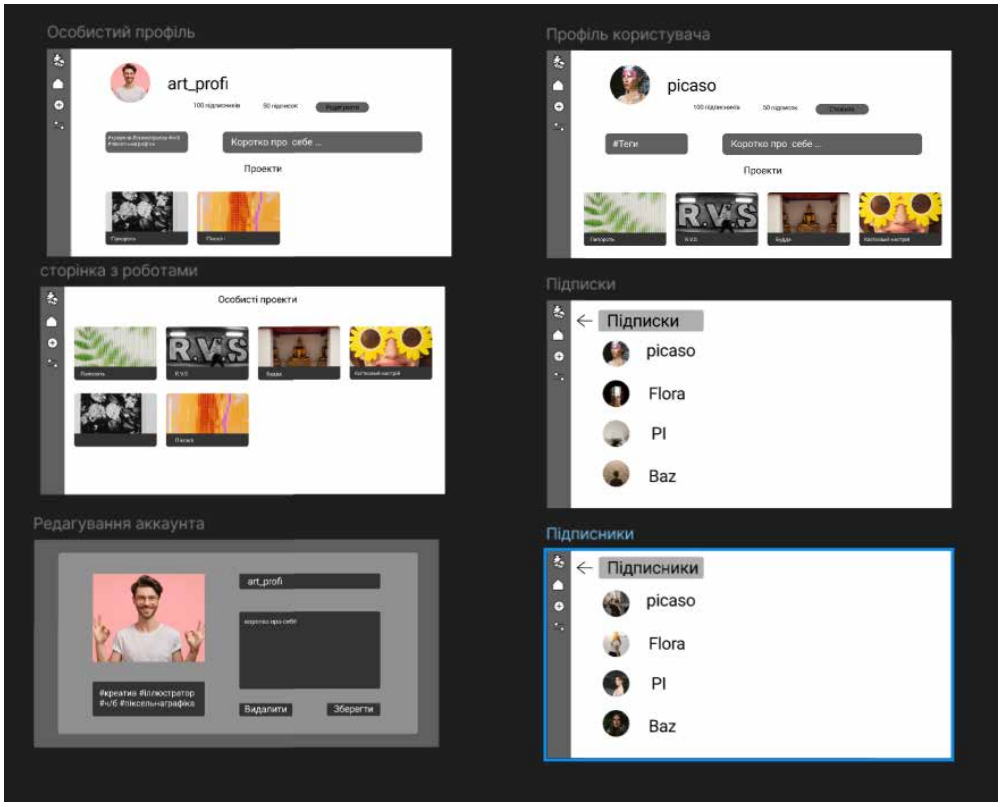

Зайти на сторінку підписки/підписники
Пошук нам потрібного користувача
Перехід на сторінку
перегляд портфоліо
Підписаний?
Ні
Так
Підписатися
Повернення на головну

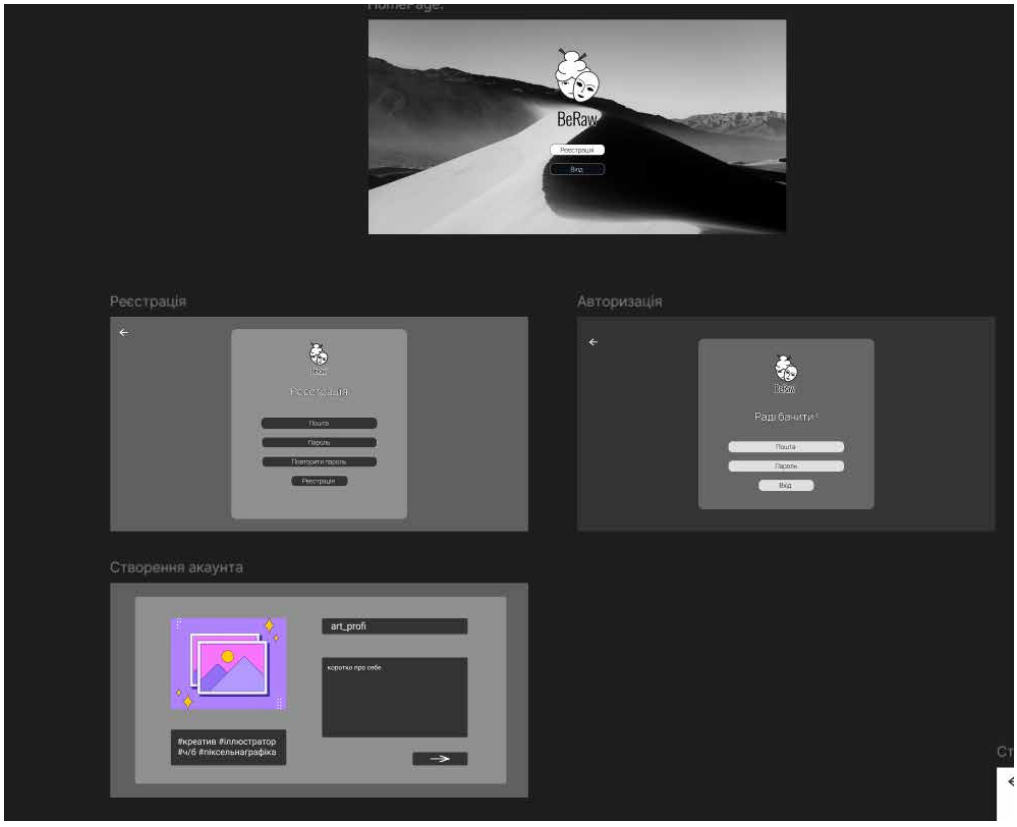
Діаграма діяльності «Пошук користувача»

Діаграма діяльності «Взаємодія з постом»






Особистий профіль
art_profi
Коротко про себе ...
Проекти
Профіль користувача
picaso
#Теги
Коротко про себе ...
Проекти
сторінка з роботами
Особисті проекти
Підписки
Підписки
picaso
Flora
PI
Baz
Редагування аккаунта
art_profi
Видалити
Зберегти
Підписники
Підписники
picaso
Flora
PI
Baz


BeRaw
Реєстрація
Авторизація
Створення акаунта
art_profi

ДОДАТОК Б

Головна сторінка

```
import React, { useState } from "react";
import "../styles/MainPage.css";
import FilterSidebar from "../components/FilterSidebar";

import logo from "../assets/logo.png";
import img1 from "../assets/1.png";
import img2 from "../assets/2.png";
import img3 from "../assets/3.png";
import img4 from "../assets/4.png";
import img5 from "../assets/5.png";
import img6 from "../assets/6.png";
import img7 from "../assets/7.png";
import img8 from "../assets/8.png";

const posts = [
  { id: 1, title: "Папороть", image: img1 },
  { id: 2, title: "R.V.S", image: img2 },
  { id: 3, title: "Будда", image: img3 },
  { id: 4, title: "Квітковий настрій", image: img4 },
  { id: 5, title: "Букет", image: img5 },
  { id: 6, title: "Пікселі", image: img6 },
  { id: 7, title: "Вікна", image: img7 },
  { id: 8, title: "Повітряні кулі", image: img8 },
];

function MainPage() {
```

```
const [isFilterOpen, setIsFilterOpen] = useState(false);
const [tagInput, setTagInput] = useState("");

return (
  <div className="main-container">
    <FilterSidebar
      isOpen={isFilterOpen}
      onClose={() => setIsFilterOpen(false)}
      onSearch={() => console.log("Шукаємо:", tagInput)}
      tagInput={tagInput}
      setTagInput={setTagInput}
    />

    <aside className="sidebar">
      <img src={logo} alt="Logo" className="sidebar-logo" />

      {/* Домік */}
      <svg className="sidebar-icon" viewBox="0 0 24 24" fill="white">
        <path d="M10 20v-6h4v6h5v-8h3L12 3 2 12h3v8z" />
      </svg>

      {/* Плюс */}
      <svg className="sidebar-icon" viewBox="0 0 24 24" fill="white">
        <path d="M19 11H13V5h-2v6H5v2h6v6h2v-6h6z" />
      </svg>

      {/* Фільтр */}
      <svg
        className="sidebar-icon"
```

```
          viewBox="0 0 24 24"
          fill="white"
          onClick={() => setIsFilterOpen(true)}
          style={{ cursor: "pointer" }}
        >
          <path d="M10 18h4v-2h-4v2zm-7-8v2h18v-2H3zm3-6v2h12V4H6z" />
        </svg>
      </aside>

      <main className="content">
        <header className="main-header">
          <h1>Головна сторінка</h1>
        </header>

        <div className="posts-grid">
          {posts.map((post) => (
            <div className="post-card" key={post.id}>
              <img src={post.image} alt={post.title} className="post-image" />
              <div className="post-title">{post.title}</div>
            </div>
          ))}
        </div>
      </main>
    </div>
  );
}

export default MainPage;
```

Сторінка з реєстрацією

```
import React, { useState } from "react";
import "../styles/Register.css";
import logo from "../assets/logo.png";
import { useNavigate } from "react-router-dom";

function Register() {
  const navigate = useNavigate();
  const [password, setPassword] = useState("");
  const [confirmPassword, setConfirmPassword] = useState("");

  const handlePasswordChange = (e) => {
    const value = e.target.value;
    if (/^[a-zA-Z0-9]*$/.test(value) && value.length <= 6) {
      setPassword(value);
    }
  };

  const handleConfirmPasswordChange = (e) => {
    const value = e.target.value;
    if (/^[a-zA-Z0-9]*$/.test(value) && value.length <= 6) {
      setConfirmPassword(value);
    }
  };

  return (
    <div className="register-wrapper">
```

```
<div className="register-card">
  <img src={logo} alt="BeRaw Logo" className="register-logo" />
  <h1 className="register-title">BeRaw</h1>
  <h2 className="register-subtitle">Реєстрація</h2>

  <input type="email" placeholder="Пошта" className="register-input" />

  <input
    type="password"
    placeholder="Пароль"
    className="register-input"
    value={password}
    onChange={handlePasswordChange}
  />

  <input
    type="password"
    placeholder="Повторити пароль"
    className="register-input"
    value={confirmPassword}
    onChange={handleConfirmPasswordChange}
  />

  <button
    className="register-button"
    onClick={() => navigate("/account-create")}
  >
    Реєстрація
  </button>
```

```
        <button onClick={() => navigate("/")} className="register-back">
          ←
        </button>
      </div>
    </div>
  );
}

export default Register;
```