МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБЗАСТОСУНКУ СОЦІАЛЬНОЇ МЕРЕЖІ ЗАСОБАМИ DJANGO»**

Виконав: студент 4 курсу, групи 6.1219-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
Р.В. Заботін
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент професор кафедри комп'ютерних наук, доцент, д.т.н. Шило Г.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ 07 ” 02 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Заботіну Роману Владиславовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебзастосунку соціальної мережі засобами Django

керівник роботи Кудін Олексій Володимирович, к.ф.-м.н, доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 26 » січня 2023 року № 102-с

2. Строк подання студентом роботи 07.06.2023 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі, аналіз предметної області.
2. Проєктування.
3. Реалізація та тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 07.02.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.02.2023	
2.	Збір вихідних даних.	22.02.2023	
3.	Обробка методичних та теоретичних	20.03.2023	
	джерел.		
4.	Розробка першого та другого розділу.	14.04.2023	
5.	Розробка третього розділу.	15.05.2023	
6.	Оформлення та нормоконтроль	01.06.2023	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	22.06.2023	

Студент _____ Р.В. Заботін
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Кудін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка вебзастосунку соціальної мережі засобами Django»: 63 с., 26 рис., 10 джерел, 4 додатків.

DJANGO, FRAMEWORK, MVC, PYTHON, UML.

Об’єкт дослідження – система, інструменти та архітектура Django, засоби взаємодії системи з користувачем.

Мета роботи – розробити вебсайт соціальної мережі.

Методи дослідження – моделювання, проєктування, програмний, аналітичний.

Соціальні мережі стали невід’ємною частиною сучасного світу. З кожним роком вони залучають все більше користувачів, надаючи їм можливість спілкування, обміну інформацією та знайомств. З цим розширенням соціального впливу мережі також зростає актуальність розробки вебсайтів соціальних мереж.

Однією з найважливіших причин розробки вебсайтів соціальних мереж є зростання популярності цих платформ серед користувачів. Люди використовують соціальні мережі для спілкування з родиною та друзями, знаходження нових контактів, отримання новин та інформації про події. Розробка вебсайту соціальної мережі дозволяє створити простір, де користувачі можуть зручно взаємодіяти між собою та змінювати світ навколо себе.

Таким чином, за результатами роботи створено зручну та ефективну систему обліку навчального навантаження з використанням Django.

SUMMARY

Bachelor’s qualifying paper «Development of a Social Network Web Application using Django»: 63 pages, 26 figures, 10 references, 4 supplements.

DJANGO, FRAMEWORK, MVC, PYTHON, UML.

The object of the system, tools, and architecture of Django, means of system interaction with the user.

The aim of the study is develop a social networking website.

The methods of research are modeling, design, programming, analytical.

Social networks have become an integral part of the modern world. Each year, they attract more users, providing them with opportunities for communication, information exchange, and making new connections. With the expanding social influence of these networks, the relevance of developing social networking websites also increases.

One of the most important reasons for developing social networking websites is the growing popularity of these platforms among users. People use social networks to communicate with family and friends, find new contacts, and obtain news and information about events. The development of a social networking website allows for creating a space where users can conveniently interact with each other and make a difference in the world around them.

As a result of the work, a convenient and efficient educational workload management system using Django has been created.

ЗМІСТ

ВСТУП

Соціальні мережі є неодмінною частиною сучасного світу. Вони стали платформою, на якій люди спілкуються, обмінюються інформацією, діляться своїми думками, фотографіями та відео. Завдяки розробці Інтернету, мобільним технологіям та швидкому доступу до мережі, соціальні мережі стають ще більш популярними і важливими для людей.

Вебсайти соціальних мереж допомагають людям спілкуватися та взаємодіяти один з одним. Вони створюють можливість побудувати мережу знайомств, знайти старих друзів, обмінюватися повідомленнями та досліджувати інтереси разом з іншими людьми. Вебсайт соціальної мережі забезпечує зручність і широкий спектр функцій для комунікації.

Розробка вебсайту соціальної мережі вимагає високих технічних знань та вмінь. Створення зручного та безпечного інтерфейсу, адаптованого для різних пристроїв, і забезпечення ефективної системи безпеки є важливими завданнями розробників. Крім того, розробка вебсайту повинна враховувати потреби користувачів, їхні вподобання та вимоги до функціональності.

Виходячи з цього, було вирішено створити систему, котра би мала зрозумілий інтерфейс та надавала необхідний функціонал користувачам.

Актуальність дослідження: актуальність теми зумовлена необхідністю створення інноваційного та конкурентоспроможного продукту для спілкування, комунікації, розваг, розвитку бізнесу та підтримки відкритого інформаційного простору.

З огляду на це, можна виділити наступні цілі і задачі нашого дослідження:

Мета: розробити вебсайт соціальної мережі засобами Django.

Задачі:

1) сформувати вимоги до системи;
2) спроєктувати та побудувати архітектуру системи;
3) реалізувати вебсайт соціальної мережі;

4) протестувати роботу системи.

Об’єкт дослідження: процес комунікації засобами соціальних мереж, інструменти для реалізації вебсайту соціальної мережі, функціонал системи, необхідний користувачам.

Предмет дослідження: створення вебсайту, що дозволяє користувачам комунікувати засобами соціальних мереж.

Методи дослідження: моделювання, проєктування, програмний, аналітичний.

Перший розділ присвячено збору та аналізуванню вимог до системи, огляду інструментів розробки та опису системи.

У другому розділі розглянуто етапи проєктування системи, наведено детальний опис прецедентів.

Третій розділ присвячено реалізації та тестуванню роботи системи, наведено детальне керівництво користувача, яке описує процес роботи з вебсайтом соціальної мережі.

1 ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Терміни та визначення

1.1.1 Загальні терміни

Система – вебдодаток створений на основі Django.

Django – це відкрите програмне забезпечення, яке надає розробникам веб-додатків повний набір інструментів для швидкої розробки безпечних та масштабованих веб-додатків.

SQLite – це легковага вбудована реляційна база даних, яка зберігає дані у вигляді файлів на диску.

ДВІ – Діаграма Варіантів Використання чи Use Case Diagram.

ДД – Діаграма Діяльності.

ДП – Діаграма Послідовності.

Користувач – людина, котра зареєстрована у системі та має певні права доступу.

1.1.2 Технічні терміни

ІС – інформаційна система.

БД – база даних, місце збереження інформації ІС.

MVC – архітектурний шаблон, який використовується під час проєктування та розробки програмного забезпечення.

1.2 Функціональні вимоги

1.2.1 Призначення і цілі створення системи

Функціональне призначення системи – реалізувати функціональні можливості соціальної мережі.

Експлуатаційне призначення системи: система може експлуатуватися користувачем системи.

Мета створення системи – розробка вебзастосунку соціальної мережі.

1.2.2 Загальні функціональні можливості системи

Система має надавати користувачам такі можливості:

- створення посту;
- реєстрація в системі;
- редагування профіля;
- підписка на користувача;
- відписка від користувача;
- поставити позначку «Подобається» під постом;
- зняти позначку «Подобається» під постом;
- перегляд профіля іншого користувача;
- пошук користувачів;
- вхід до системи;
- вихід з системи.

1.3 Нефункціональні вимоги

1.3.1 Інтерфейс користувача

Система повинна відображати коректно інтерфейс користувача на будь-якому пристрої.

1.3.2 Підтримка браузерів

Система повинна працювати для наступних браузерів останніх версій:

- Mozilla Firefox;
- Google Chrome;
- Safari;
- Opera.

1.3.3 Вимоги до продуктивності

Система повинна відображати будь-яку сторінку не довше, ніж за 2 секунди.

Система повинна відправляти повідомлення не довше, ніж 2 секунди.

1.3.4 Вимоги до безпеки

Система не повинна дозволяти вводити у поля дані, які можуть бути використані, як експлойти.

Система не повинна надавати доступ неавторизованим користувачам доступ до даних системи.

1.4 Опис предметної області

Предметною областю є розробка вебзастосунку соціальної мережі. Даний застосунок повинен надавати користувачам можливість створити власний обліковий запис та взаємодіяти з іншими користувачами. Процес взаємодії з системою проходить наступним чином. Користувач, повинен ввести адресу сайту в браузері та ввести дані в форму входу до системи. Якщо користувач вперше на сайті, він повинен спочатку зареєструватися.

Після входу до системи користувач має можливість взаємодіяти з такими розділами як: пошук користувачів, перегляд профілів користувачів, редагування власного профіля, створення нового поста.

Процес створення нового поста проходить наступним чином. Користувач натискає на кнопку «Створити новий пост», система відображає інтерфейс створення нового поста. Користувач заповнює форму, яка складається з 2-х полів: фото, підпис фото. Після введення даних користувач натискає кнопку «Зберегти», система зберігає введені дані, пост з’являється в ленті інших користувачів.

Для того, щоб інші користувачі побачили створенний пост, вони повинні бути підписаними на профіль користувача, який його створив або перейти в його профіль.

1.5 Опис системи

Створення соціальних мереж стало однією з найбільш актуальних тем в останні роки. Соціальні мережі є важливими для збереження та зміцнення зв’язків між людьми, а також створення нових знайомств та можливостей для спілкування та співпраці.

Переваги соціальних мереж надзвичайно великі, особливо в сучасному світі, де люди все більше використовують інтернет та мобільні пристрої для

комунікації та отримання інформації. Соціальні мережі дозволяють людям спілкуватися між собою без будь-яких обмежень відносно місця перебування чи часу, а також дозволяють ділитися інформацією, фотографіями та відео.

Однією з важливих переваг соціальних мереж є зміцнення соціальних зв’язків між людьми. Соціальні мережі дозволяють людям зберігати зв’язки зі своїми родичами, друзями та колегами, які можуть бути розпорошені по всьому світу. Більш того, соціальні мережі дозволяють людям знайомитися з новими людьми, які мають подібні інтереси, що в свою чергу може призвести до створення нових проєктів та співпраці.

Соціальні мережі також важливі для підприємств, які можуть використовувати їх для маркетингу та залучення нових клієнтів. Компанії можуть створювати свої власні сторінки у соціальних мережах та використовувати їх для реклами своїх продуктів та послуг. Крім того, соціальні мережі можуть бути використані для залучення нових працівник ів та відкриття нових можливостей для бізнесу.

Зважаючи на вищесказане, був розроблен вебзастосунок соціальної мережі.

Можливості системи:

- створення посту;
- редагування профіля;
- підписка над користувача;
- відписка від користувача;
- ставити позначку «Подобається» під постом;
- знімати позначку «Подобається» під постом;
- перегляд профіля іншого користувача;
- пошук користувачів.

1.6 Огляд інструментів розробки

1.6.1 Django

Django – це відкрите програмне забезпечення, яке надає розробникам веб-додатків повний набір інструментів для швидкої розробки безпечних та масштабованих веб-додатків. Django базується на мові програмування Python і використовує модель Model-View-Controller (MVC), яка дозволяє розробникам легко розділяти логіку додатку, представлення та роботу з базою даних.

Django надає широкий набір функціональності, яка забезпечує зручну роботу з базами даних, URL-ми, формами, аутентифікацією та авторизацією, а також безпекою. Django також підтримує створення API та роботу з різними форматами даних, такими як JSON та XML.

Однією з головних переваг Django є зручність роботи з базою даних. Django використовує ORM (Object-Relational Mapping), який дозволяє розробникам працювати з базою даних на рівні об’єктів, замість написання SQL-запитів. Це спрощує роботу з базою даних та зменшує кількість помилок, що можуть виникнути під час роботи з базою даних.

Django також має потужний фреймворк для роботи з URL-адресами та відображенням. Він дозволяє легко налаштовувати URL-адреси та створювати відображення для різних типів запитів HTTP, включаючи GET, POST, PUT та DELETE. Django також надає розширену підтримку для роботи з формами, що дозволяє розробникам легко створювати та обробляти форми на стороні сервера.

Для забезпечення безпеки, Django має вбудований механізм аутентифікації та авторизації. Він також має вбудований захист від атак XSS та CSRF. Django також дозволяє налаштовувати рівень доступу до окремих сторінок та функцій додатку для різних користувачів, що забезпечує високий рівень безпеки додатку.

Django також дозволяє створювати API, що дозволяє інтегрувати додаток з іншими сервісами та додатками. Django підтримує різні формати даних, такі як JSON та XML, що дозволяє передавати дані у форматі, який зручний для

користувачів.

Однією з головних переваг Django є велика кількість сторонніх бібліотек та модулів, які розроблені для Django, що значно спрощує розробку веб-додатків. Такі бібліотеки, як Django REST Framework, дозволяють швидко створювати API для додатку, а бібліотека Django-allauth надає зручний механізм для авторизації користувачів через соціальні мережі.

1.6.2 SQLite

SQLite – це легковага вбудована реляційна база даних, яка зберігає дані у вигляді файлів на диску. SQLite була розроблена для того, щоб забезпечити простоту використання та підтримку невеликих баз даних, що не потребують великої кількості обсягу даних або великої кількості одночасних користувачів.

Основні переваги SQLite полягають у його легкості використання, швидкості та портативності. SQLite має декілька варіантів API, включаючи бібліотеку C, що дозволяє використовувати SQLite в різних мовах програмування, таких як Python, Java та PHP. SQLite підтримує більшість стандартів SQL та забезпечує операції з базами даних, такі як SELECT, INSERT, UPDATE та DELETE [9].

Однією з ключових особливостей SQLite є його вбудованість. База даних SQLite може бути вбудована в інші програми, що дозволяє зберігати дані в тому ж файлі, що й програма. Це робить SQLite дуже привабливим вибором для програм, що потребують зберігання невеликих обсягів даних, таких як мобільні додатки та різні вбудовані системи.

Ще одна важлива перевага SQLite – це його підтримка транзакцій. Транзакції дозволяють гарантувати цілісність даних, які записуються в базу даних. Якщо під час транзакції виникає помилка, то всі зміни, які були зроблені до цієї помилки, будуть автоматично скасовані, що дозволяє запобігти втраті даних.

SQLite також має досить добру продуктивність. В порівнянні з іншими базами даних, такими як MySQL та PostgreSQL, SQLite має менший обсяг коду, що дозволяє зменшит и його витрати на обробку запитів. Більшість операцій з базою даних SQLite можна виконувати швидше, ніж у більш тяжких системах.

Крім того, SQLite має велику кількість різних інструментів та ресурсів, що дозволяє розширювати та змінювати його функціональність. Ці ресурси включають в себе інструменти для адміністрування бази даних, такі як SQLite Administrator та SQLite Database Browser, а також різні розширення, які дозволяють використовувати SQLite в різних контекстах та з різними програмами [8].

Нарешті, варто відзначити, що SQLite є безкоштовним програмним забезпеченням з відкритим кодом. Це означає, що будь-хто може використовувати та змінювати код SQLite, що дозволяє використовувати цю базу даних у будь-яких проєктах, включаючи комерційні.

2 ПРОЄКТУВАННЯ

2.1 Використання UML під час розробки системи

UML (Unified Modeling Language) є стандартною мовою моделювання, яка широко використовується під час розробки програмного забезпечення. UML надає нотацію та набір графічних символів для візуального представлення різних аспектів системи.

Розглянемо основні переваги використання UML під час розробки системи.

Візуалізація системи: UML дозволяє розробникам створювати графічні моделі, які візуально відображають структуру та поведінку системи. Це допомагає зрозуміти і комунікувати складні концепції системи між розробниками, клієнтами та іншими стейкхолдерами.

Аналіз та проєктування: UML надає набір діаграм, таких як діаграми класів, діаграми послідовностей, діаграми активностей тощо, що допомагають аналізувати та проєктувати систему перед реалізацією. Ці діаграми дозволяють уточнювати вимоги, визначати структуру та поведінку компонентів системи, а також виявляти можливі проблеми та помилки ще на ранніх етапах розробки.

Комунікація та співпраця: UML надає загальну мову для комунікації між розробниками, дизайнерами, аналітиками та іншими стейкхолдерами. Всі учасники проєкту можуть використовувати UML-діаграми для обміну ідеями, обговорення та уточнення вимог, що сприяє кращому розумінню та співпраці всіх зацікавлених сторін.

Генерація коду: за допомогою UML можна створювати моделі, які використовуються для генерації початкового коду. Це може суттєво зекономити час та зусилля при розробці програмного забезпечення, оскільки частина рутинної роботи може бути автоматизована.

Документування: UML діаграми можуть служити як ефективний засіб

документування системи. Вони дозволяють зберігати та передавати інформацію про структуру, поведінку та інші аспекти системи, що полегшує розуміння та підтримку системи на різних етапах її життєвого циклу.

Загалом, використання UML під час розробки системи допомагає покращити якість розробки, співпрацю між розробниками та стейкхолдерами, а також полегшує розуміння та документування системи.

2.2 Діаграма варіантів використання

Діаграма варіантів використання (Use Case Diagram) є одним із найпоширеніших видів діаграм UML. Вона допомагає ідентифікувати функціональні можливості системи та взаємодії між користувачами та системою. Діаграма варіантів використання зображує акторів (користувачів або зовнішні системи) та варіанти використання (use case), які представляють сценарії взаємодії.

Розглянемо основні елементи діаграми варіантів використання.

Актори: актори представляють зовнішніх користувачі або системи, які взаємодіють з розроблюваною системою. Вони зображуються у вигляді піктограм людини або коробки.

Варіанти використання: варіанти використання описують функціональність системи або сценарії взаємодії між акторами та системою. Вони зображуються у вигляді овалів або прямокутників з назвою.

Зв'язки між акторами та варіантами використання: зв'язки відображають взаємодію між акторами та варіантами використання. Зв'язки можуть бути асоціаціями, залежностями або уточненнями, що показують, як актори взаємодіють з варіантами використання.

Розширення: деякі варіанти використання можуть бути розширенням інших варіантів використання. Це показує, що певний сценарій може бути розширений іншим сценарієм.

Діаграма варіантів використання дозволяє розробникам та зацікавленим сторонам отримати загальний огляд функціональності системи та визначити, як користувачі будуть взаємодіяти з системою. Вона служить основою для подальшого аналізу та проєктування системи, а також комунікації між розробниками та стейкхолдерами.

На рисунку 2.1 представлена діаграма варіантів використання.

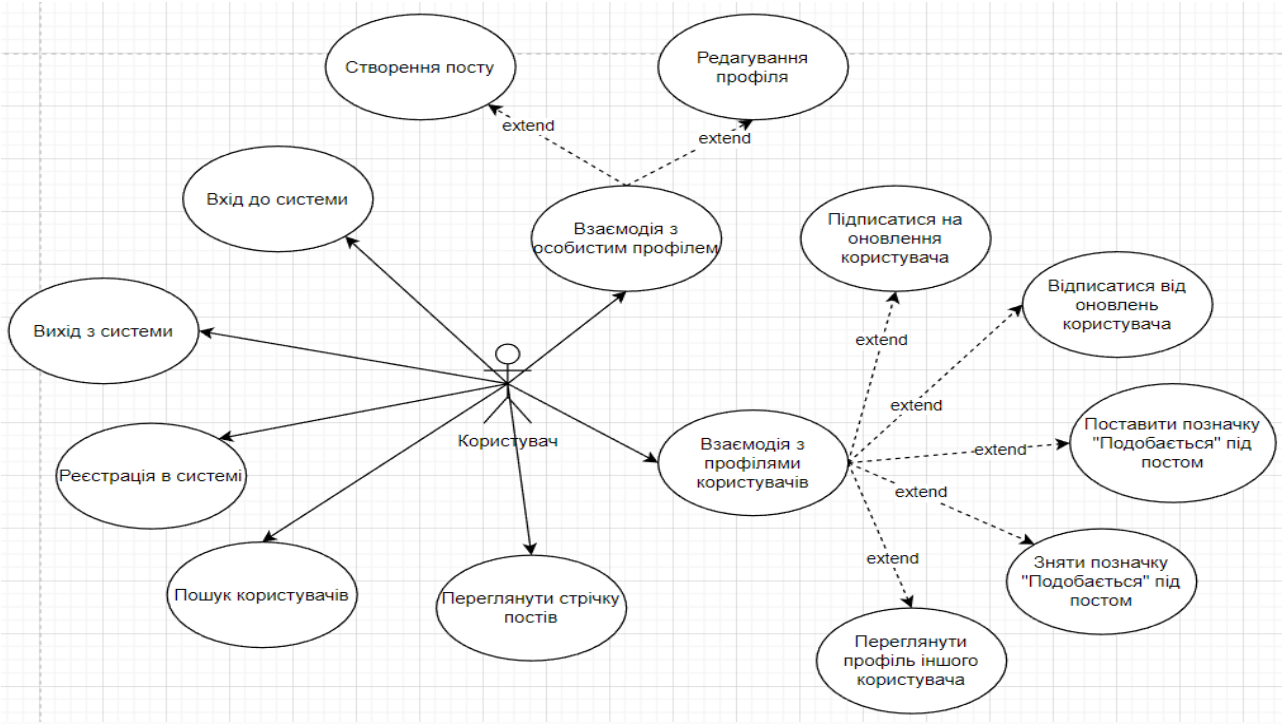

Рисунок 2.1 – Діаграма варіантів використання

На діаграмі представлено актора «Користувач», який відображає функціональні можливості взаємодії з системою.

Виділено 1 основний варіант використання – «Створення посту». Після того, як користувач зареєструється та/або авторизується в системі та натисне на кнопку «Upload Pics», система відображає форму для створення нового посту. Після заповнення форми та збереження даних, користувач має можливість перейти до особистого профілю, де відображається інформація про всі пости, які були опубліковані користувачем. Після публікації посту, інші користувачі, які підписані на користувача, що його опублікував, можуть побачити пост у стрічці

рекомендацій. На кожну з цих дій система надсилає запит до БД і повідомляє користувача про результат.

Варіанти використання визначають функціональні можливості. Кожен з них представляє певний спосіб використання. Таким чином, кожен варіант використання відповідає послідовності дій для того, щоб користувач міг отримати певний результат.

2.2.1 Опис варіантів використання

Прецедент «Реєстрація в системі»

Призначення: даний варіант використання надає можливість користувачу зареєструватися в системі.

Основний потік подій: даний варіант використання починає виконуватися, коли клієнт натискає на кнопку «Зареєструватися». Система відображає форму реєстрації, користувач вводить дані. Якщо введені дані валідні, то система створює новий обліковий запис та переадресовує на сторінку налаштування профілю.

Виняткова ситуація 1: користувач ввів невалідні дані – система відображає відповідне повідомлення. Користувач може повторно ввести дані.

Виняткова ситуація 2: користувач зареєстрований у системі – система відображає відповідне повідомлення та переадресовує користувача на сторінку входу до системи.

Виняткова ситуація 3: користувач натиснув кнопку «Скасувати» – переадресовує користувача на сторінку входу до системи.

Прецедент «Вхід до системи»

Призначення: даний варіант використання надає можливість зареєстрованому користувачу увійти у систему для використання.

Основний потік подій: даний варіант використання починає виконуватися, коли зареєстрованому користувачу потрібно авторизуватися. Система пропонує

ввести логін та пароль. Після того, як користувач ввів їх, система перевіряє правильність введених даних, і у випадку вдачі надає йому функціонал.

Передумова: перед початком виконання даного варіанта використання користувач повинен бути зареєстрований у системі.

Виняткова ситуація 1: якщо логін чи пароль невірні – система сповіщає про це користувача. Користувач може спробувати ще раз пройти авторизацію.

Прецедент «Вихід з системи»

Призначення: даний варіант використання надає можливість користувачу вийти з системи.

Основний потік подій: даний варіант використання починає виконуватися, коли користувачу потрібно вийти з системи. Користувач натискає на значок профіля та у списку дій обирає «Вихід». Після того, як користувач вийшов, система відображає сторінку входу до системи.

Передумова: перед початком виконання даного варіанта використання користувач повинен авторизуватися в системі.

Прецедент «Взаємодія з особистим профілем»

Призначення: даний варіант використання надає можливість користувачу взаємодіяти з особистим профілем та налаштовувати його.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на кнопку «Налаштування профілю» або «Upload Pics» в особистому кабінеті. Система відображає відповідну сторінку для введення та/або редагування інформації.

Передумова: перед початком виконання даного варіанта використання користувач повинен виконати вхід до системи.

Прецедент «Редагувати профіль»

Призначення: даний варіант використання надає можливість користувачу редагувати дані профілю.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає аватар профіля та обирає пункт «Налаштування профілю», система відображає сторінку редагування даних

профілю. Після внесення змін, користувач натискає кнопку «Зберегти», система зберігає оновлює інформацію.

Передумова: перед початком виконання даного варіанта використання користувач повинен виконати вхід до системи.

Виняткова ситуація 1: введено невалідні дані – система відобразить відповідне повідомлення. Користувач може змінити дані.

Виняткова ситуація 2: користувач натиснув кнопку «Скасувати» – система відобразить сторінку стрічки постів.

Прецедент «Створення посту»

Призначення: даний варіант використання надає можливість користувачу створити новий пост.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає кнопку «Upload Pics», система відображає форму для створення нового посту. Після заповнення форми, користувач натискає на кнопку «Зберегти», система зберігає до БД запис про новий пост.

Передумова: перед початком виконання даного варіанта використання користувач повинен виконати вхід до системи.

Виняткова ситуація 1: введено невалідні дані – система відобразить відповідне повідомлення. Користувач може змінити дані.

Виняткова ситуація 2: користувач натиснув кнопку «Скасувати» – система відобразить сторінку стрічки постів.

Прецедент «Взаємодія з профілями користувачів»

Призначення: даний варіант використання надає можливість користувачу взаємодіяти з профілями інших користувачів.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи взаємодіє з профілями інших користувачів, а саме: підписується на оновлення, відписується від оновлень, переглядає профіль, тощо. Система реагує на кожну дію та відображує відповідну сторінку або функціонал.

Передумова: перед початком виконання даного варіанта використання

користувач повинен виконати вхід до системи.

Прецедент «Переглянути профіль іншого користувача»

Призначення: даний варіант використання надає можливість користувачу переглядати інформацію профілю інших користувачів системи.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на аватар або ім'я профілю іншого користувача. Система відображає сторінку профілю користувача та надає доступ до взаємодії з ним.

Передумова: перед початком виконання даного варіанта використання користувач повинен виконати вхід до системи.

Прецедент «Підписатися на оновлення користувача»

Призначення: даний варіант використання надає можливість користувачу підписуватися на оновлення інших користувачів системи.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на кнопку «Підписатися» на сторінці профілю іншого користувача. Система змінює функціонал кнопки на «Відписатися» та інформацію про кількість підписників користувача.

Альтернативний потік подій: якщо вже оформлена підписка на оновлення користувача, система сповістить про це у вигляді напису кнопки «Відписатися».

Передумова: перед початком виконання даного варіанта використання користувач повинен знаходитися на сторінці профілю іншого користувача та не бути підписаним на нього.

Прецедент «Відписатися від оновлень користувача»

Призначення: даний варіант використання надає можливість користувачу відписуватися від оновлень інших користувачів системи.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на кнопку «Відписатися» на сторінці профілю іншого користувача. Система змінює функціонал кнопки на «Підписатися» та інформацію про кількість підписників користувача.

Альтернативний потік подій: якщо не оформлена підписка на оновлення

користувача, система сповістить про це у вигляді напису кнопки «Підписатися».

Передумова: перед початком виконання даного варіанта використання користувач повинен знаходитися на сторінці профілю іншого користувача та бути підписаним на нього.

Прецедент «Поставити позначку «Подобається» під постом»

Призначення: даний варіант використання надає можливість користувачу поставити позначку «Подобається» під постом іншого користувача.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на піктограму «Подобається» під постом у стрічці. Система змінює кількість позначок «Подобається» та відображає їх під постом.

Альтернативний потік подій: якщо позначка вже поставлена, система сповістить про це у вигляді відповідного напису.

Передумова: перед початком виконання даного варіанта використання користувач повинен підписатися на іншого(их) користувачів та знаходитися на сторінці стрічки постів. Також користувач не повинен був ставити позначку «Подобається» під постом, який оцінюється.

Прецедент «Зняти позначку «Подобається» під постом»

Призначення: даний варіант використання надає можливість користувачу зняти позначку «Подобається» під постом іншого користувача.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на піктограму «Подобається» під постом у стрічці. Система змінює кількість позначок «Подобається» та відображає їх під постом.

Альтернативний потік подій: якщо позначка вже знята, система сповістить про це у вигляді відповідного напису.

Передумова: перед початком виконання даного варіанта використання користувач повинен підписатися на іншого(их) користувачів та знаходитися на сторінці стрічки постів. Також користувач повинен був поставити позначку «Подобається» під постом, який оцінюється.

Прецедент «Пошук користувачів»

Призначення: даний варіант використання надає можливість користувачу шукати інших користувачів системи.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи вводить ім’я іншого користувача по строки пошуку та натискає піктограму пошуку, система відображає результат пошуку у вигляді переліку користувачів.

Передумова: перед початком виконання даного варіанта використання користувач повинен виконати вхід до системи.

Прецедент «Переглянути стрічку постів»

Призначення: даний варіант використання надає можливість користувачу переглядати стрічку постів користувачів, на оновлення яких оформлена підписка.

Основний потік подій: даний варіант використання починає виконуватися, коли авторизований користувач переходить на головну сторінку, система відображає стрічку постів інших користувачів, на оновлення яких оформлена підписка. Користувач може взаємодіяти з постами та користувачами.

Передумова: перед початком виконання даного варіанта використання користувач повинен виконати вхід до системи.

2.3 Діаграма діяльності

Діаграма діяльності (Activity Diagram) є одним з видів структурних діаграм в моделюванні програмного забезпечення. Вона використовується для візуалізації послідовності дій або процесів, що відбуваються в системі. Діаграма діяльності дозволяє показати, як об’єкти системи взаємодіють між собою та як вони змінюють свій стан внаслідок виконання різних дій.

Розглянемо основними елементи діаграми діяльності.

Дії (Actions): вони представлені прямокутниками і показують окремі дії або

операції, які виконуються в системі. Наприклад, «Відкрити файл», «Надіслати повідомлення» і т.д.

Рішення (Decisions): вони позначаються ромбами і представляють рішення або умови, які визначають напрямок виконання діаграми. Наприклад, "Якщо умова X виконується, виконати дію A, інакше виконати дію B".

Паралельність (Parallel): вона позначається паралельними лініями і використовується для показу одночасного виконання декількох дій або процесів.

Розгалуження (Fork/Join): використовується для показу розгалуження та злиття потоків виконання. Розгалуження (Fork) вказує на поділ виконання на кілька паралельних шляхів, а злиття (Join) показує об’єднання шляхів в один.

Переходи (Transitions): показують послідовність виконання дій та потоки управління. Вони позначаються стрілками, що з’єднують дії між собою або з рішеннями. Стрілки можуть мати позначення, що вказують на умови або події, які спричиняють перехід з однієї дії до іншої.

Початок (Start) і Завершення (End): показують початок і кінець діаграми діяльності. Початок позначається заповненим колом, а завершення - заповненим багатокутником.

Зв’язки між елементами показують послідовність виконання дій. Наприклад, стрілки можуть з’єднувати дії між собою або з рішеннями, щоб показати, яка дія виконується після іншої.

Діаграма діяльності дозволяє візуалізувати послідовність дій або процесів в системі. Вона допомагає аналізувати та розробляти процеси, виявляти залежності та контроль потоку даних. Основні елементи діаграми діяльності включають дії, рішення, форки та злиття, початок та завершення, а також переходи.

Ця діаграма дозволяє команді розробників та зацікавленим сторонам краще розуміти логіку та взаємозв’язки в системі. Вона може використовуватися на різних етапах розробки програмного забезпечення – від аналізу вимог до документування та впровадження. Діаграма діяльності є потужним інструментом для моделювання та візуалізації процесів, що сприяє ясному

розумінню та спілкуванню між розробниками та зацікавленими сторонами.

Наведемо діаграму діяльності, що описує модель поведінки варіанта використання «Створення посту». Діаграма представлена на рисунках 2.2 – 2.3.

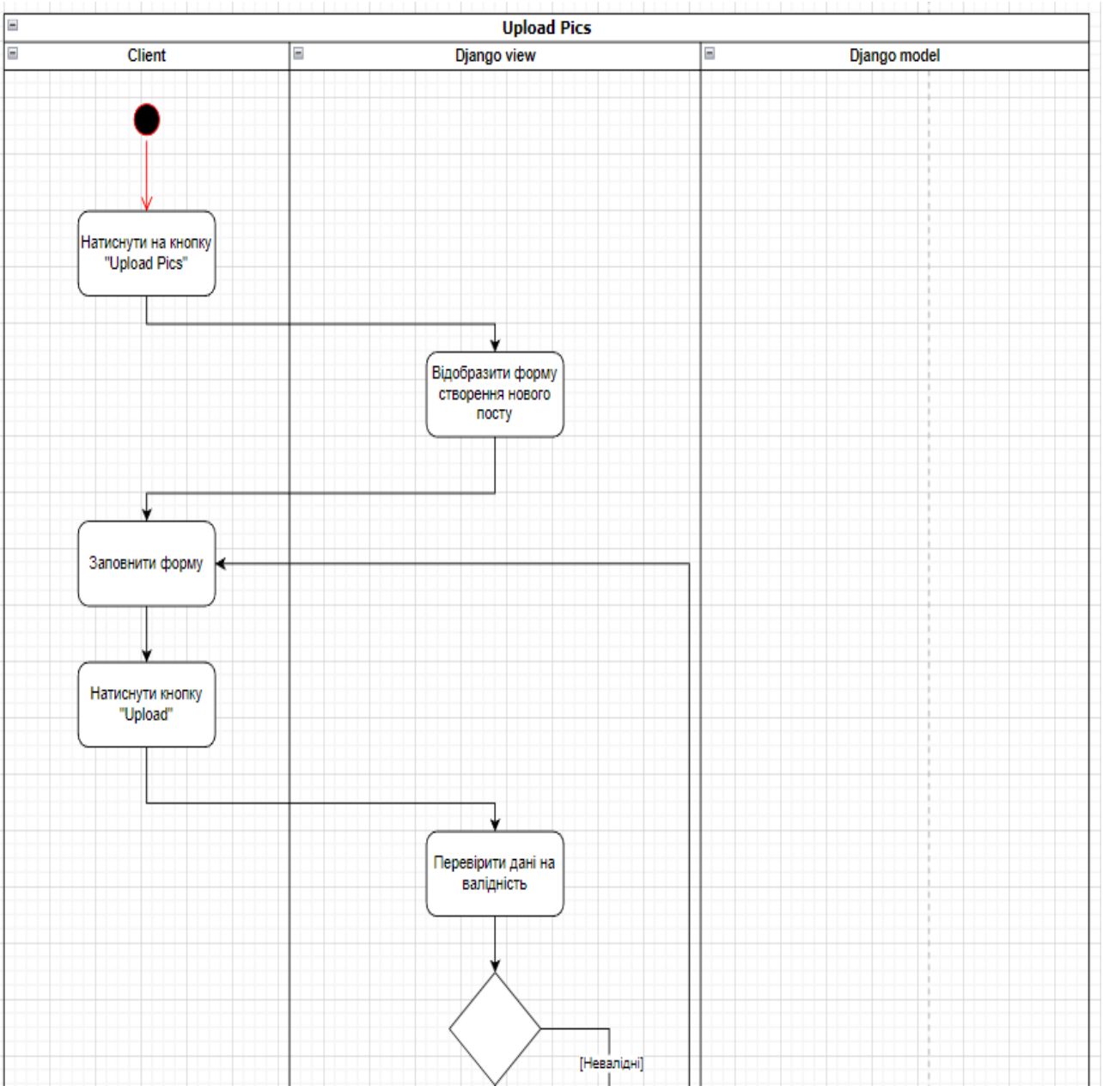

Рисунок 2.2 – Діаграма діяльності

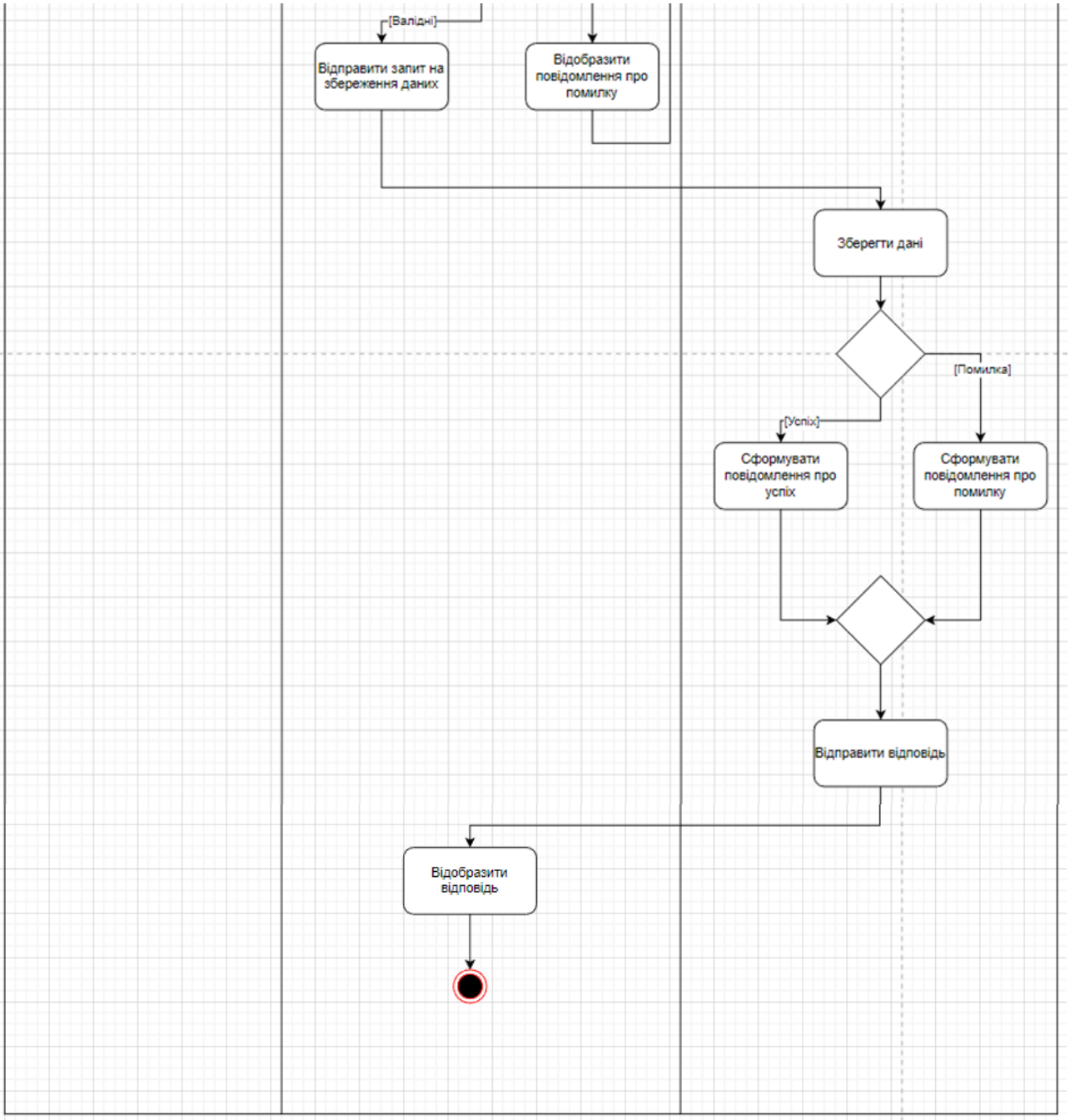

Рисунок 2.3 – Діаграма діяльності

2.4 Діаграма послідовності

Діаграма послідовності (Sequence Diagram) є одним з видів діаграм UML (Unified Modeling Language) і використовується для моделювання взаємодії між об’єктами або компонентами системи в певному часовому порядку. Вона дозволяє візуалізувати послідовність повідомлень, які передаються між

об’єктами під час виконання певної функціональності або сценарію.

Розглянемо основні елементи діаграми послідовності.

Об’єкти (Objects): представляють учасників системи, з якими взаємодіємо. Кожен об’єкт позначається прямокутником з назвою об’єкта. Об’єкти можуть мати свої властивості та стан, що показується в діаграмі.

Послання повідомлень (Message): вони показують взаємодію між об’єктами, передачу повідомлень або виклик методів. Послання позначаються стрілками, які показують напрямок передачі повідомлення, а також можуть мати назву та параметри.

Ретурни (Return): показують повернення значення від об’єкта, який отримав повідомлення.

Засилання (Link): використовуються для показу зв’язків або асоціацій між об’єктами. Засилання позначаються лініями з маленькими кільцями на кінцях.

Життєвий цикл об’єкта (Object Lifeline): показує часовий проміжок існування об’єкта. Він представляється вертикальною лінією, яка простягається вздовж діаграми та позначає послідовність дій об’єкта.

Діаграма послідовності використовується для аналізу, проєктування та візуалізації взаємодії між об’єктами під час виконання функціональності або сценарію. Вона допомагає розібрати складну систему на прості етапи, виявити залежності між об’єктами та зрозуміти порядок виконання операцій.

Діаграма послідовності UML є потужним інструментом для моделювання та аналізу взаємодії між об’єктами системи. Вона дозволяє візуалізувати послідовність повідомлень, передачу даних та потік управління між об’єктами. Основні елементи діаграми послідовності включають об’єкти, повідомлення, ретурни, засилання та життєвий цикл об’єкта.

Використання діаграми послідовності дозволяє розуміти та аналізувати взаємодію між об’єктами, ідентифікувати можливі проблеми або неузгодженості в логіці програми та полегшує комунікацію між розробниками та зацікавленими сторонами. Вона є важливим інструментом в процесі розробки програмного забезпечення та допомагає покращити розуміння та взаємодію всіх учасників

проєкту.

На рисунку 2.4 описана діаграма послідовності прецедента «Створення посту».

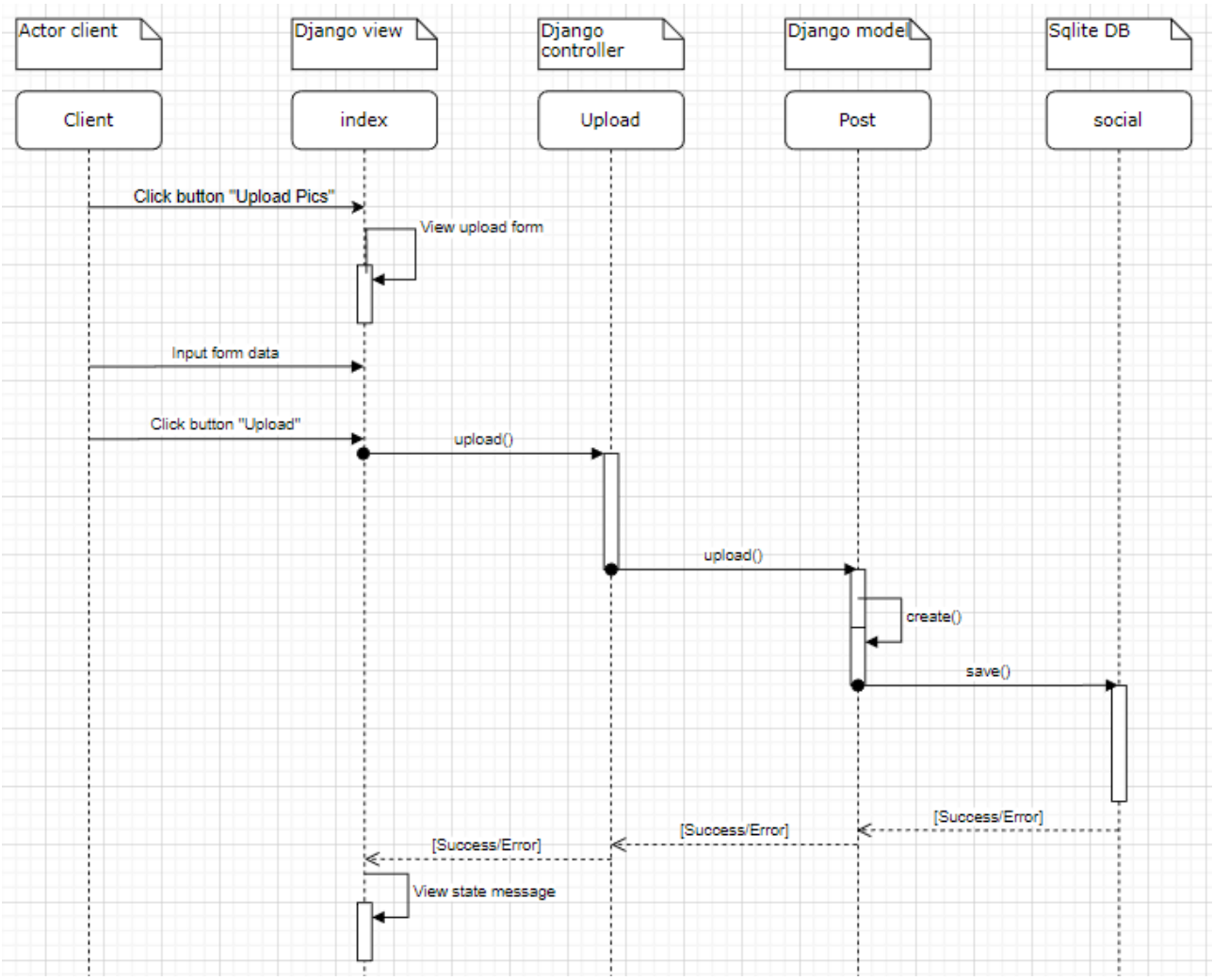

Рисунок 2.4 – Діаграма послідовності

2.5 Діаграма розгортання

Діаграма розгортання (Deployment Diagram) є одним з видів діаграм UML (Unified Modeling Language) і використовується для моделювання фізичного розміщення компонентів програмної системи на апаратному забезпеченні. Ця діаграма надає візуальне представлення інфраструктури системи, включаючи

сервери, комп’ютери, мережеві вузли, зв’язки між ними та компоненти програмного забезпечення.

Розглянемо основні елементи діаграми розгортання.

Вузли (Nodes): представляють фізичні або віртуальні обчислювальні пристрої, такі як сервери, комп’ютери або мобільні пристрої. Кожен вузол позначається прямокутником з назвою вузла.

Артефакти (Artifacts): представляють фізичні або логічні компоненти програмного забезпечення, які виконуються на вузлах. Наприклад, це можуть бути файли, бібліотеки, модулі або контейнери. Артефакти позначаються прямокутниками з назвою або символом артефакта.

Зв’язки (Connections): показують зв’язки та комунікацію між вузлами. Вони позначаються лініями, що з’єднують вузли та артефакти.

Канали зв’язку (Communication Channels): використовуються для показу засобів передачі даних або комунікації між вузлами, наприклад, мережеві канали, протоколи або інші механізми зв’язку.

Вузли-контейнери (Container Nodes): використовуються для групування артефактів, які виконуються на одному вузлі. Вони допомагають організувати компоненти програмного забезпечення та їх взаємозв’язки.

Діаграма розгортання використовується для моделювання фізичного розміщення компонентів системи, інфраструктури та взаємозв’язків між ними. Вона допомагає визначити архітектурні аспекти системи, розробити план розгортання, оцінити необхідні ресурси та з’ясувати можливі проблеми зі зв’язками або доступністю.

Діаграма розгортання UML є важливим інструментом для моделювання та планування фізичного розміщення компонентів програмної системи на апаратному забезпеченні. Вона дозволяє візуалізувати інфраструктуру системи, включаючи вузли, артефакти, зв’язки та вузли-контейнери. Діаграма розгортання допомагає зрозуміти архітектурні аспекти системи, оцінити ресурси та виявити можливі проблеми зі зв’язками. Вона є важливим інструментом для комунікації між розробниками та інженерами і допомагає забезпечити успішне

розгортання програмного забезпечення.

На рисунку 2.5 наведено діаграму розгортання для вебсайту соціальної мережі, створеного засобами Django.

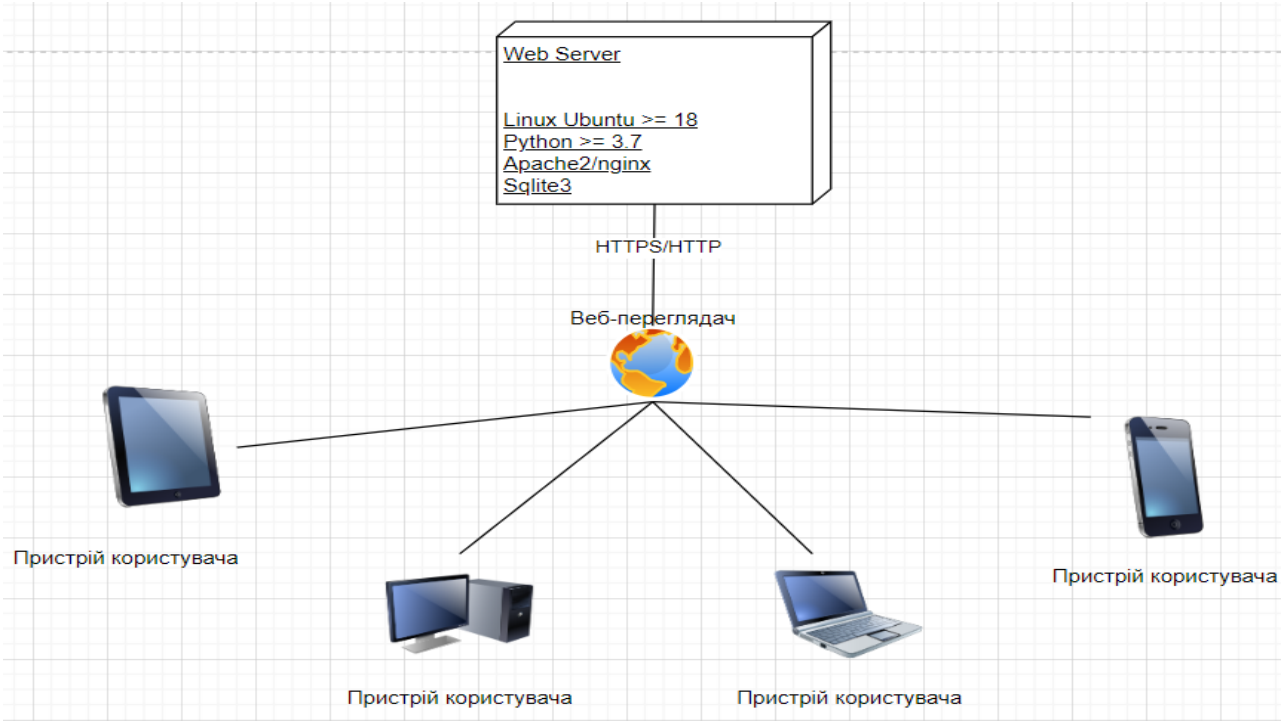

Рисунок 2.5 – Діаграма розгортання

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Опис інструментів розробки

Для реалізації був використаний python-фреймворк Django.

Sqlite3 – це вбудована реляційна база даних, яка зберігає дані у локальних файлових базах даних. Вона досить легка у використанні, не потребує окремого сервера баз даних та надає можливість прямого доступу до бази даних з допомогою SQL-запитів.

Bootstrap – фреймворк для розробки інтерфейсів веб-сторінок (фронтенду). Він надає набір готових компонентів CSS і JavaScript, які можна використовувати для швидкої розробки сучасних і адаптивних веб-інтерфейсів.

3.2 Структура Django проєкту

Django пропонує специфічну структуру проєкту, яка допомагає організовувати код і ресурси веб-додатку. Розглянемо основні компоненти структури Django проєкту [10].

Проєкт (Project): Django проєкт представляє собою кореневу папку, що містить всі файли і налаштування веб-додатку. В цій папці зазвичай файли управління проєктом, такі як *manage.py*, *settings.py*, *urls.py* та інші.

Додатки (Apps): Django додатки є незалежними компонентами, які виконують певні функції в рамках проєкту. Кожен додаток може мати свою модель даних, представлення (views), шаблони (templates) і URL-маршрутизацію. У папці проєкту зазвичай створюються окремі папки для кожного додатку.

Моделі (Models): моделі Django визначають структуру бази даних і використовуються для збереження і отримання даних. Вони представляють

собою класи Python, які використовують ORM (Object-Relational Mapping) для взаємодії з базою даних.

Представлення (Views): представлення Django відповідають за обробку HTTP-запитів і повернення HTTP-відповідей. Вони виконують бізнес-логіку додатку, отримують дані з моделей, інтерпретують їх і передають до шаблонів.

Шаблони (Templates): шаблони Django використовуються для генерації веб-сторінок з динамічними даними. Вони використовують шаблонний движок Django, який дозволяє вставляти дані з моделей і передавати їх на веб-сторінку.

URL-маршрутизація (URL Routing): файл *urls.py* визначає маршрутизацію URL-адрес до відповідних представлень в додатках. Він вказує, яке представлення буде виконано для кожного URL-запиту.

3.3 Моделі

У Django моделі використовуються для визначення структури даних і взаємодії з базою даних. Модель Django є класом Python, який наслідується від базового класу *django.db.models.Model*. Кожне поле моделі представляє стовпців бази даних [1, 2].

Нижче наведено приклад моделі (див. рис. 3.1).

```
class Post(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4)
    user = models.CharField(max_length=100)
    image = models.ImageField(upload_to='post_images')
    caption = models.TextField()
    created_at = models.DateTimeField(default=datetime.now)
    no_of_likes = models.IntegerField(default=0)

    def __str__(self):
        return self.user
```

Рисунок 3.1 – Приклад моделі

Код моделей проєкту наведено в Додатку А.

3.4 Представлення

У Django представлення (views) відповідають за обробку HTTP-запитів і повернення HTTP-відповідей. Вони містять бізнес-логіку веб-додатку і забезпечують взаємодію між моделями (дані) та шаблонами (вигляд) [1, 7].

Нижче наведено приклад представлення (див. рис. 3.2).

```python
@login_required(login_url='signin')
def upload(request):
    if request.method == 'POST':
        user = request.user.username
        image = request.FILES.get('image_upload')
        caption = request.POST['caption']

        new_post = Post.objects.create(user=user, image=image, caption=caption)
        new_post.save()
        return redirect('/')
    else:
        return redirect('/')
```

Рисунок 3.2 – Приклад представлення

Код представлень проєкту наведено в Додатку Б.

3.5 Шаблони

Один із ключових компонентів Django – це система шаблонів, яка дозволяє розділити логіку веб-сторінок від їх відображення.

Шаблони Django використовують мову шаблонів, яка має простий синтаксис і дозволяє виконувати динамічне відображення даних. Основна ідея полягає в тому, що ви створюєте HTML-шаблон зі спеціальними розмітками та вбудованими виразами, які виконуються під час відображення сторінки [2, 7].

Код шаблону наведено в Додатку В.

3.6 URL-маршрутизація

Django маршрутизація відповідає за визначення, який веб-шлях (URL) співставляється з певним переглядом (view) або функцією. Маршрутизація дозволяє визначати, який код буде виконуватися при отриманні запиту на певну URL-адресу.

Основним компонентом маршрутизації в Django є файл *urls.py*, який знаходиться в кожному додатку або в кореневій папці проєкту. В цьому файлі визначаються шляхи (URL-адреси) та співставляються зі специфічними переглядами [4, 10].

Нижче наведено приклад маршрутизації (див. рис. 3.3).

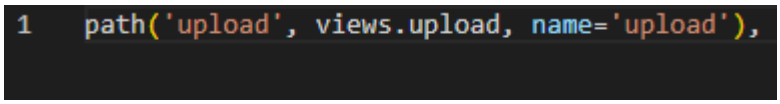

Рисунок 3.3 – Приклад маршрутизації

Маршрутизація проєкту наведена в додатку Г.

3.7 Тестування проєкту

3.7.1 Unit-тест

Unit-тестування (unit testing) – це процес виконання автоматизованих тестів для окремих "одиниць" програмного коду, таких як функції, методи чи класи, для перевірки їх правильності та відповідності очікуваному поведінці.

Unit-тестування є важливою практикою розробки програмного забезпечення, оскільки допомагає забезпечити якість коду, полегшує рефакторинг та зменшує кількість помилок. В Django та багатьох інших фреймворках є вбудовані засоби для написання та виконання unit-тестів [3, 6].

Приклад такого тестування наведено на рисунку 3.4.

```python
from django.test import TestCase, RequestFactory
from myapp.models import Post
from myapp.views import upload

class UploadTestCase(TestCase):
    def setUp(self):
        self.factory = RequestFactory()

    def test_upload_post(self):
        # Create a fake request object
        request = self.factory.post('/upload/', {
            'caption': 'Test caption',
            'image_upload': 'test_image.jpg'
        })
        request.user = User.objects.create(username='john')
        # Call the view function
        response = upload(request)
        # Check if the response is a redirect to the home page
        self.assertEqual(response.status_code, 302)
        self.assertEqual(response.url, '/')
        # Check if a new post was created in the database
        self.assertEqual(Post.objects.count(), 1)
        # Retrieve the created post from the database
        new_post = Post.objects.first()
        # Check if the post attributes are set correctly
        self.assertEqual(new_post.user.username, 'john')
        self.assertEqual(new_post.caption, 'Test caption')
        self.assertEqual(new_post.image.name, 'test_image.jpg')

    def test_upload_get_request(self):
        # Create a fake GET request object
        request = self.factory.get('/upload/')
        # Call the view function
        response = upload(request)
        # Check if the response is a redirect to the home page
        self.assertEqual(response.status_code, 302)
```

Рисунок 3.4 – Unit-тест функції upload

3.7.2 Integration-тест

Integration-тестування (integration testing) – це процес перевірки взаємодії між різними компонентами або модулями програмного забезпечення, щоб виявити помилки на рівні їх інтеграції [5].

На рисунку 3.5 наведено приклад integration-тестування на функції upload.

```python
from django.test import TestCase, Client
from django.contrib.auth.models import User
from myapp.models import Post

class UploadIntegrationTestCase(TestCase):
    def setUp(self):
        self.client = Client()
        self.user = User.objects.create_user(username='testuser', password='testpassword')

    def test_upload_post(self):
        self.client.login(username='testuser', password='testpassword')
        # Create a POST request with form data
        with open('test_image.jpg', 'rb') as image_file:
            response = self.client.post('/upload/', {
                'caption': 'Test caption',
                'image_upload': image_file,
            })
        # Check if the response is a redirect to the home page
        self.assertRedirects(response, '/')
        # Check if a new post was created in the database
        self.assertEqual(Post.objects.count(), 1)
        # Retrieve the created post from the database
        new_post = Post.objects.first()
        # Check if the post attributes are set correctly
        self.assertEqual(new_post.user.username, 'testuser')
        self.assertEqual(new_post.caption, 'Test caption')
        self.assertEqual(new_post.image.name, 'test_image.jpg')

    def test_upload_get_request(self):
        # Create a GET request
        response = self.client.get('/upload/')
        # Check if the response is a redirect to the home page
        self.assertRedirects(response, '/')
```

Рисунок 3.5 – Integration-тест функції upload

3.8 Керівництво користувача

3.8.1 Рівень підготовки користувача

Користувач сайту повинен володіти певною кваліфікацією. Навички користувача для роботи з ПК, та з web-браузером. Знайомство з Керівництвом користувача.

3.8.2 Підготовка до роботи

Запуск системи.

Доступ до сайту здійснюється через мережу Інтернет за допомогою звичайного web-браузера. Адреса сайту в мережі Інтернет: https://social.loc.

Для коректної роботи клієнтської частини повинен використовуватися браузер Google Chrome, Mozilla Firefox, Opera, Safari.

При вході на Сайт користувач потрапляє на сторінку входу до системи.

На Сайті розрізняються наступні групи користувачів:

Анонімний користувач (не увійшли або не зареєстровані, мають доступ до сторінок входу).

Користувач - приватна особа, авторизований на сайті (далі Користувач), що має доступ до функцій користувача у особистому кабінеті.

3.8.3 Реєстрація та вхід до системи

При вході на сайт, система відображає сторінку входу до системи (див. рис. 3.6). Якщо Ви зареєстровані, то потрібно ввести логін та пароль у відповідні поля та натиснути кнопку «Login».

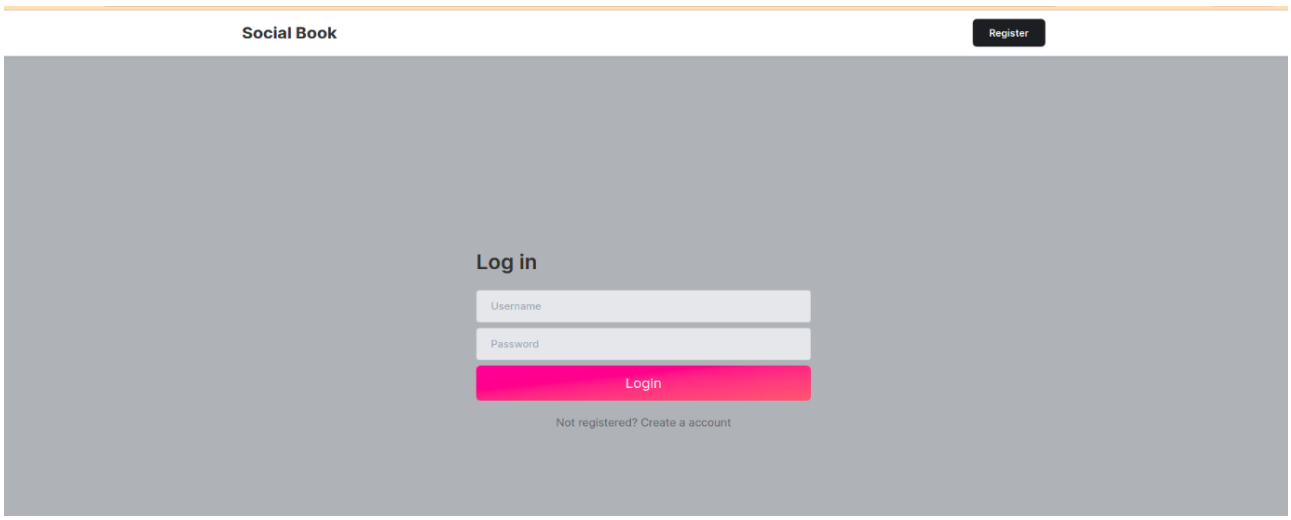

Рисунок 3.6 – Форма входу до системи

Якщо Ви ще не маєте профілю користувача, то потрібно натиснути на кнопку «Register». Система відобразить форму створення нового облікового запису (див. рис. 3.7).

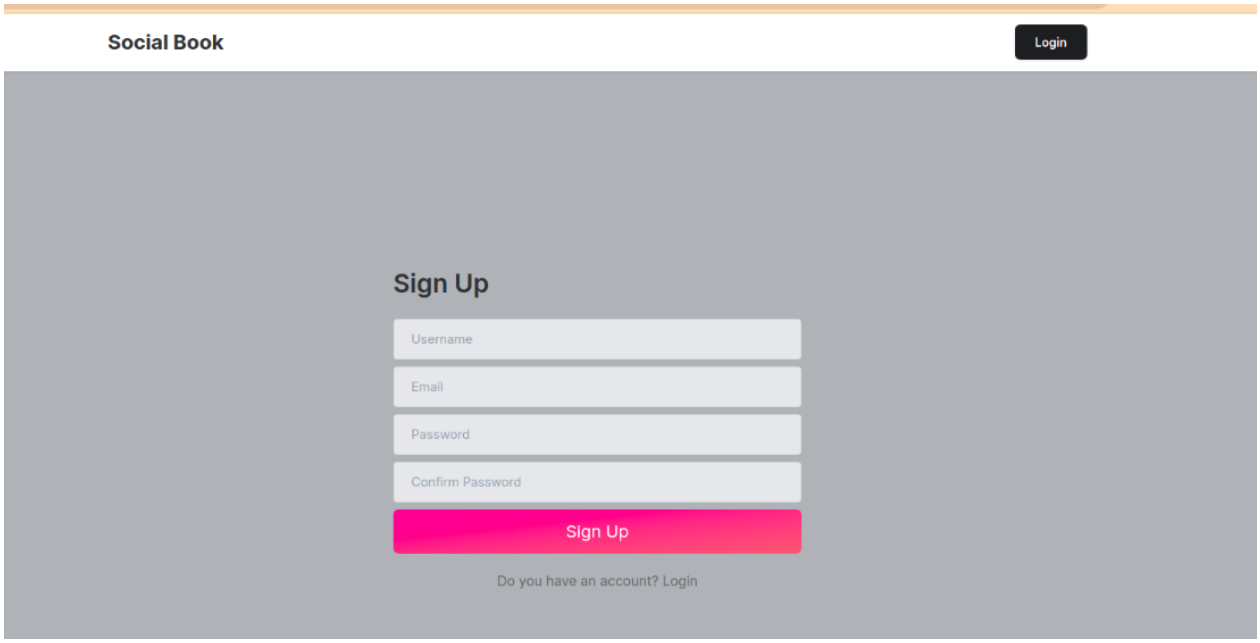

Рисунок 3.7 – Форма реєстрації

Після заповнення форми входу або реєстрації, система автоматично перевіряє валідність, якщо дані валідні, то відображає особистий кабінет користувача (див. рис. 3.8) або сторінку налаштувань профіля, якщо користувач вперше увійшов в аккаунт (див. рис. 3.9).

Рисунок 3.8 – Особистий кабінет

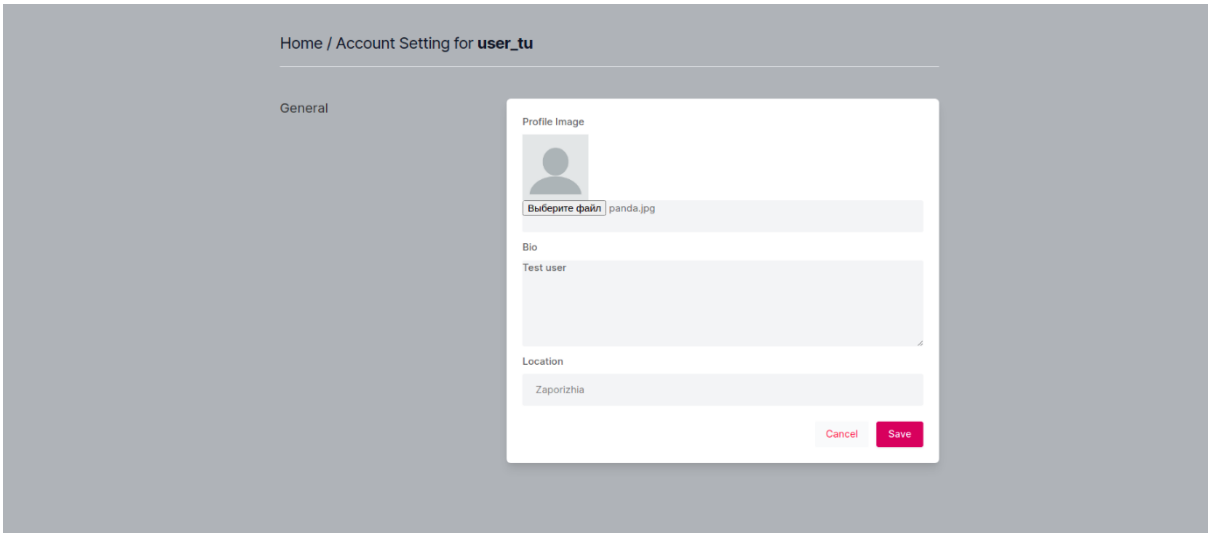

Рисунок 3.9 – Налаштування профілю

Якщо при вході або авторизації виникли помилки (користувача вже зареєстровано, невалідні дані, невірний логін або пароль, тощо), то система відобразить відповідне повідомлення. Приклад повідомлень наведено на рисунках 3.10 – 3.11.

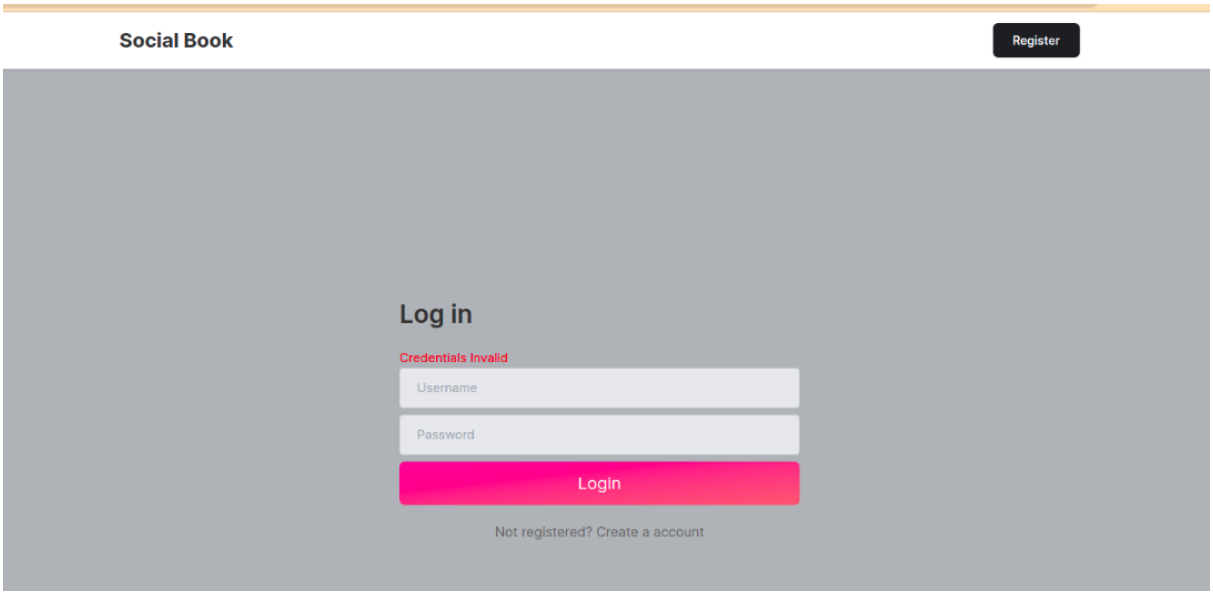

Рисунок 3.10 – Невірний логін та/або пароль

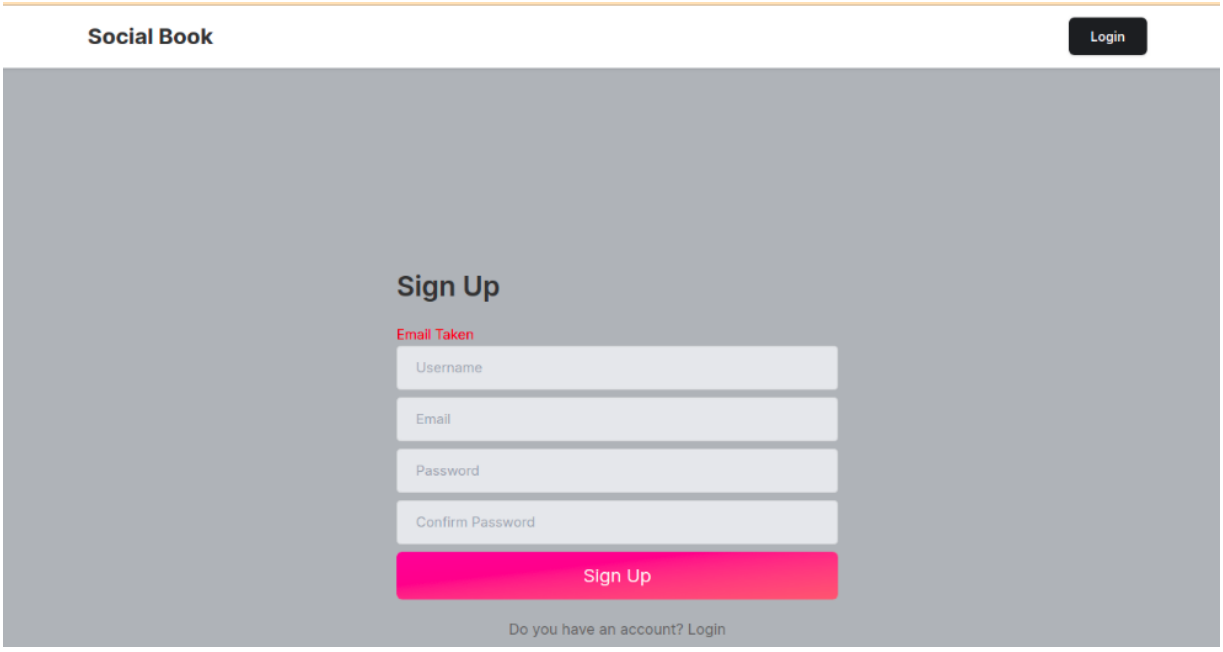

Рисунок 3.11 – Користувач з таким email вже існує

3.8.4 Взаємодія з профілем

Після авторизації в системі, Ви може переглянути або редагувати дані профілю. Для цього потрібно натиснути на аватар профілю (див. рис. 3.12) та обрати пункт «Profile» для перегляду інформації профілю (див. рис. 3.13) або «Account setting» для редагування інформації про користувача (див. рис. 3.14).

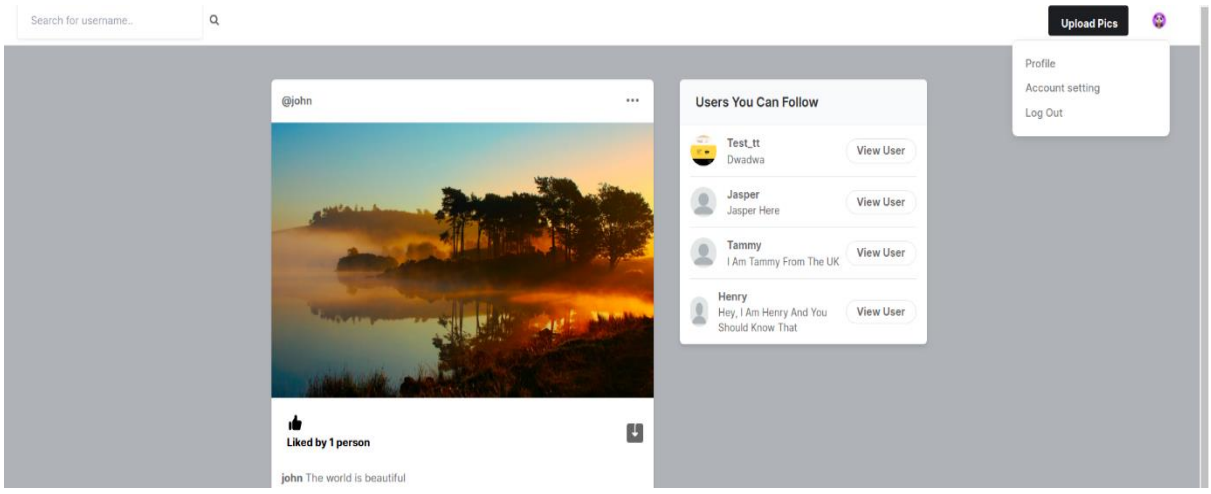

Рисунок 3.12 – Взаємодія з профілем

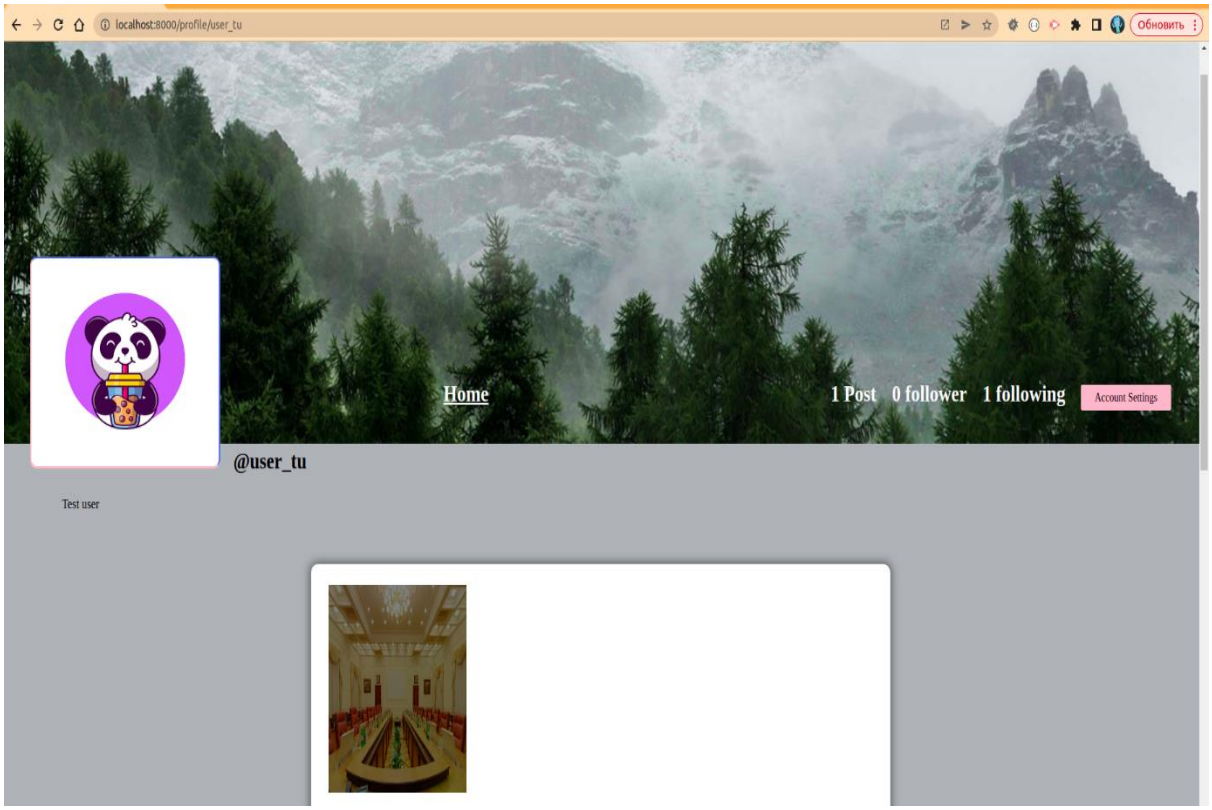

Рисунок 3.13 – Перегляд профілю

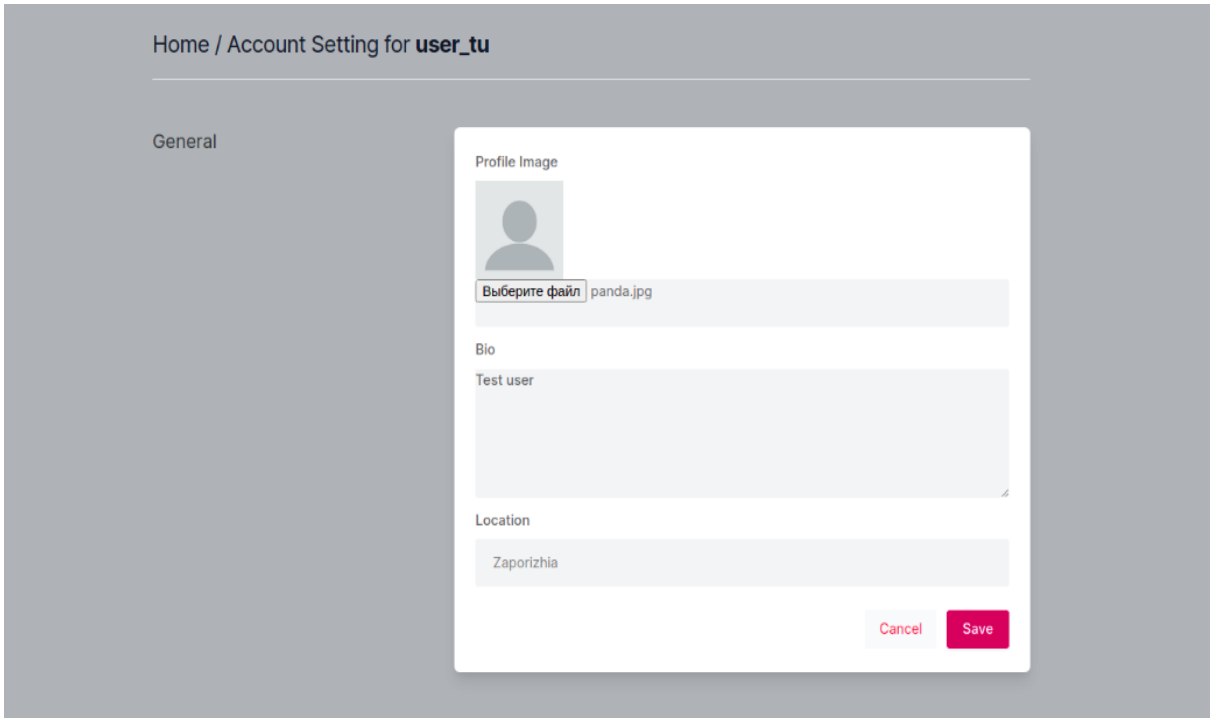

Рисунок 3.14 – Редагування профілю

3.8.5 Взаємодія з іншими користувачами

Після входу до системи та переходу на головну сторінку (стрічку постів), Ви маєте можливість взаємодіяти з іншими користувачі, наприклад: передивлятися профіль, підписуватися на оновлення, взаємодіяти з постами, тощо. Якщо Ви ще не підписані на жодного користувача, то сторінка буде мати вигляд, як зображено на рисунку 3.15.

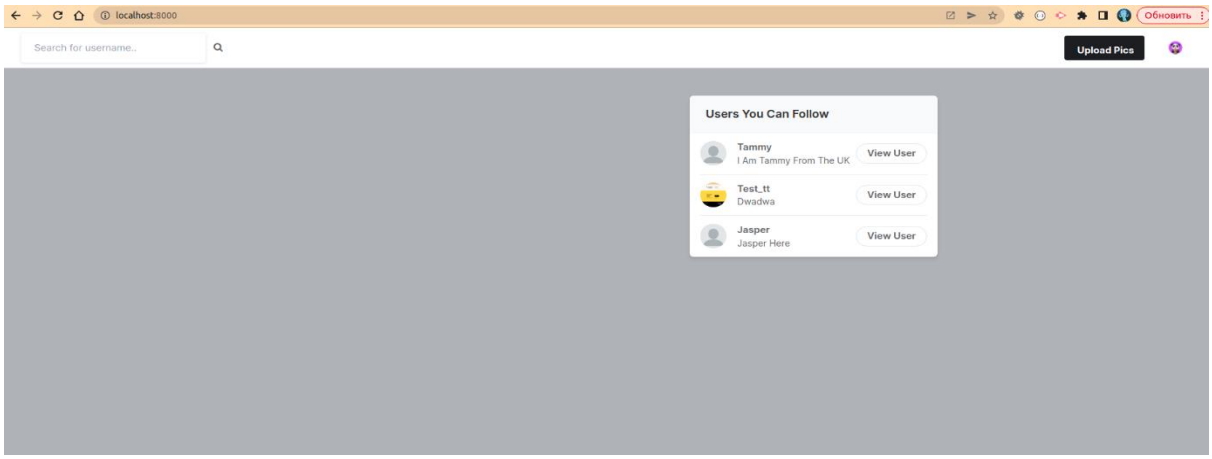

Рисунок 3.15 – Головна сторінка без підписок

Для підписки на оновлення іншого користувача, потрібно або натиснути на кнопку «View User» на боковій панелі запропонованих користувачів, або скористатися пошуком за ім’ям користувача (див. рис. 3.16 – 3.17).

Рисунок 3.16 – Введення пошукового запиту

Рисунок 3.17 – Результат пошуку

Після переходу на сторінку користувача, Ви можете переглянути його пости, підписатися/відписатися від оновлень. Для оформлення підписки на оновлення користувача, потрібно натиснути на кнопку «Follow», аналогічно для відписки – «Unfollow» (див. рис. 3.18).

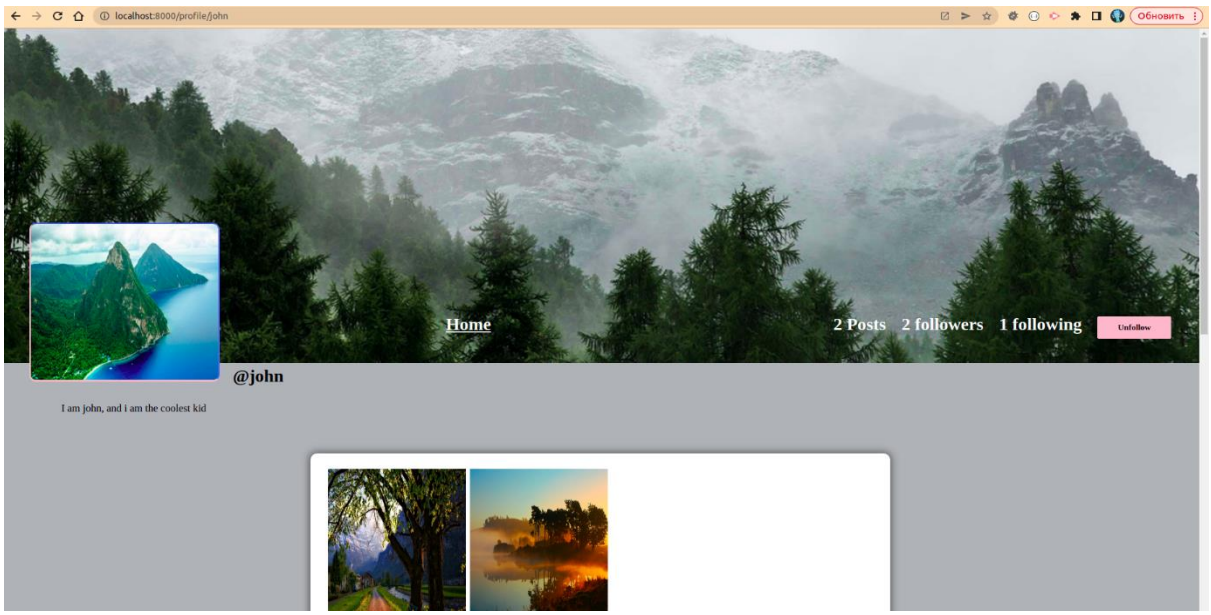

Рисунок 3.18 – Оформлення підписки на користувача

Якщо після оформлення підписки на користувача повернутися на головну сторінку, то можна побачити пости цього користувача, а також взаємодіяти з ними (рис. 3.19).

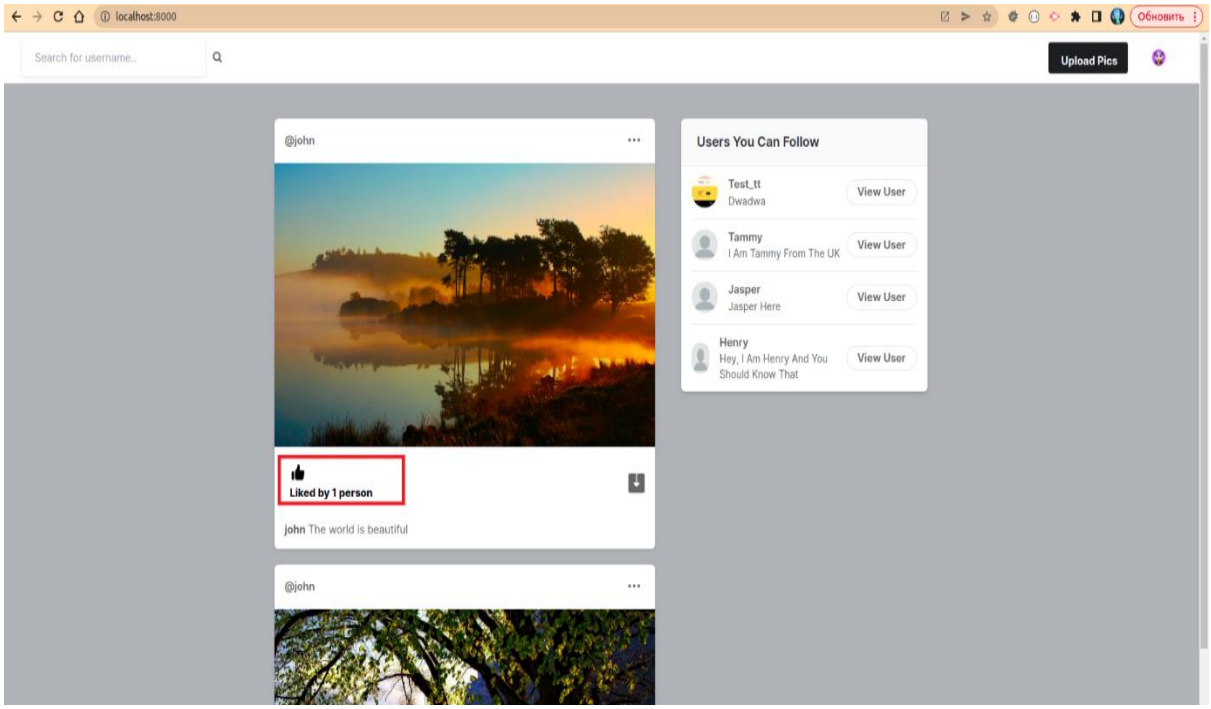

Рисунок 3.19 – Взаємодія з постом

3.8.6 Створення посту

Для того щоб створити власний пост, потрібно натиснути на кнопку «Upload Pics» у верхньому правому куті, система відобразить форму створення нового посту. Обираємо фото з пристрою та підписуємо його (див. рис. 3.20).

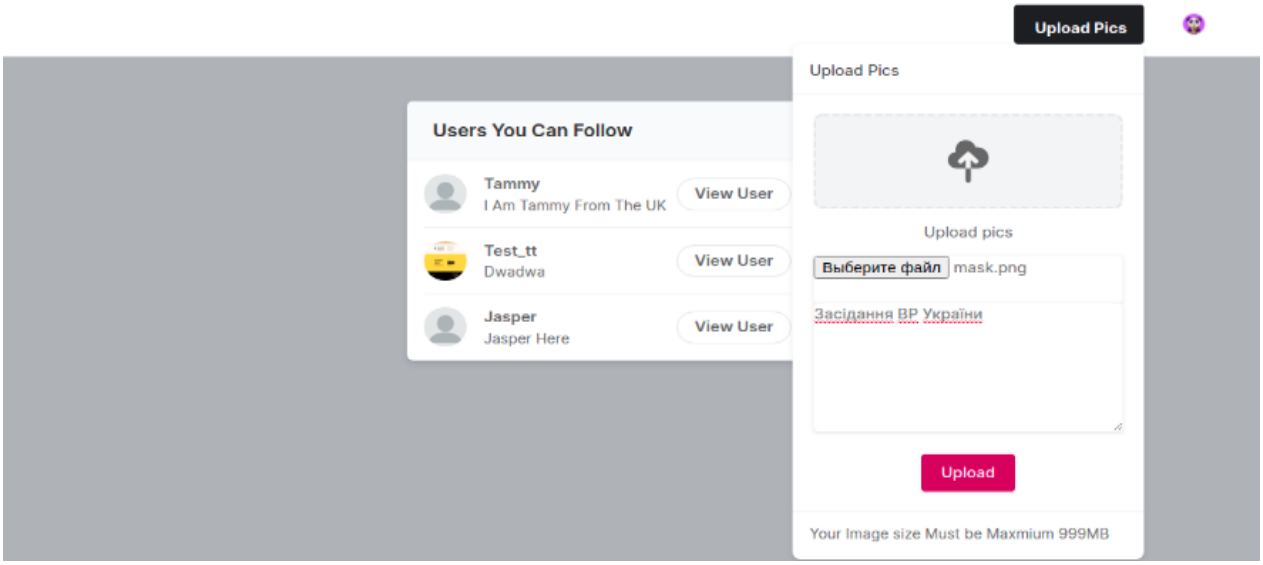

Рисунок 3.20 – Створення посту

Після введення даних, натискаємо на кнопку «Upload». Пост можна подивитися в профілі користувача, також його будуть бачити інші користувачі, які підписані на Ваші оновлення (рис. 3.21).

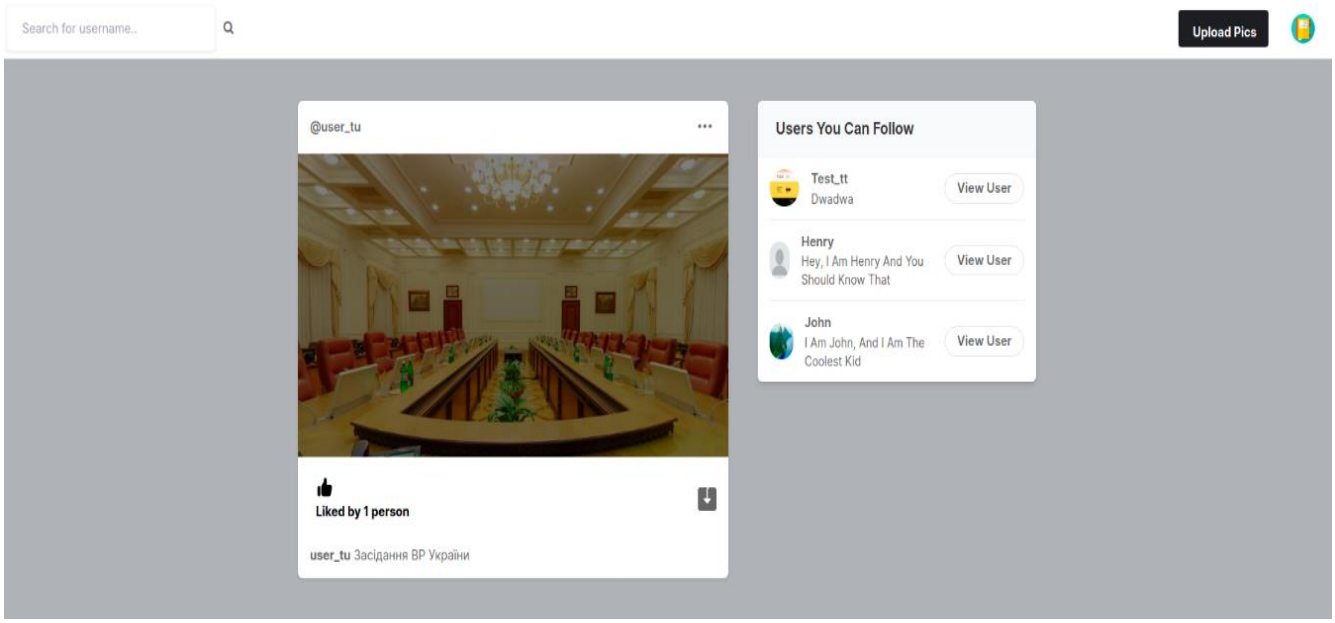

Рисунок 3.21 – Створений пост у стрічці іншого користувача

ВИСНОВКИ

В результаті роботи було написано технічне завдання на розробку вебсайту соціальної мережі. Для створення цієї системи було обрано фреймворк Django як зі сторони клієнта, так і зі сторони сервера, за його широкі можливості у сфері створення подібних систем.

У відповідності з метою кваліфікаційної роботи був розроблен вебсайт соціальної мережі із застосуванням наступних технологій:

- Django для реалізації клієнтської і серверної частини;
- SQLite для зберігання даних.

У відповідності з поставленими задачами були виконані наступні етапи створення системи:

- сформовані вимоги до системи (функціональні та нефункціональні (інтерфейс, кросбраузерність, безпека, продуктивність)), а також проведено огляд предметної області та інструментів розробки;
- спроєктована та побудована структура системи (побудовані діаграми прецедентів, діяльності, послідовності та розгортання; надано детальний опис прецедентів);
- реалізовано вебсайт соціальної мережі (наведена інструкція по створенню компонентів системи, надано керівництво користувача та структура проєкту);
- протестована робота системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Dinder M. Becoming an Enterprise Django Developer: Discover best practices, tooling, and solutions for writing and organizing Django applications in production. Birmingham : Packt Publishing, 2022. 526 p.

2. Django Developer Documentation. URL: https://docs.djangoproject.com/ (дата звернення: 11.03.2023).

3. Matthes E. Python Crash Course, 3rd Edition: A Hands-On, Project-Based Introduction to Programming. No Starch Press, 2023. 552 p.

4. Mele A. Django 4 By Example: Build powerful and reliable Python web applications from scratch. Birmingham : Packt Publishing, 2022. 766 p.

5. Ramalho L. Fluent Python: Clear, Concise, and Effective Programming. Sebastopol : O'Reilly Media, 2022. 1012 p.

6. Robbins P. Python Programming for Beginners: The Complete Guide to Mastering Python in 7 Days with Hands-On Exercises – Top Secret Coding Tips to Get an Unfair Advantage and Land Your Dream Job! : Independently published, 2023. 114 p.

7. Shaw B., Badhwar S., Bird A., Bharath C., Guest C. Web Development with Django: Learn to build modern web applications with a Python-based framework. Birmingham : Packt Publishing, 2021. 826 p.

8. Siahaan V., Hasiholan S. R. SQLITE FOR DATA ANALYSIS AND VISUALIZATION WITH PYTHON GUI : Independently published, 2022. 412 p.

9. SQLite Developer Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 22.03.2023).

10. Vincent W. S. Django for Beginners: Build websites with Python and Django. WelcomeToCode, 2020. 221 p.

ДОДАТОК А

Моделі

```
from django.db import models
from django.contrib.auth import get_user_model
import uuid
from datetime import datetime

User = get_user_model()

# Create your models here.
class Profile(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    id_user = models.IntegerField()
    bio = models.TextField(blank=True)
    profileimg = models.ImageField(upload_to='profile_images', default='blank-profile-picture.png')
    location = models.CharField(max_length=100, blank=True)

    def __str__(self):
        return self.user.username

class Post(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4)
    user = models.CharField(max_length=100)
    image = models.ImageField(upload_to='post_images')
    caption = models.TextField()
    created_at = models.DateTimeField(default=datetime.now)
    no_of_likes = models.IntegerField(default=0)

    def __str__(self):
        return self.user

class LikePost(models.Model):
    post_id = models.CharField(max_length=500)
    username = models.CharField(max_length=100)
```

```
    def __str__(self):
        return self.username

class FollowersCount(models.Model):
    follower = models.CharField(max_length=100)
    user = models.CharField(max_length=100)

    def __str__(self):
        return self.user
```

ДОДАТОК Б

Представлення

```
from django.shortcuts import render, redirect
from django.contrib.auth.models import User, auth
from django.contrib import messages
from django.http import HttpResponse
from django.contrib.auth.decorators import login_required
from .models import Profile, Post, LikePost, FollowersCount
from itertools import chain
import random

# Create your views here.

@login_required(login_url='signin')
def index(request):
    user_object = User.objects.get(username=request.user.username)
    user_profile = Profile.objects.get(user=user_object)

    user_following_list = []
    feed = []

    user_following = FollowersCount.objects.filter(follower=request.user.username)

    for users in user_following:
        user_following_list.append(users.user)

    for usernames in user_following_list:
        feed_lists = Post.objects.filter(user=usernames)
        feed.append(feed_lists)

    feed_list = list(chain(*feed))

    # user suggestion starts
    all_users = User.objects.all()
    user_following_all = []
```

```
    for user in user_following:
        user_list = User.objects.get(username=user.user)
        user_following_all.append(user_list)

    new_suggestions_list = [x for x in list(all_users) if (x not in list(user_following_all))]
    current_user = User.objects.filter(username=request.user.username)
    final_suggestions_list = [x for x in list(new_suggestions_list) if ( x not in list(current_user))]
    random.shuffle(final_suggestions_list)

    username_profile = []
    username_profile_list = []

    for users in final_suggestions_list:
        username_profile.append(users.id)

    for ids in username_profile:
        profile_lists = Profile.objects.filter(id_user=ids)
        username_profile_list.append(profile_lists)

    suggestions_username_profile_list = list(chain(*username_profile_list))


    return render(request, 'index.html', {'user_profile': user_profile, 'posts':feed_list, 'suggestions_username_profile_list': suggestions_username_profile_list[:4]})

@login_required(login_url='signin')
def upload(request):

    if request.method == 'POST':
        user = request.user.username
        image = request.FILES.get('image_upload')
        caption = request.POST['caption']

        new_post = Post.objects.create(user=user, image=image, caption=caption)
        new_post.save()

        return redirect('/')
```

```
    else:
        return redirect('/')

@login_required(login_url='signin')
def search(request):
    user_object = User.objects.get(username=request.user.username)
    user_profile = Profile.objects.get(user=user_object)

    if request.method == 'POST':
        username = request.POST['username']
        username_object = User.objects.filter(username__icontains=username)

        username_profile = []
        username_profile_list = []

        for users in username_object:
            username_profile.append(users.id)

        for ids in username_profile:
            profile_lists = Profile.objects.filter(id_user=ids)
            username_profile_list.append(profile_lists)

        username_profile_list = list(chain(*username_profile_list))
    return render(request, 'search.html', {'user_profile': user_profile, 'username_profile_list': username_profile_list})

@login_required(login_url='signin')
def like_post(request):
    username = request.user.username
    post_id = request.GET.get('post_id')

    post = Post.objects.get(id=post_id)

    like_filter = LikePost.objects.filter(post_id=post_id, username=username).first()

    if like_filter == None:
        new_like = LikePost.objects.create(post_id=post_id, username=username)
        new_like.save()
```

```
            post.no_of_likes = post.no_of_likes+1
            post.save()
            return redirect('/')
        else:
            like_filter.delete()
            post.no_of_likes = post.no_of_likes-1
            post.save()
            return redirect('/')

@login_required(login_url='signin')
def profile(request, pk):
    user_object = User.objects.get(username=pk)
    user_profile = Profile.objects.get(user=user_object)
    user_posts = Post.objects.filter(user=pk)
    user_post_length = len(user_posts)

    follower = request.user.username
    user = pk

    if FollowersCount.objects.filter(follower=follower, user=user).first():
        button_text = 'Unfollow'
    else:
        button_text = 'Follow'

    user_followers = len(FollowersCount.objects.filter(user=pk))
    user_following = len(FollowersCount.objects.filter(follower=pk))

    context = {
        'user_object': user_object,
        'user_profile': user_profile,
        'user_posts': user_posts,
        'user_post_length': user_post_length,
        'button_text': button_text,
        'user_followers': user_followers,
        'user_following': user_following,
    }
    return render(request, 'profile.html', context)
```

```
@login_required(login_url='signin')
def follow(request):
    if request.method == 'POST':
        follower = request.POST['follower']
        user = request.POST['user']

        if FollowersCount.objects.filter(follower=follower, user=user).first():
            delete_follower = FollowersCount.objects.get(follower=follower, user=user)
            delete_follower.delete()
            return redirect('/profile/'+user)
        else:
            new_follower = FollowersCount.objects.create(follower=follower, user=user)
            new_follower.save()
            return redirect('/profile/'+user)
    else:
        return redirect('/')

@login_required(login_url='signin')
def settings(request):
    user_profile = Profile.objects.get(user=request.user)

    if request.method == 'POST':

        if request.FILES.get('image') == None:
            image = user_profile.profileimg
            bio = request.POST['bio']
            location = request.POST['location']

            user_profile.profileimg = image
            user_profile.bio = bio
            user_profile.location = location
            user_profile.save()
        if request.FILES.get('image') != None:
            image = request.FILES.get('image')
            bio = request.POST['bio']
            location = request.POST['location']

            user_profile.profileimg = image
```

```
            user_profile.bio = bio
            user_profile.location = location
            user_profile.save()

        return redirect('settings')
    return render(request, 'setting.html', {'user_profile': user_profile})

def signup(request):

    if request.method == 'POST':
        username = request.POST['username']
        email = request.POST['email']
        password = request.POST['password']
        password2 = request.POST['password2']

        if password == password2:
            if User.objects.filter(email=email).exists():
                messages.info(request, 'Email Taken')
                return redirect('signup')
            elif User.objects.filter(username=username).exists():
                messages.info(request, 'Username Taken')
                return redirect('signup')
            else:
                user = User.objects.create_user(username=username, email=email, password=password)
                user.save()

                #log user in and redirect to settings page
                user_login = auth.authenticate(username=username, password=password)
                auth.login(request, user_login)

                #create a Profile object for the new user
                user_model = User.objects.get(username=username)
                new_profile = Profile.objects.create(user=user_model, id_user=user_model.id)
                new_profile.save()
                return redirect('settings')
        else:
            messages.info(request, 'Password Not Matching')
            return redirect('signup')
```

```
    else:
        return render(request, 'signup.html')

def signin(request):

    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']

        user = auth.authenticate(username=username, password=password)

        if user is not None:
            auth.login(request, user)
            return redirect('/')
        else:
            messages.info(request, 'Credentials Invalid')
            return redirect('signin')

    else:
        return render(request, 'signin.html')

@login_required(login_url='signin')
def logout(request):
    auth.logout(request)
    return redirect('signin')
```

ДОДАТОК В

Шаблони

```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="{% static ‘assets/images/favicon.png’ %}" rel="icon" type="image/png">
    <title>Sign In - Social Book</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="{% static ‘assets/css/icons.css’ %}">
    <link rel="stylesheet" href="{% static ‘assets/css/uikit.css’ %}">
    <link rel="stylesheet" href="{% static ‘assets/css/style.css’ %}">
    <link rel="stylesheet" href="{% static ‘assets/css/tailwind.css’ %}">

</head>

<body class="bg-gray-100">


    <div id="wrapper" class="flex flex-col justify-between h-screen">

        <!-- header-->
        <div class="bg-white py-4 shadow dark:bg-gray-800">
            <div class="max-w-6xl mx-auto">


                <div class="flex items-center lg:justify-between justify-around">

                    <a href="trending.html">
                        <b><h1 style="font-size: 1.5rem;">Social Book</h1></b>
                    </a>
```

```
            <div class="capitalize flex font-semibold hidden lg:block my-2 space-x-3 text-center text-sm">

                <a href="/signup" class="bg-pink-500 pink-500 px-6 py-3 rounded-md shadow text-
white">Register</a>
            </div>

        </div>
    </div>
</div>

<!-- Content-->
<div>
    <div class="lg:p-12 max-w-md max-w-xl lg:my-0 my-12 mx-auto p-6 space-y-">
        <h1 class="lg:text-3xl text-xl font-semibold  mb-6"> Log in</h1>


        <div>
            <style>
                h5{
                    color: red;
                }
            </style>
            {% for message in messages %}
            <h5>{{message}}</h5>
            {% endfor %}
        </div>


        <form action="" method="POST">
            {% csrf_token %}
            <input type="text" name="username" placeholder="Username" class="bg-gray-200 mb-2
shadow-none dark:bg-gray-800" style="border: 1px solid #d3d5d8 !important;">
            <input type="password" name="password" placeholder="Password" class="bg-gray-200 mb-2
shadow-none dark:bg-gray-800" style="border: 1px solid #d3d5d8 !important;">
            <!-- <div class="flex justify-between my-4">
                <div class="checkbox">
                    <input type="checkbox" id="chekcbox1" checked>
```

```
                    <label for="chekcbox1"><span class="checkbox-icon"></span>Remember Me</label>
                </div>
                <a href="#"> Forgot Your Password? </a>
            </div> -->
            <button type="submit" class="bg-gradient-to-br from-pink-500 py-3 rounded-md text-white
text-xl to-red-400 w-full">Login</button>
            <div class="text-center mt-5 space-x-2">
                <p class="text-base"> Not registered? <a href="/signup" class=""> Create a account </a></p>
            </div>
        </form>


      </div>
    </div>

    <!-- Footer -->

    <div class="lg:mb-5 py-3 uk-link-reset">
      <div class="flex flex-col items-center justify-between lg:flex-row max-w-6xl mx-auto lg:space-y-0
space-y-3">
        <div class="flex space-x-2 text-gray-700 uppercase">
          <a href="#"> About</a>
          <a href="#"> Help</a>
          <a href="#"> Terms</a>
          <a href="#"> Privacy</a>
        </div>
        <p class="capitalize"> © copyright 2020 by socol</p>
      </div>
    </div>

  </div>
  <script src="{% static ‘assets/js/tippy.all.min.js’ %}"></script>
  <script src="{% static ‘assets/js/jquery-3.3.1.min.js’ %}"></script>
  <script src="{% static ‘assets/js/uikit.js’ %}"></script>
  <script src="{% static ‘assets/js/simplebar.js’ %}"></script>
  <script src="{% static ‘assets/js/custom.js’ %}"></script>
</body>
</html>
```

ДОДАТОК Г

URL-маршрутизація

```
from django.urls import path
from . import views

urlpatterns = [
    path(‘’, views.index, name=‘index’),
    path(‘settings’, views.settings, name=‘settings’),
    path(‘upload’, views.upload, name=‘upload’),
    path(‘follow’, views.follow, name=‘follow’),
    path(‘search’, views.search, name=‘search’),
    path(‘profile/<str:pk>‘, views.profile, name=‘profile’),
    path(‘like-post’, views.like_post, name=‘like-post’),
    path(‘signup’, views.signup, name=‘signup’),
    path(‘signin’, views.signin, name=‘signin’),
    path(‘logout’, views.logout, name=‘logout’),
]
```