Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка соціальної мережі Connectify засобами PHP Laravel, MySQL, Bootstrap, JavaScript

Виконав: студент IV курсу, групи СНс-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Лісовий М.В.
(підпис) (прізвище та ініціали)

Керівник _____ Млинко Б.Б.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«____» _червня_ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Лісовому Максиму Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка соціальної мережі Connectify засобами PHP Laravel, MySQL, Bootstrap, JavaScript

Керівник роботи Млинко Богдана Богданівна, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*29*» *квітня* 2024 року № *4/7-472*

2. Термін подання студентом завершеної роботи 24 червня 2024р.

3. Вихідні дані до роботи Літературні та інтернет джерела інформації щодо розробки соціальної мережі Connectify засобами PHP Laravel, MySQL, Bootstrap, JavaScript

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та постановка завдання розробки соціальної мережі Connectify. 1.1 Аналіз соціальних мереж та їх функціональності. 1.2 Постановка завдання та визначення вимог до соціальної мережі Connectify. 1.3 Пошук актантів та варіантів використання соціальної мережі Connectify. 1.4 Висновок до першого розділу. 2. Проєктування соціальної мережі Connectify. 2.1 Вибір інформаційно-технологічного стеку для розробки соціальної мережі Connectify. 2.2 Моделювання архітектури соціальної мережі Connectify. 2.3 Розробка моделей даних соціальної мережі Connectify. 2.3.1 Перелік інформаційних сутностей соціальної мережі Connectify. 2.3.2 Проєктування концептуальної моделі даних соціальної мережі Connectify. 2.4 Висновок до другого розділу. 3. Розробка, валідація та тестування соціальної мережі Connectify. 3.1 Розробка соціальної мережі Connectify. 3.1.1 Розробка інтерфейсу користувача соціальної мережі Connectify. 3.1.2 Реалізація функціональних можливостей соціальної мережі Connectify. 3.2 Валідація та тестування соціальної мережі Connectify. 3.2.1 Валідація даних соціальної мережі. 3.2.2 Тестування функціональних можливостей соціальної мережі. 3.3 Висновок до третього розділу. 4. Безпека життєдіяльності, основи охорони праці. Висновки. . Перелік джерел. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульна сторінка. 2. Актуальність обраної теми. 3. Мета, об'єкт та предмет дослідження 4. Завдання розробки соціальної мережі Connectify. 5. Аналіз предметної області. 6. Актанти соціальної мережі Connectify. 6. Інформаційно-технологічний стек. 7. Моделювання архітектури. 8. ER-діаграма. 9. Інтерфейс користувача соціальної мережі.Connectify 11. Функціональні можливості соціальної мережі Connectify. 12. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С., доцент кафедри МТ		

7. Дата видачі завдання 29 січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	30.01.2024	*Виконано*
2.	Підбір джерел про розробку веб-застосунків засобами PHP Laravel, MySQL, Bootstrap, JavaScript	31.01.2024-03.02.2024	*Виконано*
3.	Опрацювання джерел щодо розробки соціальної мережі Connectify засобами PHP Laravel, MySQL, Bootstrap, JavaScript	04.02.2024-06.02.2024	*Виконано*
4.	Виконання дослідження щодо розробки соціальної мережі Connectify засобами PHP Laravel, MySQL, Bootstrap, JavaScript. Розроблення соціальної мережі Connectify засобами PHP Laravel, MySQL, Bootstrap, JavaScript	07.02.2024-11.02.2024	*Виконано*
5.	Оформлення розділу «Аналіз предметної області та постановка завдання розробки соціальної мережі Connectify»		
6.	Оформлення розділу «Проєктування соціальної мережі Connectify»		
7.	Оформлення розділу «Розробка, валідація та тестування соціальної мережі Connectify»		
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»		
9.	Виконання завдання до підрозділу «Основи охорони праці»		
10.	Оформлення кваліфікаційної роботи		
11.	Нормоконтроль		
12.	Перевірка на плагіат		
13.	Попередній захист кваліфікаційної роботи	21.06.2024	
14.	Захист кваліфікаційної роботи	29.06.2024	

Студент _____ Лісовий М.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Млинко Б.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка соціальної мережі Connectify засобами PHP Laravel, MySQL, Bootstrap, JavaScript // Кваліфікаційна робота освітнього рівня «Бакалавр» // Лісовий Максим Володимирович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс-42 // Тернопіль, 2024 // С. 58, рис. – 24, табл. – 7, презент. – 12, додат. – 4, бібліогр. – 41.

Ключові слова: соціальна мережа, бази даних, модель, веб-сторінка, Laravel, PHP, Bootstrap, MySQL

Кваліфікаційна робота присвячена дослідженню розробки соціальної мережі Connectify засобами PHP Laravel, MySQL, Bootstrap, JavaScript. В першому розділі кваліфікаційної роботи проаналізовано переваги та недоліки існуючих сучасних соціальних мереж. Висвітлено вимоги та постановку завдання для розробки соціальної мережі Connectify. Розглянуто пошук актантів та можливі варіанти використання соціальної мережі.

В другому розділі кваліфікаційної роботи досліджено вибір інформаційно-технологічного стеку для розробки соціальної мережі Connectify, включаючи PHP Laravel, MySQL, Bootstrap та JavaScript. Подано моделювання архітектури системи та розроблено концептуальну модель даних, перелік інформаційних сутностей.

В третьому розділі кваліфікаційної роботи описано розробку інтерфейсу користувача та функціональних можливостей соціальної мережі Connectify. Проаналізовано методи валідації даних. Проведено тестування функціональних можливостей для забезпечення стабільної роботи системи.

Об'єкт дослідження: процес розробки соціальної мережі Connectify.

Предмет дослідження: засоби і методи розробки соціальних мереж із використанням PHP Laravel, MySQL, Bootstrap та JavaScript.

ANNOTATION

Development of the Connectify Social Network Using PHP Laravel, MySQL, Bootstrap, JavaScript // Qualification work of the educational level «Bachelor» // Lisovyi Maksym // Ternopil Ivan Puluj National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SNs-42 // Ternopil, 2024 // P. 58, fig. – 24, tabl. – 7, present. – 12, annexes. – 4, references – 41.

Keywords: social network, databases, model, web page, Laravel, PHP, Bootstrap, MySQL

The qualification work is dedicated to the development of the Connectify social network using PHP Laravel, MySQL, Bootstrap, JavaScript.

The first chapter of the qualification work analyzes the advantages and disadvantages of existing modern social networks. The requirements and task setting for the development of the Connectify social network are highlighted. The search for actors and possible use cases for the social network are considered.

The second chapter of the qualification work investigates the choice of the information technology stack for the development of the Connectify social network, including PHP Laravel, MySQL, Bootstrap, and JavaScript. The system architecture modeling and the conceptual data model, as well as the list of informational entities, are presented.

The third chapter of the qualification work describes the development of the user interface and functional capabilities of the Connectify social network. The methods of data validation are analyzed. The functional capabilities testing to ensure the stable operation of the system is conducted.

Object of research: the process of developing the Connectify social network.

Subject of research: the means and methods of developing social networks using PHP Laravel, MySQL, Bootstrap, and JavaScript.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

HTML (HyperText Markup Language) – Гіпертекстова мова розмітки.

CSS (Cascading Style Sheets) – Каскадні таблиці стилів.

API (Application Programming Interface) – Інтерфейс програмування додатків.

PHP (Hypertext Preprocessor) – Препроцесор гіпертекстового програмування.

ORM (Object-Relational Mapping) – Відображення об'єктно-реляційної моделі.

Bootstrap – Фреймворк для швидкого розроблення інтерфейсів користувача.

Laravel – Фреймворк для веб-розробки з високою якістю кодування.

Blade – Механізм шаблонів для фреймворка Laravel.

SQL (Structured Query Language) – Мова структурованих запитів.

URL (Uniform Resource Locator) – Рівномірний локатор ресурсів.

HTTP (Hypertext Transfer Protocol) – Протокол передачі гіпертексту.

ЗМІСТ

ВСТУП

Соціальні мережі є одним з найпопулярніших та найвпливовіших засобів комунікації в сучасному світі, забезпечуючи людям можливість обмінюватися інформацією, спілкуватися, співпрацювати та вирішувати різні проблеми [1]. Крім того, соціальні мережі створюють нові можливості для бізнесу, освіти, науки, культури та громадського життя. Враховуючи це, важливо постійно розробляти та вдосконалювати соціальні мережі, які б відповідали потребам та інтересам користувачів, а також забезпечували безпеку та якість послуг.

Метою даної курсової роботи є розробка соціальної мережі Connectify для спілкування, використовуючи PHP Laravel, MySQL, Bootstrap, JavaScript. Для досягнення цієї мети необхідно виконати наступні завдання:

– Проаналізувати предметну область, оглянувши сучасні соціальні мережі.

– Сформулювати перелік вимог до соціальної мережі, ґрунтуючись на аналізі потреб та очікувань потенційних користувачів.

– Провести пошук актантів та варіантів використання соціальної мережі Connectify, використовуючи методологію UML.

– Спроєктувати структуру даних соціальної мережі, архітектуру системи та розробити концептуальну модель сутностей.

– Реалізувати функціональні можливості соціальної мережі за допомогою сучасних технологій розробки веб-додатків.

– Провести валідацію та тестування соціальної мережі з метою перевірки її функціонування, коректності даних та задоволення користувацьких сценаріїв.

Практичне значення одержаних результатів полягає в тому, що розроблена соціальна мережа може бути використана для сприяння комунікації, навчання, розваги та саморозвитку користувачів. Крім того, розроблена соціальна мережа Connectify може стати відмінним прикладом або навіть фундаментом для подальших досліджень та розробок у цьому напрямку.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ РОЗРОБКИ СОЦІАЛЬНОЇ МЕРЕЖІ CONNECTIFY

1.1 Аналіз соціальних мереж та їх функціональності

Сучасне суспільство неможливо уявити без соціальних мереж, які стали невід'ємною складовою життя людей. Вони забезпечують користувачам широкі можливості взаємодії, обміну інформацією та формування спільнот.

Соціальні мережі перетворилися на глобальну платформу, де мільйони людей з'єднані в одну велику віртуальну сітку, яка сповнена життя, ідей та нових можливостей [2].

Завдяки соціальним мережам люди можливість знайти нових друзів, обговорювати актуальні теми, ділитися своїми досягненнями та створювати власний цифровий імідж, вони відкривають шлях до світу безмежних можливостей.

Для розробки соціальної мережі Connectify розглянуто та проаналізовано популярні сучасні соціальні мережі, такі як Facebook, Instagram, X, LinkedIn та TikTok.

Facebook є найбільшою соціальною мережею у світі з мільярдами активних користувачів [3]. Користувачі Facebook можуть створювати особисті профілі, вказуючи основну інформацію про себе, завантажувати фотографії та відео, реалізована можливість додавати друзів, підписуватися на оновлення інших користувачів.

Головна сторінка відображає оновлення від друзів, сторінок та груп, на які підписаний користувач. Також є можливість створення спільнот за інтересами, публікації контенту в них та інтегрована система обміну повідомленнями для приватного спілкування. На рисунку 1.1 зображено частину сторінки користувача Facebook.

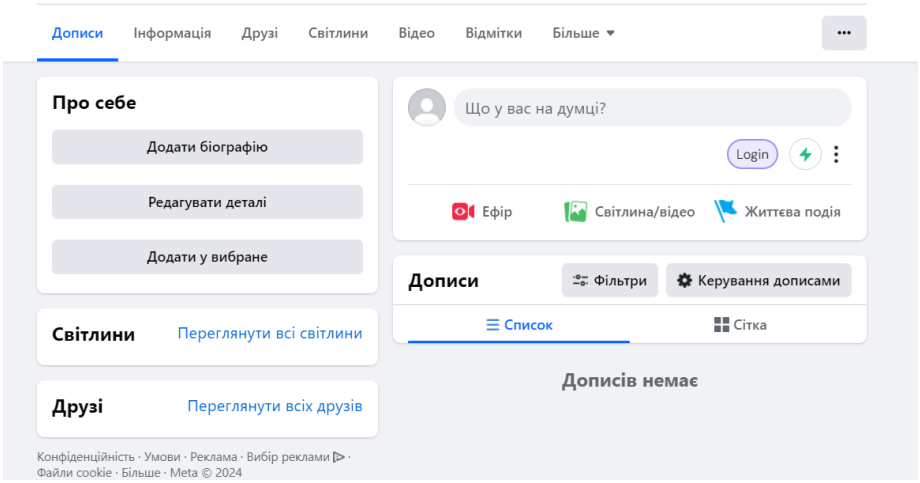

Рисунок 1.1 – Частина сторінки користувача Facebook

Instagram як і Facebook є власністю компанії Meta, тому можна зв’язати профілі цих соціальних мереж та легко дублювати контент, що публікується в них. Не зважаючи на це, соціальні мережі мають певні відмінності.

Instagram фокусується на візуальному контенті, надаючи можливість користувачам ділитися фотографіями та відео, які є основним контентом платформи. Користувачі можуть додавати фільтри, підписи та хештеги. Найголовнішим функціоналом Instagram є історії – тимчасові публікації, які зникають через 24 години, вони ідеально підходять для щоденних оновлень [4]. Існує можливість слідкувати за цікавими користувачами та публікувати контент для різних людей, додавши їх в список «Близькі друзі» або обмеживши інших підписників. На рисунку 1.2 зображено сторінку користувача Instagram.

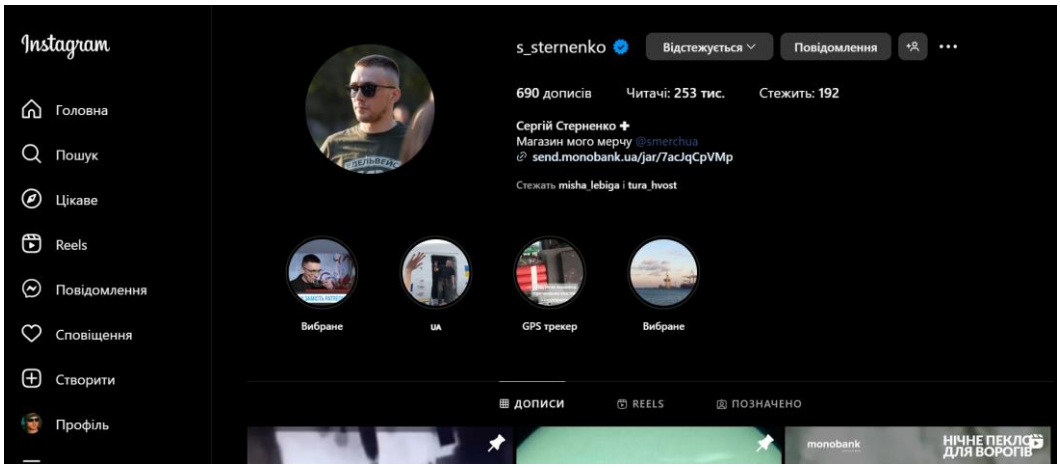

Рисунок 1.2 – Сторінка користувача Instagram

Також варто зауважити що є можливість трансляції відео в реальному часі для своїх підписників та система Direct для приватних повідомлень між користувачами.

X – це платформа для мікроблогінгу, яка надає користувачам можливість публікувати короткі повідомлення, відомі як твіти, що складаються з не більше ніж 280 символів [5]. Ця обмежена кількість символів стимулює лаконічність та концентрацію на головній думці або ідеї.

Крім тексту, X дозволяє користувачам додавати фотографії, відео та гіфки до своїх твітів, що зображено на рисунку 1.3. Це дозволяє зміцнити враження від повідомлення та надати йому візуальну складову.

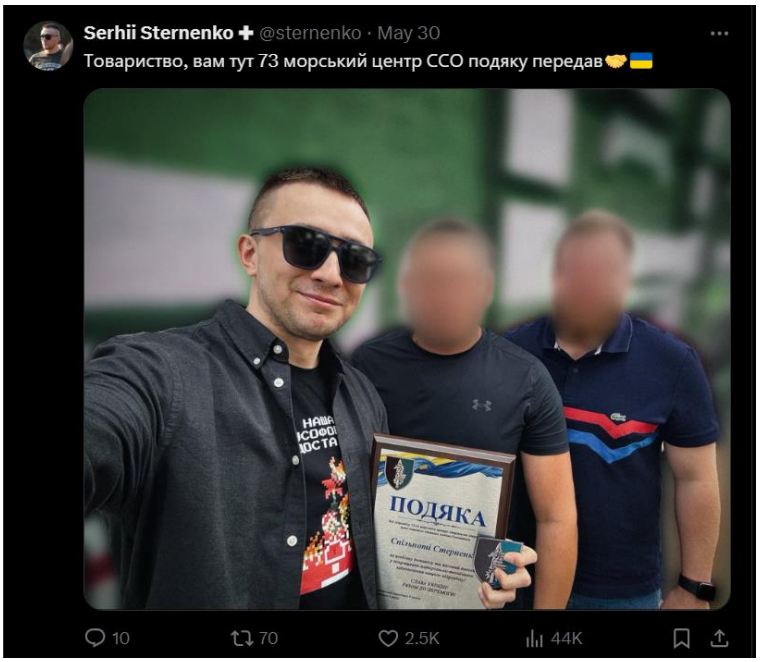

Рисунок 1.3 – Приклад твіту з зображенням в X

Однією з ключових можливостей X є можливість ретвітнути твіти інших користувачів. Це дозволяє поширювати цікавий або важливий контент, дати йому більше видимості та залучити увагу більшої аудиторії. Крім того, користувачі можуть використовувати хештеги для категоризації своїх твітів та полегшення пошуку контенту на конкретну тему. Це допомагає знаходити пов'язані твіти та приєднуватися до розмов про певну тему.

LinkedIn є професійною соціальною мережею, спрямованою на з'єднання людей у сфері бізнесу, розвиток кар'єри та встановлення професійних контактів. Ця платформа надає можливість користувачам створювати свої профілі, де вони можуть додати свій професійний досвід, освіту, навички та досягнення. Основна мета LinkedIn полягає у сприянні пошуку роботи та найму співробітників.

Користувачі можуть створювати детальні профілі, де вони можуть вказати свої попередні місця роботи, посади, відповідальні обов'язки та досягнення. Приклад такого профілю наведено на рисунку 1.4. Це надає потенційним роботодавцям можливість знайти відповідних кандидатів та зв'язатися з ними.

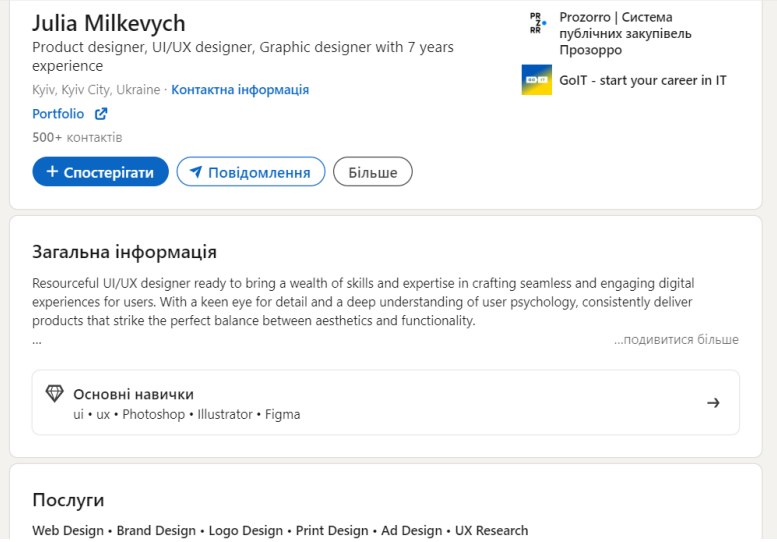

Рисунок 1.4 – Профіль користувача в LinkedIn

Користувачі ж можуть активно шукати вакансії, підписуватися на компанії та спеціалістів, які цікавлять їх, та навіть отримувати рекомендації та поради щодо кар'єрного розвитку. Загалом, LinkedIn є незамінним інструментом для професіоналів, які бажають розвиватися в своїй кар'єрі, будувати бізнес-контакти та залишатися в курсі останніх тенденцій у своїй галузі.

TikTok – це динамічна платформа, яка запропонувала світові новий спосіб створення та обміну короткими відео. Ця соціальна мережа перетворилася на справжню креативну сцену, де користувачі з усього світу мають можливість виразити свою унікальність та талант у форматі відео.

Основною особливістю TikTok є короткі відео, які зазвичай тривають від 15 до 60 секунд. Цей формат дозволяє швидко і ефективно передати повідомлення, розповісти історію або продемонструвати щось. Користувачі можуть додавати музику та візуальні ефекти до своїх відео, створюючи захопливий та вражаючий контент, яким наповнюють власний профіль, приклад якого наведено на рисунку 1.5.

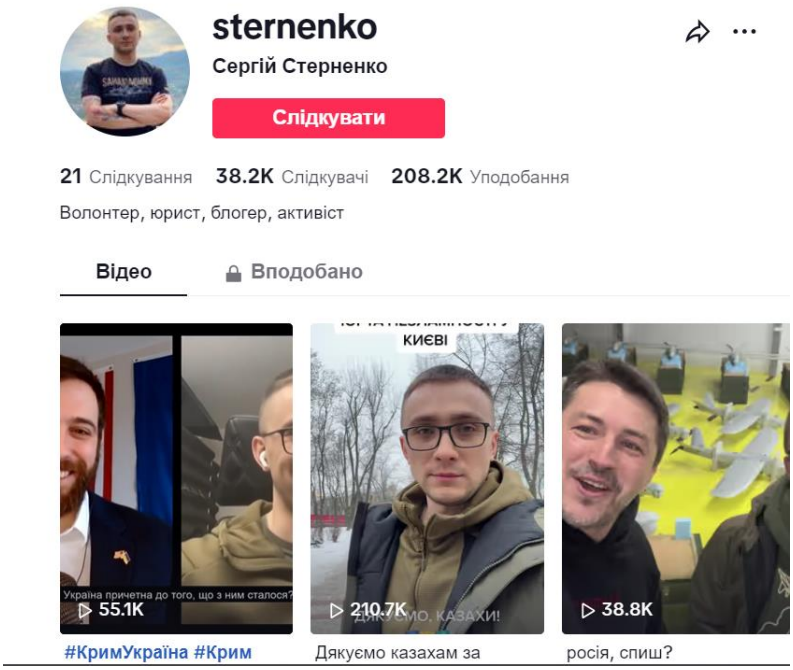

Рисунок 1.5 – Профіль користувача TikTok

Також TikTok відомий своїми трендами, що поширюються в мережі і залучають велику кількість користувачів. Завдяки своїм інноваціям та привабливій формі контенту, TikTok став однією з найпопулярніших соціальних мереж у світі [6].

Для розробки соціальної мережі Connectify проаналізовану інформацію щодо інших соціальних мереж систематизовано за допомогою таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз сучасних соціальних мереж

Соціальна мережа	**Основні функції**	**Тип основного контенту**	**Цільова аудиторія**
Facebook	Профілі, друзі, групи, сторінки	Текст, фото, відео	Широка аудиторія
Instagram	Фотографії, історії, прямі ефіри	Фото, відео	Молодь, підлітки
Twitter	Твіти, ретвіти, хештеги	Короткі повідомлення	Професіонали, молодь
LinkedIn	Профілі, зв'язки, вакансії	Текст, фото	Професіонали, бізнесмени
TikTok	Короткі відео, музика, ефекти	Відео	Підлітки, молодь

Кожна мережа має свої унікальні функції, це дозволяє користувачам вибирати найбільш підходящу для їх потреб платформу.

Основною причиною для створення соціальної мережі Connectify є забезпечення користувачів простим та інтуїтивно зрозумілим інтерфейсом, що дозволяє легко взаємодіяти в соціальній мережі. Користувачі сучасних соціальних мереж часто стикаються зі складними налаштуваннями та перевантаженими інтерфейсами, тому Connectify зосереджується на найважливіших функціях для щоденного користування, які не будуть включати в себе нічого лишнього. Крім того, наявність не складної адміністративної панелі спрощує управління платформою, забезпечуючи її безпеку та ефективність.

Враховуючи те, для створення Connectify використані сучасні технології розробки, вона має потенціал для подальшого розвитку, що дозволить додавати нові функції відповідно до потреб користувачів і технологічних тенденцій.

1.2 Постановка завдання та визначення вимог до соціальної мережі Connectify

Соціальна мережа Connectify розробляється з метою створення зручної та функціональної платформи для користувачів, які бажають спілкуватися, обмінюватися контентом та підтримувати зв'язки з іншими людьми. Основним завданням розробки полягає в наданні користувачам інструментів для створення та управління своїми профілями, додавання друзів, публікації статусів та фотографій, а також взаємодії з контентом інших користувачів. Для успішної розробки соціальної мережі визначено функціональні вимоги, які будуть впливати на функціональність та якість системи [7].

Користувачі повинні мати можливість зареєструватися на платформі, створивши обліковий запис за допомогою електронної пошти. Після реєстрації користувачі повинні мати можливість увійти в свій обліковий запис за допомогою своїх облікових даних та додати чи редагувати свою особисту інформацію, таку як ім'я, прізвище, місто, дата народження. Також важливо щоб була можливість завантаження та зміни зображення профілю користувача. Основна інформація профілю повинна відображатись для інших користувачів соціальної мережі.

Користувачі соціальної мережі повинні мати можливість пошуку інших користувачів за іменем та додавати їх в друзі, надіславши запит, а вони в свою чергу могли підтвердити або відхилити запит. Список друзів повинен відображатись на сторінці профілю користувача.

Користувачі повинні мати можливість публікувати текстові статуси та за потреби додавати фотографії до статусів. Статуси користувача повинні відображатись на його сторінці профілю та в стрічці для його друзів.

Користувачі повинні мати можливість прокоментувати та вподобати статуси своїх друзів.

На сторінці користувача повинна бути фотогалерея з фотографіями, завантаженими користувачем у статусах та можливість перегляду фотографій в повноекранному режимі.

Обов’язково повинна бути адміністративна панель, в якій користувачі з правами адміністратора матимуть можливість керувати іншими користувачами, тобто активовувати та деактивовувати їхні облікові записи та видаляти статуси чи коментарі, що порушують правила соціальної мережі Connectify.

Також визначено нефункціональні вимоги до системи [8]. Соціальна мережа повинна бути стабільною та працездатною у всьому своєму життєвому циклу, мати механізми для запобігання та усунення помилок та збоїв, повинна захищати конфіденційність та цілісність даних користувачів від несанкціонованого доступу за допомогою надійної аутентифікації та шифрування даних, забезпечувати швидке та відповідне виконання запитів користувачів.

Дизайн соціальної мережі Connectify повинен мати інтуїтивний та привабливий інтерфейс, який забезпечує легке та приємне користування, підтримувати різні платформи та пристрої, такі як комп'ютери, смартфони, планшети тощо.

Дотримання всіх вимог дозволить розробити соціальну мережу, що буде ефективним інструментом для забезпечення зв'язку та взаємодії між користувачами.

1.3 Пошук актантів та варіантів використання соціальної мережі Connectify

Важливим етапом розробки соціальної мережі є пошук актантів та варіантів використання, які допоможуть визначити основні сценарії поведінки користувачів та інших зацікавлених сторін у системі. Актанти – це ролі, які можуть брати участь у використанні системи або впливати на неї [9]. До актантів даної системи належать:

– Зареєстрований користувач – особа, яка має свій профіль в соціальній мережі та може користуватися її функціональними можливостями, а саме – публікувати статуси, взаємодіяти з іншими користувачами, редагувати свої особисті дані та інше.

– Гість – особа, яка не має зареєстрованого профіля в соціальній мережі, але може переглядати головну сторінку або зареєструвати новий профіль.

– Адміністратор – особа, що має зареєстрований профіль і має доступ до адміністративної панелі, може активовувати та деактивовувати інші облікові записи або видаляти статуси інших користувачів.

Варіанти використання – це описи послідовностей дій, які виконують актанти для досягнення певної мети у системі [10].

Базуючись на вимогах до соціальної мережі Connectify сформовано варіанти використання для актантів системи. Для актанта «Гість» варіанти використання наведено в таблиці 1.2.

Таблиця 1.2 – Опис варіантів використання для актанта «Гість» соціальної мережі Connectify

Актант	**Варіант використання**	**Формулювання**
Гість	Перегляд головної сторінки	Дозволяє перегляд головної сторінки соціальної мережі
	Реєстрація	Дозволяє створити новий обліковий запис

Як видно з таблиці 1.2, гість має дуже вузьке коло можливих варіантів використання. Так і повинно бути, адже всі інші можливості користувачу доступні лише після реєстрації та авторизації в соціальній мережі.

Для актанта «Адміністратор» варіанти використання наведено в таблиці 1.3, а для актанта «Зареєстрований користувач» – у таблиці А.1.

Таблиця 1.3 – Опис варіантів використання для актанта «Адміністратор» соціальної мережі Connectify

Актант	**Варіант використання**	**Формулювання**
Адміністратор	Перегляд адміністративної панелі	Дозволяє переглядати списки всіх користувачів, статусів та коментарів соціальної мережі
	Активація профілю	Дозволяє активувати неактивний профіль користувача соціальної мережі
	Деактивція профілю	Дозволяє деактивувати профіль користувача соціальної мережі
	Видалення статусів та коментарів	Дозволяє видаляти статуси та коментарі інших користувачів

На рисунку 1.6 зображено діаграму варіантів [11] використання розроблюваної соціальної мережі.

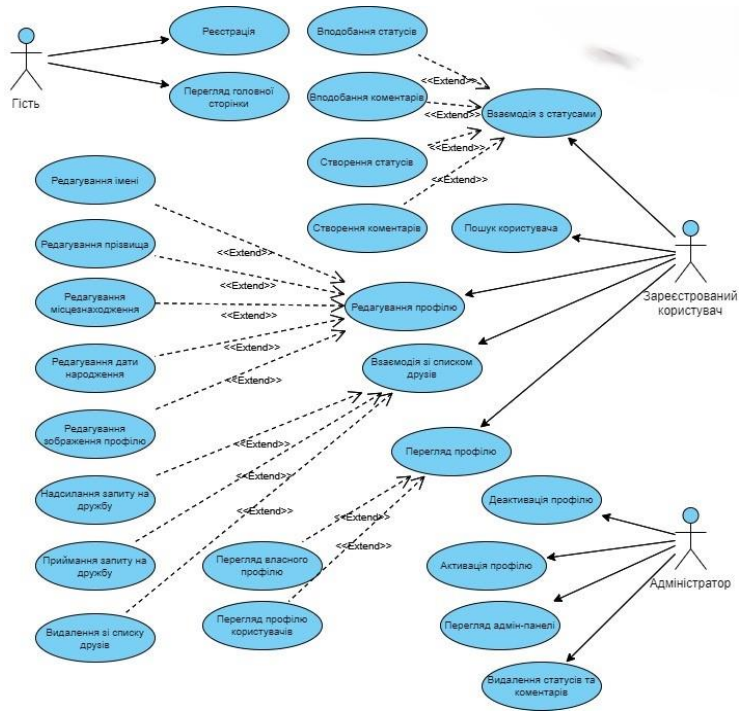

Рисунок 1.6 – Діаграма прецедентів соціальної мережі Connectify

Як видно з рисунку, зареєстрований користувач має найбільше варіантів варіантів використання, які включають в себе майже всі функціональні можливості розроблюваної системи.

1.4 Висновок до першого розділу

У першому розділі кваліфікаційної роботи бакалавра здійснено аналіз предметної області соціальних мереж, що дозволило визначити основні функції та можливості, які вони надають користувачам. Розглянуто приклади популярних соціальних мереж та визначено, які з функцій будуть інтегровані в соціальну мережу Connectify. Виконано постановку завдання, яке полягає в наданні користувачам інструментів для створення та управління своїми профілями, додавання друзів, публікації статусів та фотографій, а також взаємодії з контентом інших користувачів. Визначено основні вимоги до функціоналу соціальної мережі Connectify та основні актанти системи їх варіанти використання. Серед них актанти «Зареєстрований користувач», «Гість» та «Адміністратор». Це допомогло зрозуміти, хто з можливих користувачів буде використовувати функціональні можливості соціальної мережі та які саме.

Таким чином, аналіз предметної області та постановка завдання розробки соціальної мережі Connectify дозволили визначити ключові аспекти та забезпечили міцну основу для проєктування соціальної мережі.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СОЦІАЛЬНОЇ МЕРЕЖІ CONNECTIFY

2.1 Вибір інформаційно-технологічного стеку для розробки соціальної мережі Connectify

Розробка соціальної мережі Connectify вимагала ретельного підходу до вибору технологічного стека, який би забезпечив високу продуктивність, надійність, масштабованість та зручність розробки. Було обрано PHP Laravel, MySQL, Bootstrap та JavaScript, кожна з яких має свої переваги та специфіку.

Laravel, як один з найпопулярніших PHP-фреймворків, став основним вибором для серверної частини. Його популярність та активна підтримка спільноти забезпечують доступ до великої кількості ресурсів та навчальних матеріалів, що значно полегшує процес розробки [12]. Laravel дотримується архітектури MVC (Model-View-Controller), що сприяє чіткому розділенню логіки додатку, інтерфейсу користувача та управління даними. На рисунку 2.1 схематично зображено архітектуру MVC. Це робить код більш організованим та легким у підтримці, що є важливим для довготривалих проєктів [13].

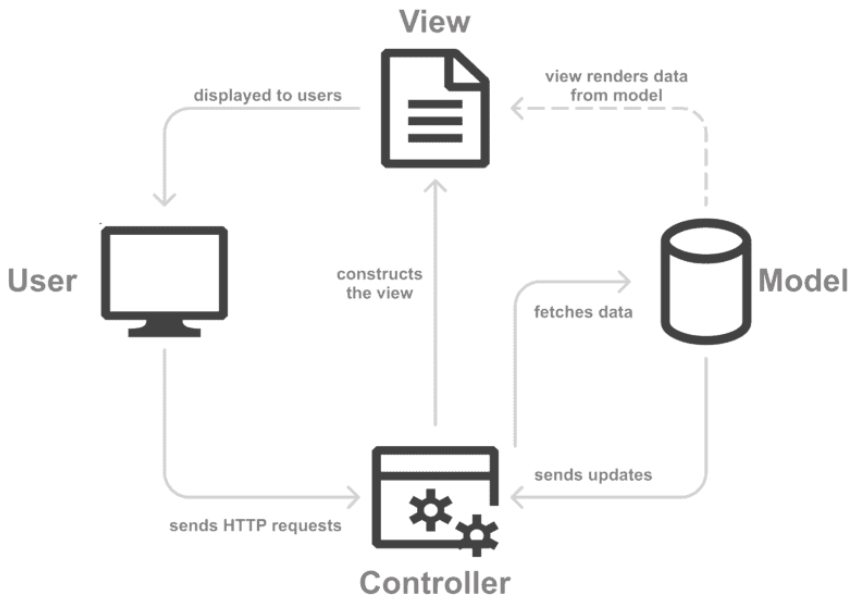

Рисунок 2.1 – MVC архітектура фреймворку Laravel

Однією з головних причин вибору Laravel є його висока продуктивність. Фреймворк забезпечує високий рівень абстракції над складними операціями, такими як маршрутизація, аутентифікація, управління сесіями, кешування та робота з базами даних. Це значно прискорює процес розробки та дозволяє зосередитися на бізнес-логіці додатку, а не на низькорівневих деталях.

Крім того, Laravel має елегантний та зрозумілий синтаксис, що полегшує написання та читання коду. Він пропонує безліч вбудованих функцій та інструментів, таких як Eloquent ORM для роботи з базами даних, Blade шаблонізатор для побудови інтерфейсів та Queue API для обробки фонових завдань [14]. Це робить його ідеальним вибором для розробки соціальної мережі, де потрібно обробляти велику кількість даних та забезпечувати швидкий доступ до них.

З точки зору безпеки, Laravel включає вбудовані механізми захисту від найбільш поширених веб-загроз, таких як SQL-ін'єкції, міжсайтовий скриптинг (XSS) та підробка міжсайтових запитів (CSRF). Це забезпечує високий рівень захисту даних користувачів та стабільну роботу додатку [15].

MySQL був обраний як основна база даних для зберігання даних. Цей вибір був обумовлений його надійністю та високою продуктивністю. MySQL є однією з найпопулярніших реляційних баз даних, здатною обробляти великі обсяги даних та забезпечувати швидкий доступ до них. Вона підтримує реплікацію та кластеризацію, що дозволяє легко масштабувати базу даних відповідно до зростаючих потреб додатку [16].

Окрім цього, MySQL має широке поширення і підтримується різними мовами програмування та фреймворками, включаючи PHP та Laravel. Це спрощує інтеграцію та управління даними, що є важливим для забезпечення ефективної роботи соціальної мережі. Відкритий код MySQL робить його доступним для широкого кола користувачів без необхідності платити за ліцензії, а інструменти управління, такі як phpMyAdmin та MySQL Workbench, полегшують адміністрування та налаштування.

Для побудови адаптивного інтерфейсу було обрано Bootstrap. Цей фреймворк є одним з найпопулярніших для розробки адаптивних веб-інтерфейсів, забезпечуючи набір готових компонентів та стилів, що дозволяє швидко створювати сучасні веб-додатки. Bootstrap дозволяє створювати адаптивні веб-сторінки, які коректно відображаються на різних пристроях та розмірах екранів, що є критично важливим для соціальної мережі. Bootstrap має зрозумілу документацію та простий у використанні синтаксис, що дозволяє швидко розпочати роботу з фреймворком. Завдяки вбудованим класам та компонентам, Bootstrap дозволяє створювати адаптивні інтерфейси без додаткових зусиль. Велика кількість готових компонентів, таких як навігаційні панелі, форми, модальні вікна, каруселі та інші, значно спрощує розробку інтерфейсу.

JavaScript це мова програмування для додавання динамічності та інтерактивності до веб-сторінок. Можливості JavaScript дозволяють створювати реактивні інтерфейси, обробляти події користувача та взаємодіяти з сервером без перезавантаження сторінки. Широке розповсюдження та підтримка всіма сучасними браузерами роблять JavaScript ідеальним вибором для веб-розробки.

Важливою перевагою JavaScript є можливості маніпуляції DOM (Document Object Model) веб-сторінки, що дає можливість динамічно змінювати вміст та структуру сторінок в реальному часі. Крім того, JavaScript легко інтегрується з іншими веб-технологіями, такими як HTML, CSS, AJAX та WebSocket, що забезпечує створення сучасних та потужних веб-додатків [17].

2.2 Моделювання архітектури соціальної мережі Connectify

Архітектура соціальної мережі Connectify розброблена для забезпечення високої продуктивності, масштабованості та надійності. Система складається з трьох основних компонентів: клієнтської частини (frontend), серверної частини (backend) і бази даних [18].

Frontend відповідає за відображення інтерфейсу користувача та обробку його взаємодії з системою. За допомогою HTML, CSS та JavaScript фронтенд створить зручний та естетичний інтерфейс для користувачів соціальної мережі. Фронтенд використовує шаблонізатор Blade для побудови динамічних та ефективних інтерфейсів. Laravel Blade – це механізм, який дозволяє легко вбудовувати PHP-код безпосередньо в HTML. За допомогою Blade можна створювати шаблони для сторінок, використовувати умови та цикли для генерації динамічного контенту та динамічно підключати дані з бекенду. Він буде взаємодіяти з бекендом через API [19], надсилаючи запити на отримання та відправлення даних.

Бекенд відповідає за обробку логіки додатку, зберігання та взаємодію з даними в базі даних. За допомогою фреймворку Laravel, бекенд реалізує функціональність соціальної мережі. Це включає обробку реєстрації та авторизації користувачів, створення статусів, управління списком друзів та інші функціональні можливості. Бекенд забезпечує безпеку за допомогою механізмів аутентифікації, а також оптимізованої роботи з базою даних для ефективного взаємодії із запитами фронтенду. Він також надає API для взаємодії з фронтендом, що дозволяє розширювати та модифікувати функціонал системи.

Для зберігання даних використовується база даних MySQL. Вона обрана за її надійність, продуктивність і широке поширення. База даних зберігає всю необхідну інформацію про користувачів, їх статуси, коментарі, зв'язки друзів та інші дані. Запити до бази даних здійснюються за допомогою ORM Laravel Eloquent, що забезпечує зручний та ефективний доступ до даних.

Комунікація між фронтендом і бекендом здійснюється за допомогою HTTP-запитів, що дозволяє клієнту надсилати дані на сервер і отримувати відповіді. Запити обробляються контролерами Laravel, які взаємодіють з моделями для доступу до даних у базі даних.

Для ілюстрації цієї архітектури, наведено діаграму на рисунку 2.2.

Рисунок 2.2 – Архітектура соціальної мережі Connectify

Ця діаграма показує основні компоненти системи та взаємодію між ними. Фронтенд відповідає за взаємодію з користувачем і відправляє API-запити до бекенду. Бекенд обробляє ці запити і надсилає SQL-запити до бази даних для отримання або збереження даних.

2.3 Розробка моделей даних соціальної мережі Connectify

2.3.1 Перелік інформаційних сутностей соціальної мережі Connectify

У соціальній мережі Connectify визначено кілька ключових інформаційних сутностей, які зберігають різні типи даних, а саме – сутності «Користувач», «Статус», «Друг» та «Лайк». В таблиці 2.1 наведено атрибути сутності «Користувач».

Таблиця 2.1 – Атрибути сутності «Користувач» соціальної мережі Connectify

Атрибут	Тип даних	Опис атрибуту
id	bigint unsigned	Унікальний ідентифікатор користувача
email	varchar(255)	Адреса електронної пошти
vatar	varchar(255)	Шлях до аватару
username	varchar(255)	Ім'я користувача
password	varchar(255)	Хеш пароля
first_name	varchar(255)	Ім'я користувача
last_name	varchar(255)	Прізвище користувача
location	varchar(255)	Місцезнаходження користувача
dateOfBirth	date	Дата народження
remember_token	varchar(255)	Токен для функції «Запам'ятати мене»
role	varchar(255)	Роль користувача в системі
is_active	tinyint(1)	Статус активності користувача

Таблиця зберігає інформацію про користувачів системи. Кожен користувач має унікальний ідентифікатор, електронну пошту, ім'я користувача, пароль та іншу інформацію. В таблиці 2.2 наведено атрибути сутності «Статус».

Таблиця 2.2 – Атрибути сутності «Статус» соціальної мережі Connectify

Атрибут	Тип даних	Опис атрибуту
1	**2**	**3**
id	bigint unsigned	Унікальний ідентифікатор статусу

Продовження таблиці 2.2

1	2	3
user_id	int	Ідентифікатор користувача, який створив статус
parent_id	int	Ідентифікатор батьківського статусу для коментарів
body	text	Текст статусу
image	varchar(255)	Шлях до зображення статусу

Таблиця зберігає статуси користувачів. Кожен статус містить текст, зображення та пов'язаний з конкретним користувачем.

Таблиця 2.3 містить атрибути сутності «Друг», які зберігають інформацію про дружні зв'язки між користувачами. Вона містить записи про запити на дружбу та їх статус.

Таблиця 2.3 – Атрибути сутності «Друг» соціальної мережі Connectify

Атрибут	Тип даних	Опис атрибуту
id	bigint unsigned	Унікальний ідентифікатор дружби
user_id	int	Ідентифікатор користувача, який надіслав запит на дружбу
friend_id	int	Ідентифікатор користувача, якому надіслано запит на дружбу
accepted	tinyint(1)	Статус прийняття дружби

В таблиці 2.4 наведено атрибути сутності «Лайк» соціальної мережі.

Таблиця 2.4 – Атрибути сутності «Лайк» соціальної мережі Connectify

Атрибут	Тип даних	Опис атрибуту
id	bigint unsigned	Унікальний ідентифікатор лайка
user_id	int	Ідентифікатор користувача, який поставив лайк
likeable_id	int	Ідентифікатор об'єкта, який отримав лайк
likeable_type	varchar(255)	Тип об'єкта, який отримав лайк

Дана сутність містить в собі інформацію про лайки, які користувачі ставлять статусам та коментарям. Вона містить тип та ідентифікатор об'єкта, який отримав лайк.

2.3.2 Проєктування концептуальної моделі даних соціальної мережі Connectify

Концептуальна модель даних для соціальної мережі Connectify була розроблена з урахуванням вимог до функціональності платформи та забезпечення ефективного зберігання і обробки даних. Основними сутностями в цій моделі є «Користувач», «Статус», «Друг» та «Лайк».

Сутність «Користувач» є головною у системі, оскільки вона зберігає інформацію про користувачів. Кожен користувач має унікальний ідентифікатор, електронну пошту, ім'я користувача, пароль, а також інші атрибути, такі як аватар, місцезнаходження і дата народження. «Користувач» має зв'язок «один до багатьох» із сутністю «Статус», це означає, що один користувач може мати багато статусів. Крім того, існує зв'язок «багато до багатьох» із сутністю «Друг», оскільки кожен користувач може мати багато друзів, і кожен користувач може бути другом багатьох інших користувачів.

Сутність «Статус» зберігає статуси користувачів, кожен з яких містить текст статусу, зображення та посилання на ідентифікатор користувача, який

створив цей статус. Кожен статус може мати коментарі, які також є статусами, що створює самозв'язок «один до багатьох» в межах сутності «Статус». Також, один статус може бути вподобаний багатьма користувачами, що створює зв'язок «один до багатьох» із сутністю «Лайк».

Сутність «Друг» зберігає інформацію про дружні зв'язки між користувачами. Ця таблиця містить записи про запити на дружбу, а також статуси цих запитів. Сутність має два зв'язки «багато до багатьох» з сутністю «Користувач», один для користувача, який надіслав запит, і один для користувача, який отримав запит.

Сутність «Лайк» відповідає за зберігання інформації про лайки. Ця таблиця містить дані про те, який користувач поставив лайк, і до якого об'єкта (статусу або коментаря) належить цей лайк. Вона має зв'язок «багато до одного» із сутністю «Користувач», що вказує на користувача, який поставив лайк, і зв'язок «багато до одного» із сутністю «Статус», що вказує на об'єкт, який отримав лайк.

Для того щоб продемонструвати визначені зв’язки між сутностями, на рисунку 2.3 зображено концептуальну модель соціальної мереже Connectify у вигляді ER-діаграми [20].

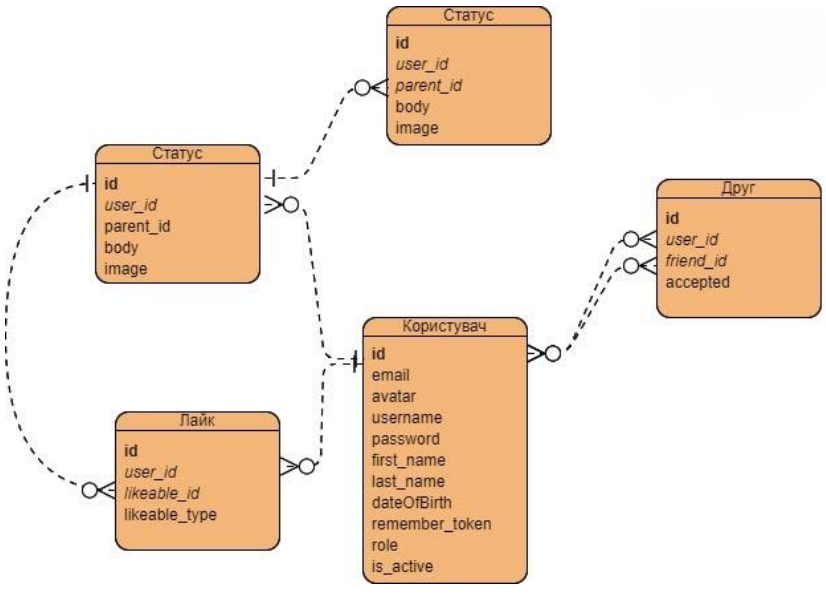

Рисунок 2.3 – ER-діаграма соціальної мережі Connectify

Концептуальна модель даних забезпечує логічне та послідовне зберігання інформації, дозволяючи ефективно обробляти запити користувачів і забезпечувати необхідний функціонал соціальної мережі [21]. Вона створює структуру, яка легко масштабується та підтримує інтеграцію нових функцій без порушення існуючих даних та зв'язків.

2.4 Висновок до другого розділу

У другому розділі було здійснено детальне проєктування соціальної мережі Connectify, починаючи з моделювання архітектури та закінчуючи розробкою концептуальної моделі даних. Спочатку вибрано інформаційно-технологічний стек для розробки та розглянуто загальну архітектуру системи, визначивши основні компоненти та їх взаємодію. Описано, як ці компоненти забезпечують роботу всієї платформи, включаючи користувацький інтерфейс, серверну частину, базу даних та інші сервіси.

Розробка моделей даних соціальної мережі Connectify включала в себе створення переліку інформаційних сутностей та їх детальний опис. Визначено ключові сутності, описано їхні атрибут. Кожна сутність була проаналізована та описана з точки зору її ролі в системі та взаємодії з іншими сутностями.

Проєктування концептуальної моделі даних дозволило визначити логічні зв'язки між сутностями та забезпечити структуру, яка підтримує ефективне зберігання та обробку даних. Таким чином, у другому розділі було закладено міцний фундамент для подальшої розробки соціальної мережі Connectify.

РОЗДІЛ 3. РОЗРОБКА, ВАЛІДАЦІЯ ТА ТЕСТУВАННЯ СОЦІАЛЬНОЇ МЕРЕЖІ CONNECTIFY

3.1 Розробка соціальної мережі Connectify

3.1.1 Розробка інтерфейсу користувача соціальної мережі Connectify

Для розробки інтерфейсу користувача соціальної мережі використано Bootstrap та шаблонізатор Blade. Bootstrap, як фреймворк для фронтенду, забезпечив готові компоненти та стилі, що полегшили розробку та внесли сучасний вигляд. Компоненти Bootstrap також використовувалися для створення адаптивного дизайну. Даний підхід дозволив полегшити розробку дизайну, що дало можливість більше уваги приділити функціоналу.

На рисунку 3.1 зображено головну сторінку для не зареєстрованих користувачів, на якій розміщено форму для авторизації.

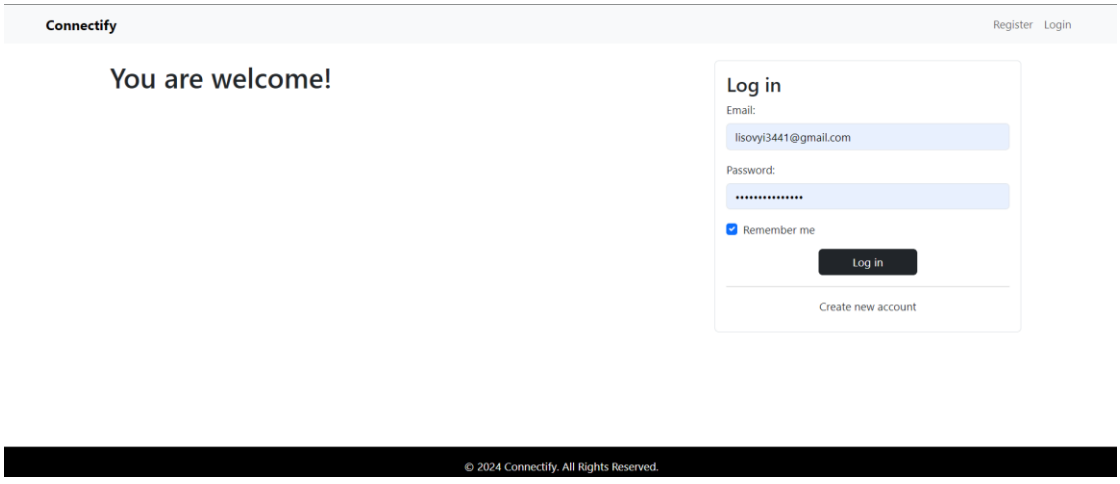

Рисунок 3.1 – Головна сторінка для гостя соціальної мережі Connectify

З даної сторінки є можливість перейти на сторінку реєстрації, яка має схожий вигляд, лише з додатковим полем для нікнейму.

На рисунку 3.2 зображено головну сторінку для авторизованого користувача соціальної мережі.

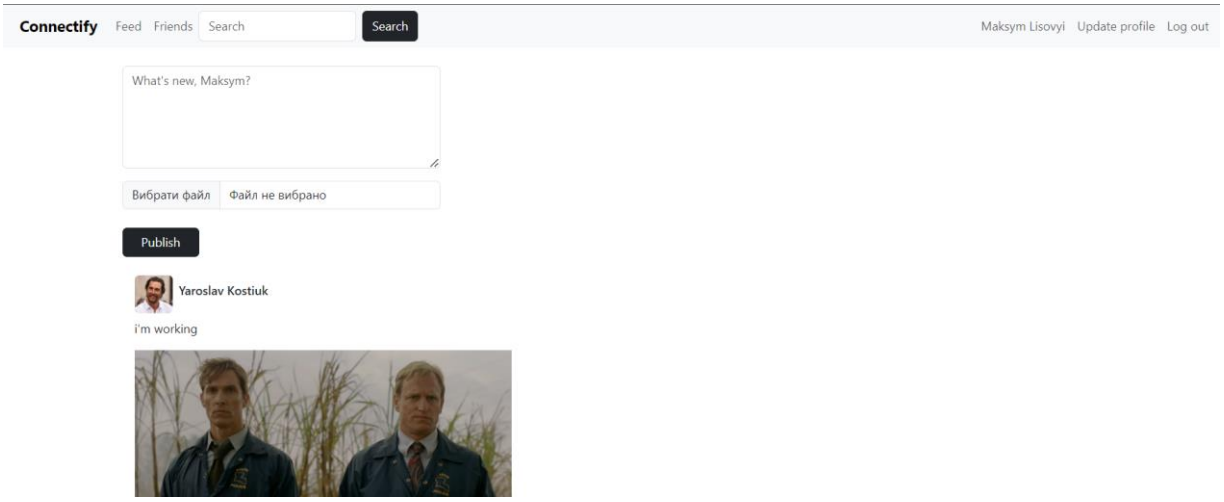

Рисунок 3.2 – Головна сторінка для авторизованого користувача соціальної мережі Connectify

Зверху сторінки розміщено навігаційну панель, з пунктами меню, кнопкою «Вийти», назвою соціальної мережі та формою пошуку, за допомогою якої можна шукати інших користувачів за іменем чи нікнеймом. Також на сторінці розміщено форму для створення нового статусу з можливістю додавання зображення, а під нею список опублікованих статусів друзів користувача.

Варто зауважити, що за допомогою шаблонізатора Blade створено шаблон навігаційної панелі, який підключається для всіх сторінок, що дозволяє мінімізувати написання коду [22]. Такий підхід використано для більшості структурних елементів веб-проєкту, це дозволяє виводити їх кілька разів в проєкті, використовуючи оператор include шаблонізатора Blade. Наприклад – блок користувача та зображення профілю, яке використовується майже всюди. Не потрібно щоразу описувати ці блоки за допомогою HTML, а лише викликати в тому місці де це потрібно. Також, Blade дає можливість описати підключення таблиць стилів та JS-скриптів лише один раз в файлі default.blade.php.

На рисунку 3.3 зображено домашню сторінку користувача.

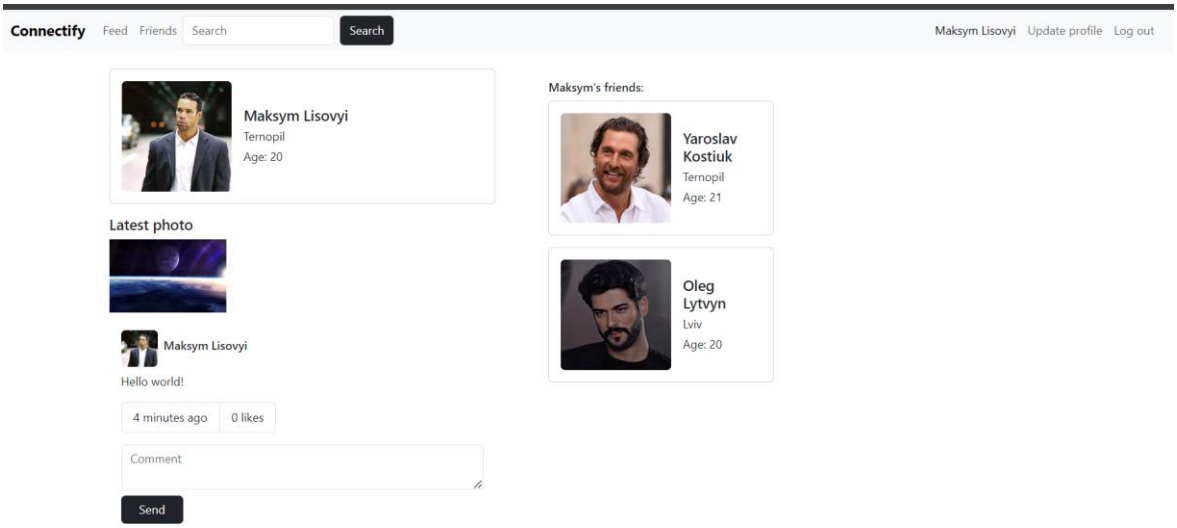

Рисунок 3.3 – Домашня сторінка користувача соціальної мережі Connectify

На сторінці розміщено блок користувача, який містить основну інформацію про нього, якщо така є. В правій частині – список друзів користувача. Під основною інформацією блок «Останні фото», який містить останні завантажені користувачем зображення. Нижче – список опублікованих статусів користувача.

На рисунку 3.4 зображено сторінку для редагування особистих даних користувача соціальної мережі.

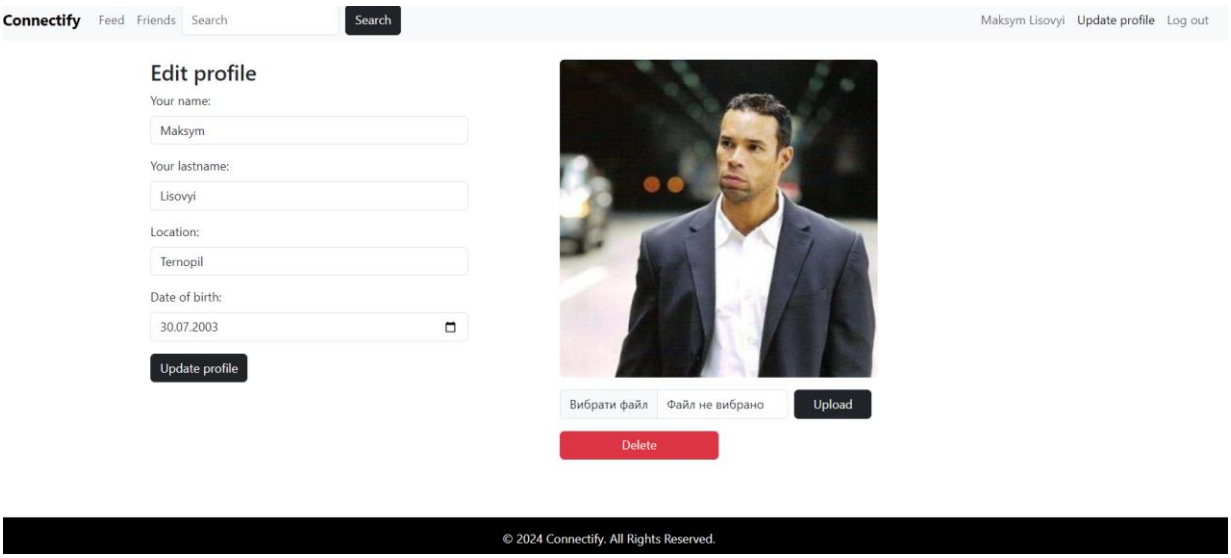

Рисунок 3.4 – Сторінка для редагування особистих даних користувача соціальної мережі Connectify

Оскільки постів в користувача може бути багато, також додано Bootstrap-шаблон пагінації, що наведено на рисунку 3.5. Пагінація – процес розбиття великого масиву даних (даному випадку статусів) на невеликі за обсягом сторінки, з відображенням навігації по цих сторінках за допомогою кнопок [23]. Пагінація використана і всіх масивах даних, що є в соціальній мережі Connectify.

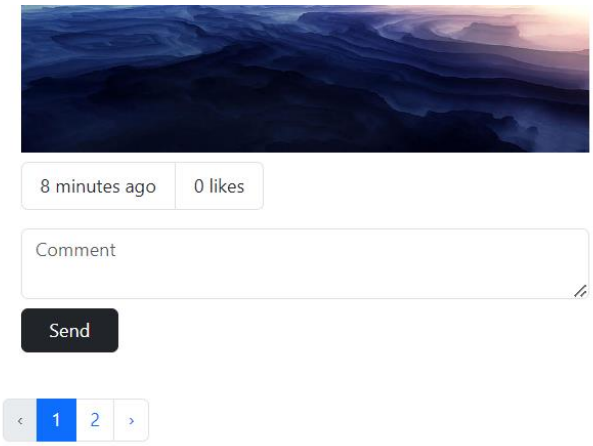

Рисунок 3.5 – Пагінація для статусів соціальної мережі Connectify

Це дозволить виводити одночасно тільки певну кількість постів на сторінці, що збільшить швидкість завантаження сторінки та зробить сторінку більш компактною.

На рисунку 3.6 зображено сторінку для адміністратора.

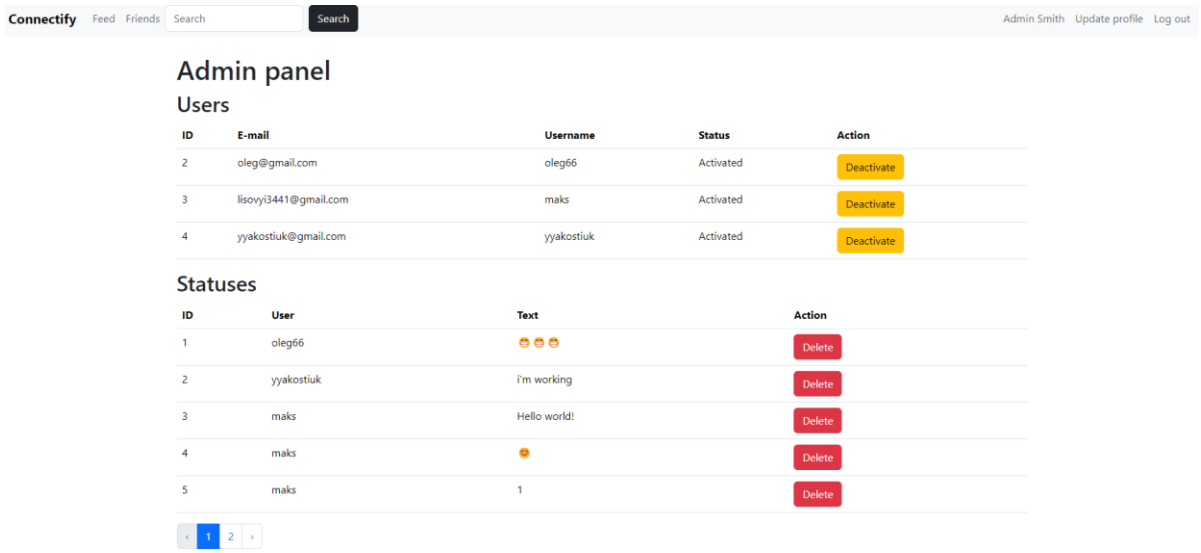

Рисунок 3.6 – Сторінка адміністратора соціальної мережі Connectify

На цій сторінці розміщено два блоки для керування списком користувачів та списком статусів, з використанням пагінації.

3.1.2 Реалізація функціональних можливостей соціальної мережі Connectify

Для початку створено міграції для всіх необхідних таблиць за допомогою командного рядка. Нижче наведена стрічка для створення міграції користувачів

```
php artisan make:migration create_users_table
```

Після її створення додано до неї необхідні поля, лістинг міграції для створення таблиці користувачів соціальної мережі наведено в лістингу 3.1.

Лістинг 3.1 – PHP-код міграції create_users_table

```
class CreateUsersTable extends Migration
{
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('email');
            $table->timestamp('email_verified_at')->nullable();
            $table->string('avatar')->nullable();
            $table->string('username');
            $table->string('password');
            $table->string('first_name')->nullable();
            $table->string('last_name')->nullable();
            $table->string('location')->nullable();
            $table->date('dateOfBirth')->nullable();
            $table->string('remember_token')->nullable();
            $table->string('role')->default('user');
            $table->boolean('is_active')->default(true);
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('users');
```

```
    }
}
```

Дана міграція при її запуску створює таблицю users, з описаними в міграції полями, а при відкаті – видаляє таблицю з бази даних.

Таким же способом створено міграції create_friends_table, create_statuses_table, create_likeable_table.

За допомогою стрічки, наведеної нижче запущено міграції:

```
php artisan migrate
```

Після запуску міграцій в базі даних створені відповідні таблиці. Схему таблиць бази даних, що створились після запуску міграцій наведено на рисунку 3.7.

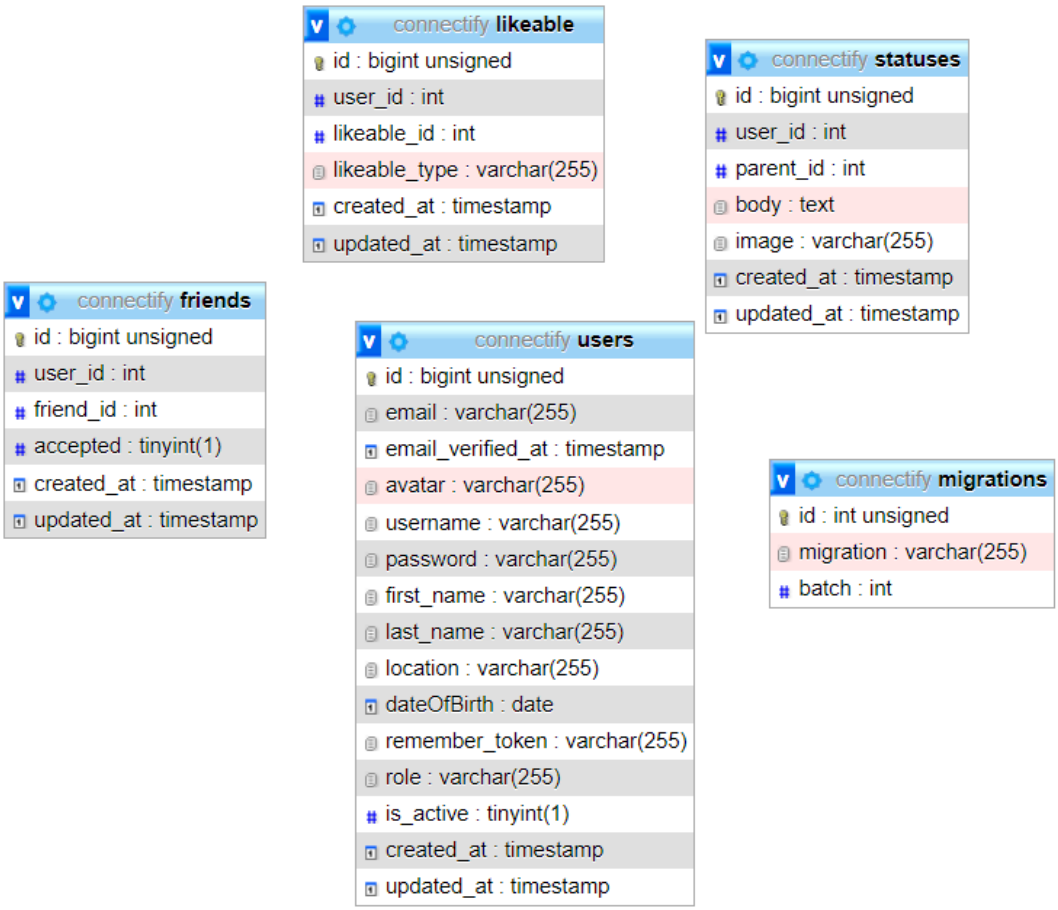

Рисунок 3.7 – Таблиці бази даних соціальної мережі Connectify

Таблиця friends використовується для створення зв’язку один до багатьох між користувачами таблиці users.

Таблиця statuses містить статуси користувачів та поле parent_id, що реалізує систему коментарів.

Таблиця likeable використовує поле likeable_type, щоб ідентифікувати тип об'єкта, який може мати лайки (пост або коментар). Це дозволяє реалізувати, так званий, поліморфний зв’язок.

Таблиця migration містить список створених міграцій.

Перед тим як створювати контролери для сторінок, які б додали функціонал статичним сторінкам соціальної мережі, створено моделі User.php, Status.php та Like.php.

Дані моделі необхідні для того щоб в контролерах створювати об’єкти на основі моделей та використовувати зарезервовані методи. Наприклад, в моделі User використано методи для отримання нікнейму, повного імені або нікнейму та імені або нікнейму користувача соціальної мережі, код яких наведено у лістингу 3.2.

Лістинг 3.2 – PHP-код методів моделі User соціальної мережі Connectify

```
public function getName() {
    if ($this->first_name && $this->last_name)
    {
      return "{$this->first_name} {$this->last_name}";
    }
    if ($this->first_name)
    {
      return $this->first_name;
    }
    return null;
  }
  public function getNameOrUsername()
  {
    return $this->getName() ?: $this->username;
  }
  public function getFirstNameOrUsername()
  {
    return $this->first_name ?: $this->username;
  }
```

Дані методи дозволяють використовувати необхідні дані користувача на веб-сторінках соціальної мережі.

Варто також згадати, що за допомогою моделей створюється зв’язок між сутностями, для цього описуються спеціальні методи, які співзвучні з назвою сутності з якою зв’язують [24]. Наприклад:

```
public function statuses()
 {
  return $this->hasMany('App\Models\Status', 'user_id');
 }
```

Даний метод дозволить для об’єкта класу User отримати масив з усіма статусами, які містять його user_id.

В лістингу 3.3 наведено методи для додавання та видалення друзів.

Лістинг 3.3 – PHP-код методів додавання та видалення друзів

```
public function addFriend(User $user)
{
 $this->friendOf()->attach($user->id);
}
public function deleteFriend(User $user)
{
 $this->friendOf()->detach($user->id);
 $this->friendsOfMine()->detach($user->id);
}
```

Ці методи використовують методи attach та detach [25] для додавання та видалення записів у таблиці friends. При додаванні друга викликається метод attach для відношення friendOf, щоб додати користувача до списку друзів. При видаленні друга спочатку відбувається видалення запису у відношенні friendOf, а потім у відношенні friendsOfMine, щоб повністю видалити зв'язок.

Для усіх моделей створено необхідні методи для подальшої реалізації функціональних можливостей системи. Програмний код моделей наведено в додатку Б.

Наступним етапом є створення контролерів для обробки get та post запитів. В лістингу 3.4 наведено приклад контролера AdminController.

Лістинг 3.4 – PHP-код класу AdminController

```
class AdminController extends Controller {
    public function index() {
        $users = User::where('role', 'user')->paginate(5);
        $statuses = Status::paginate(5);
        return view('admin.index', compact('users', 'statuses'));
    }
    public function deactivateUser($id) {
        $user = User::findOrFail($id);
        $user->is_active = false;
        $user->save();
        return redirect('/admin')->with('success', 'The user has
been deactivated');
    }
    public function activateUser($id) {
        $user = User::findOrFail($id);
        $user->is_active = true;
        $user->save();
        return redirect('/admin')->with('success', 'The user has
been activated');
    }
    public function deleteStatus($id) {
        $status = Status::findOrFail($id);
        $status->delete();
        return     redirect('/admin')->with('success',     'Status
deleted');
    }
}
```

Клас AdminController розширює базовий контролер Laravel [26] і містить методи для адміністрування користувачів і статусів. У методі index витягуються всі користувачі з роллю "user" та всі статуси, розбиваючи їх на сторінки по 5 елементів, після чого ці дані передаються у view admin.index.

Метод deactivateUser змінює статус користувача з активного на неактивний за його id, зберігає зміни в базі даних і перенаправляє на сторінку адміністратора з повідомленням про успіх. Метод activateUser робить протилежне: активує користувача за його id, зберігає зміни і також перенаправляє на сторінку адміністратора з повідомленням про успіх. Метод deleteStatus видаляє статус за його id, зберігає зміни і перенаправляє на сторінку адміністратора з повідомленням про успіх.

Також в лістингу 3.5 наведено програмний код контролера SearchController, який реалізує пошук користувачів.

Лістинг 3.5 – PHP-код класу SearchController

```
class SearchController extends Controller
{
    public function getResults(Request $request)
    {
        $query = $request->input('query');
        if (!$query)
        {
            redirect()->route('home');
        }
        $users = User::where('first_name', 'like', "%$query%")
            ->orWhere('last_name', 'like', "%$query%")
            ->orWhere('username', 'like', "%$query%")
            ->get();;
        return view('search.results')->with('users',$users);
    }
}
```

Лістинги інших контролерів наведено в додатку В.

Для коректної взаємодії візуальної частини з контролерами створено роути. В Laravel роутинг відповідає за визначення, які дії повинні виконуватися при конкретних HTTP-запитах до веб-застосунка [27]. Роути допомагають системі визначити, які контролери та їх методи повинні бути викликані при отриманні певного URL.

В лістингу 3.6 наведено приклад роутів для контролера ProfileController.

Лістинг 3.6 – PHP-код роутів для методів класу ProfileController

```
Route::middleware(['auth', 'admin'])->group(function () {
    Route::get('/admin', [AdminController::class, 'index']);
    Route::post('/admin/deactivate-user/{id}',
[AdminController::class, 'deactivateUser']);
    Route::post('/admin/activate-user/{id}',
[AdminController::class, 'activateUser']);
    Route::post('/admin/delete-status/{id}',
[AdminController::class, 'deleteStatus']);
```

Ці маршрути визначають URL-адреси і методи контролера, які виконуються при зверненні до них. Вони згруповані з використанням посередників auth і admin, що забезпечує доступ до них тільки для автентифікованих користувачів з роллю адміністратора.

Маршрут GET /admin виконує метод index класу AdminController, що відображає адміністративну панель з переліком користувачів та статусів. Маршрут POST /admin/deactivate-user/{id} виконує метод deactivateUser класу AdminController, що деактивує користувача з вказаним ID. Маршрут POST /admin/activate-user/{id} виконує метод activateUser класу AdminController, що активує користувача з вказаним ID. Маршрут POST /admin/delete-status/{id} виконує метод deleteStatus класу AdminController, що видаляє статус з вказаним id.

Лістинг файлу web.php, що містить всі роути соціальної мережі наведено в додатку Д.

3.2 Валідація та тестування соціальної мережі Connectify

3.2.1 Валідація даних соціальної мережі

Валідація даних є критичним етапом розробки соціальної мережі, оскільки вона допомагає уникнути неправильного введення та забезпечити безпеку інформації. В Laravel валідація даних використовується для перевірки коректності введених користувачем даних перед їх обробкою або збереженням у базу даних [28]. Для валідації використовуються правила, які можна застосовувати до різних полів форми або інших введених даних. Основна ідея валідації полягає в тому, щоб переконатися, що дані відповідають визначеним правилам. Нижче наведено програмний код що демонструє валідацію даних введення форми реєстрації.

Лістинг 3.7 – PHP-код валідації форми реєстрації.

```
$this->validate($request, [
        'email' => 'required|unique:users|email|max:255',
        'username' => 'required|unique:users|alpha_dash|max:20',
        'password' => 'required|min:6',
        ]);
```

В даному прикладі, для поля email валідація встановлює вимогу про його обов'язковість (required), унікальність в базі даних (unique:users), правильний формат електронної пошти (email), та обмеження довжини до 255 символів (max:255). Для username встановлені правила щодо обов'язковості (required), унікальності в базі даних (unique:users), дозволення лише буквено-цифрових символів та деяких спеціальних (alpha_dash), та обмеження довжини до 20 символів (max:20). Для password встановлено правила обов'язковості (required) та мінімальної довжини в 6 символів (min:6).

Якщо валідація не пройде для будь-якого з цих полів, користувач буде перенаправлений на попередню сторінку з відображеними повідомленнями про помилки [29]. Для цього використано шаблонізатор Blade з умовою if, приклад коду якого в лістингу 3.8.

Лістинг 3.8 – HTML-код виведення повідомлення про неправильно введене поле

```
@if($errors->has('email'))
    <span class="help-block text-danger">
        {{$errors->first('email')}}
    </span>
@endif
```

Даний код виводить повідомлення про помилку введення поля email, при умові що масив $errors має елемент з індексом «email».

3.2.2 Тестування функціональних можливостей соціальної мережі

Спочатку протестовано можливість реєстрації нового користувача. Для цього в форму реєстрації введено нові дані та відправлено за допомогою кнопки. В результаті створено нового користувача, запис про якого наведено на рисунку 3.8.

id	email	email_verified_at	avatar	username
3	example@gmail.com	*NULL*	*NULL*	example

Рисунок 3.8 – Новий користувач соціальної мережі Connectify

Після успішної реєстрації нового користувача авторизовано за допомогою форми авторизації. За допомогою форми редагування додано інформацію про користувача, таку як ім’я, прізвище, місцезнаходження та дату народження. Також додано зображення профілю. На рисунку 3.9 зображено результат додавання особистої інформації користувача.

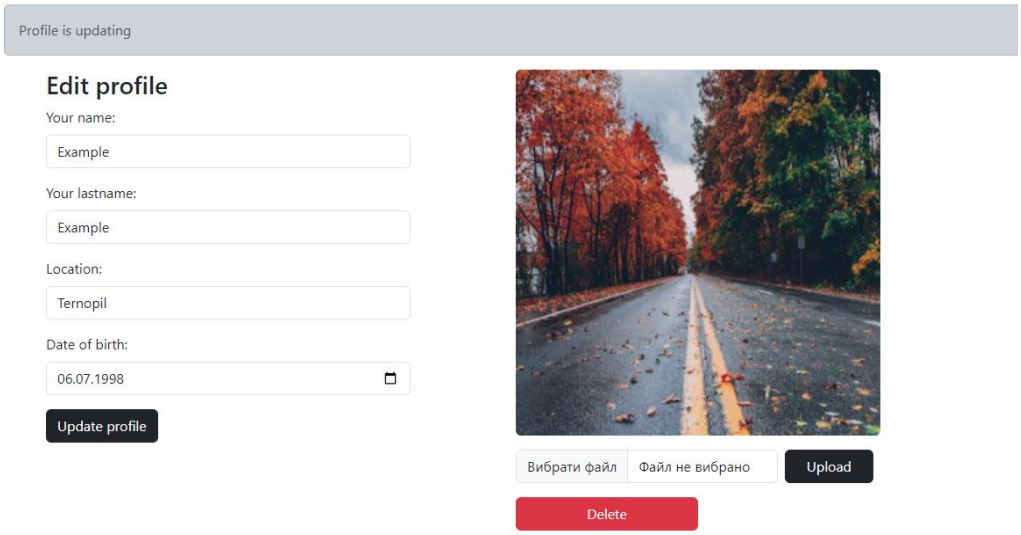

Рисунок 3.9 – Додавання інформації про користувача соціальної мережі Connectify

На сторінці Feed за допомогою форми створено новий статус для перевірки коректності роботи цієї функції, результат на рисунку 3.10.

Рисунок 3.10 – Новий статус користувача соціальної мережі Connectify

Для подальшого тестування системи додавання в друзі та системи коментарів створено ще декількох користувачів та заповнено їхні сторінки.

Після чого, авторизувавшись в першому створеному акаунті, протестовано пошук користувачів, результат на рисунку 3.11.

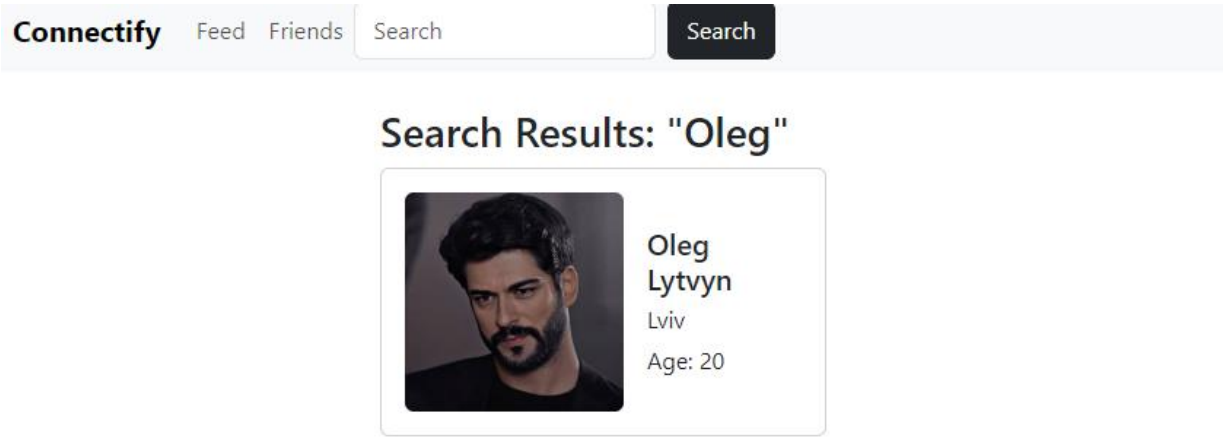

Рисунок 3.11 – Пошук користувачів за ключовим словом «Oleg»

Після перегляду профілю знайденого користувача, який зображено на рисунку 3.12, видно що немає можливості коментувати його пост. Так і повинно бути, адже він не є другом поки не надіслано і не прийнято запит на додавання в друзі.

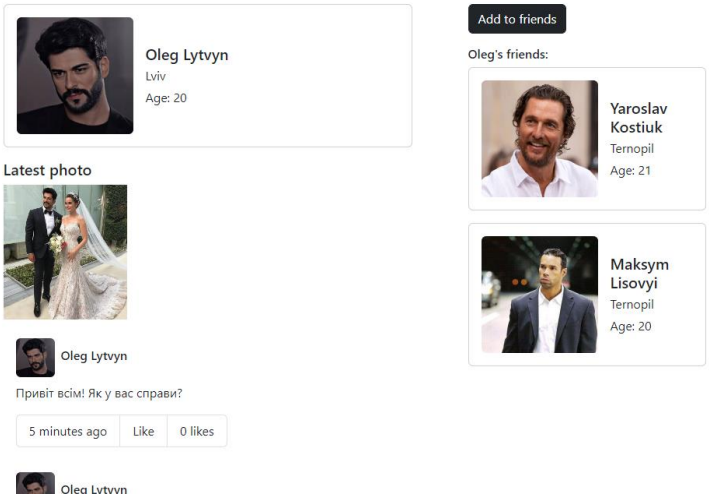

Рисунок 3.12 – Перегляд профілю іншого користувача соціальної мережі Connectify

Надіслано користувачу запит в друзі за допомогою кнопки та підтверджено запит з його профілю. Після цього сторінка користувача змінилась, результат на рисунку 3.13

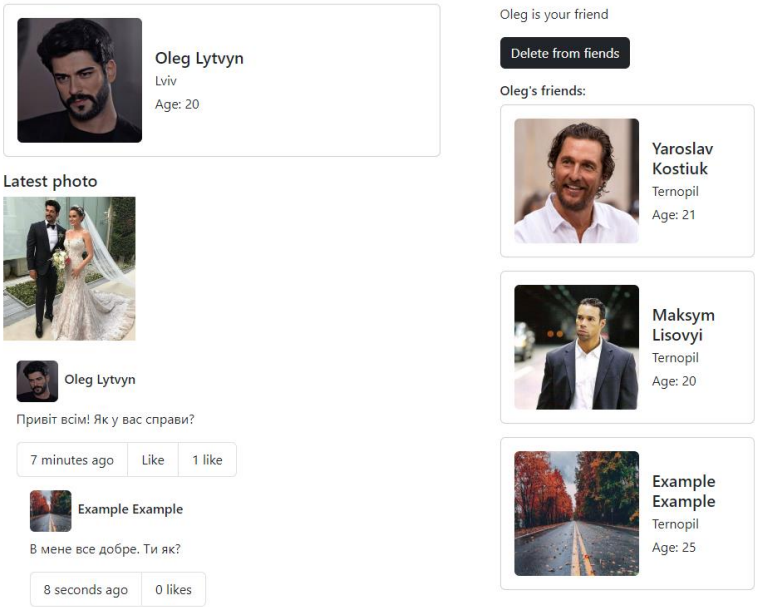

Рисунок 3.13 – Перегляд профілю друга

Як видно з рисунку, написано новий коментар та вподобано пост користувача. Також профіль авторизованого користувача додано у список

друзів на сторінці, про що говорить можливість писати коментарі і відображення авторизованого користувача в списку друзів праворуч.

Наступним етапом тестування було видалення користувача з друзів та вихід з системи, все працює коректно.

Далі створено нового користувача, який буде виконувати роль адміністратора, для цього в phpMyAdmin дано йому права адміністрування, змінивши значення поля role на «admin». Перейшовши в адміністративну панель, видалено кілька статусів та деактивовано користувача. Для того щоб переконатись, що деактивація пройшла успішно, виконана спроба увійти з облікового запису користувача який було деактивовано. Результат наведено на рисунку 3.14.

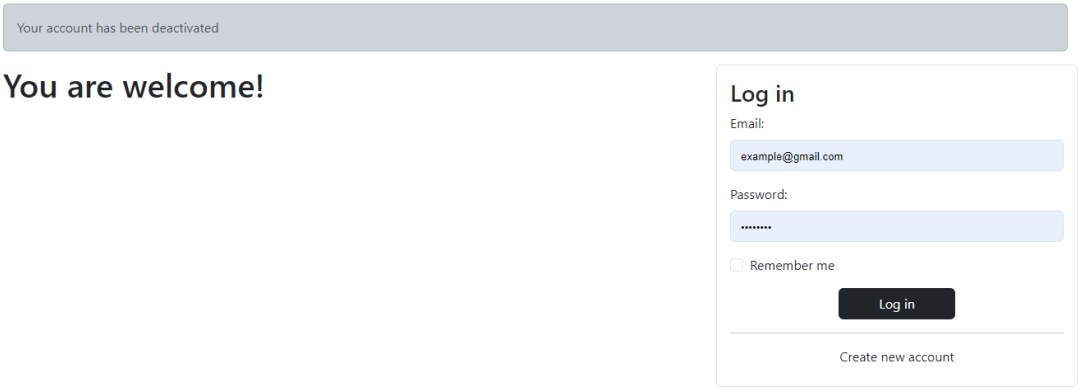

Рисунок 3.14 – Спроба входу в деактивований профіль соціальної мережі Connectify

Як видно з рисунка, отримано повідомлення про те що аккаунт деактивований та можливість входу відсутня.

Тестування пройшло успішно, всі функціональні можливості соціальної мережі працюють належним чином.

3.3 Висновок до третього розділу

У третьому розділі кваліфікаційної роботи було розглянуто процес розробки, валідації та тестування соціальної мережі Connectify. Детально описано розробку інтерфейсу користувача, реалізацію функціональних можливостей платформи та заходи з валідації даних. Крім того, проведено ретельне тестування системи, що забезпечує її стабільну та безпечну роботу.

Результати цього розділу свідчать про успішну реалізацію основних функцій соціальної мережі Connectify, високу якість коду та надійність системи.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Значення адаптації в трудовому процесі

Праця людини безпосередньо пов’язана з виробничим середовищем. Працівник може ефективно виконувати свої обов'язки тільки за умови, що зовнішнє середовище є оптимальним. Якщо умови праці змінюються та стають несприятливими, організм людини активує спеціальний механізм, який підтримує стабільність внутрішнього середовища або змінює його в межах допустимого. Цей механізм називається адаптацією [37].

Адаптація (від лат. adapto – пристосування) – це динамічний процес пристосування організму та його систем до змінних умов зовнішнього середовища. У трудовій діяльності вона поділяється на фізіологічну, психічну, соціальну та професійну [37].

Фізіологічна адаптація – це сукупність фізіологічних реакцій, що лежать в основі пристосування організму до змін зовнішніх умов і спрямовані на збереження відносної стабільності внутрішнього середовища – гомеостазу. Гомеостаз (від грец. homoios – подібний, однаковий та грец. stasis – стан, непорушність) означає відносну динамічну стабільність складу та властивостей внутрішнього середовища і стійкість основних фізіологічних функцій організму людини. Гомеостаз підтримується на всіх рівнях організації організму, забезпечуючи динамічну рівновагу між організмом і зовнішнім середовищем.

Механізм адаптації полягає в змінах чутливості аналізаторів, розширенні діапазону фізіологічних резервів організму та зміні параметрів фізіологічних функцій у певних межах. Завдяки фізіологічній адаптації, фізичні та біохімічні параметри, які визначають життєдіяльність організму, змінюються у вузьких межах порівняно зі значними змінами зовнішніх умов. Це підвищує стійкість організму до холоду, тепла, нестачі кисню, змін барометричного тиску та інших факторів. Велике значення у фізіологічній адаптації мають реактивність

організму, його початковий функціональний стан (вік, тренованість тощо), від яких залежать відповідні реакції організму на різні впливи.

Фізіологічна адаптація до праці має активний характер і за сприятливих умов виробничого середовища та оптимальних навантажень веде до підвищення стійкості та працездатності організму, збільшення його резервних можливостей, зменшення захворювань і травматизму. Проте коливання умов середовища, в яких відбувається фізіологічна адаптація, мають певну межу, характерну для кожного організму.

Психічна адаптація – це процес встановлення оптимальної відповідності особистості до навколишнього середовища в процесі діяльності. Властивості, такі як гальмування мислення, низька швидкість обробки інформації, обмежений діапазон сприйняття та порушення функцій пам'яті, можуть перешкоджати адаптації; водночас висока рухливість нервових процесів сприяє їй. Психічна адаптація на робочому місці залежить від психічних властивостей працівника, його психічного стану, реакцій на стресові ситуації, що виникають на роботі, кваліфікації, культури, особливостей професійної діяльності та конкретних умов праці.

Соціальна адаптація – це пристосування працівника до системи відносин у робочому колективі з його нормами, правилами, традиціями та ціннісними орієнтаціями. Якщо соціальна адаптація проходить несприятливо, рівень стресу на роботі підвищується, що може призвести до міжособистісних конфліктів і нещасних випадків.

Професійна адаптація – це адаптація до трудової діяльності з усіма її складовими: робочим місцем, знаряддями та засобами праці, об’єктами та предметами праці, особливостями технологічного процесу, часовими параметрами роботи тощо. Професійна адаптація виражається у формуванні стійкого позитивного ставлення працівника до своєї професії, оволодінні специфічними навичками та вміннями, необхідними для якісного виконання роботи. Вона визначається рівнем знань та навичок, отриманих під час навчання, ступенем відповідальності, практичністю та діловитістю. Адаптація

вважається завершеною, коли працівник досягає кваліфікації, відповідної встановленим стандартам [38].

У цьому контексті соціальна мережа Connectify відіграє важливу роль, надаючи можливість співробітникам взаємодіяти, обмінюватися інформацією та досвідом, що сприяє швидкій адаптації до нових технологій та способів спілкування в колективі. Будучи побудованою на Laravel, ця соціальна мережа забезпечує високий рівень безпеки та ефективну роботу, що робить її важливим інструментом для забезпечення успішної адаптації працівників у сучасному робочому середовищі.

4.2 Загальні вимоги безпеки з охорони праці для користувачів ПК

У першу чергу, для самостійної роботи з ПК слід допускати лише тих осіб, які ознайомлені з правилами безпеки. Вони також мають пройти відповідне навчання з охорони праці та пожежної безпеки при використанні комп'ютера. Згодом, у необхідних випадках, ці особи мають мати медичний огляд і дозвіл на роботу з ПК, враховуючи їхній стан здоров'я [39].

Перед початком роботи з ПК, користувачі повинні переконатися, що обладнання та проводка не пошкоджені. У разі виявлення пошкоджень вони повинні повідомити відповідальну особу та утриматися від роботи, поки не буде усунуто небезпеку.

Робоче місце користувача ПК має бути розташоване так, щоб уникнути відблисків на екрані монітора. Найкраще розміщення – перпендикулярно до вікна. Важливо забезпечити достатній простір для зручності та вільного доступу до всіх необхідних елементів (монітор, клавіатура, миша, документи).

Освітлення повинно бути достатнім, але не надто яскравим, щоб уникнути відблисків і зайвого напруження очей. Рекомендується використовувати комбінацію природного та штучного освітлення. Робоче місце має бути обладнане локальними джерелами світла (настільні лампи) для додаткового освітлення.

Екран монітора повинен бути розташований на відстані 50-70 см від очей користувача, верхній край екрану має бути на рівні або трохи нижче рівня очей. Необхідно налаштувати яскравість і контрастність екрану для комфортного перегляду. Клавіатура має бути розміщена на рівні ліктів або трохи нижче, щоб уникнути надмірного напруження рук і плечей. Миша повинна бути розташована поруч з клавіатурою.

Загальна організація робочого місця включає також забезпечення комфортного крісла з підтримкою поперекового відділу хребта. Під час роботи необхідно робити регулярні перерви для розминки та відпочинку очей. Важливо слідкувати за правильною поставою, щоб уникнути довготривалих проблем зі спиною і шиєю.

Приклад коректної посадки користувача ПК так його робочого місця відповідно до загальних вимог безпеки з охорони праці подано на рисунку 4.1 [40].

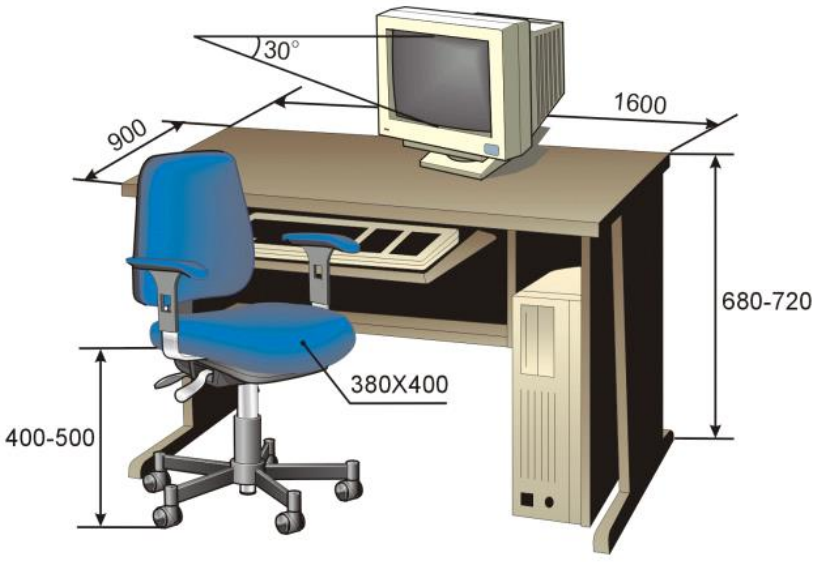

Рисунок 4.1 – Коректна посадка користувача ПК і його робоче місце згідно вимог безпеки з охорони праці [41]

Згідно з вимогами безпеки охорони праці для користувачів ПК, вони повинні бути ознайомлені з процедурою дій у критичних або аварійних ситуаціях. Так, у випадку почуття стороннього звуку чи горіння запаху,

важливо негайно припинити роботу з ПК, вимкнути всі пристрої, відключити їх від електромережі та повідомити про це відповідальну особу [42].

При використанні ПК користувачам необхідно дотримуватися всіх зазначених вище правил та рекомендацій, що сприятиме комфортній та ефективній роботі.

4.3 Висновок до четвертого розділу

В даному розділі кваліфікаційної роботи описано значення адаптації в трудовому процесі та загальні вимоги безпеки з охорони праці для користувачів ПК. Підкреслено важливість адаптаційних процесів, які сприяють підвищенню працездатності, зниженню рівня захворюваності та травматизму серед працівників. Розглянуто чотири основні типи адаптації: фізіологічну, психічну, соціальну та професійну. Кожен з них має свої особливості та вплив на загальний стан працівника і його здатність ефективно виконувати робочі завдання.

Детально розглянуто загальні вимоги безпеки з охорони праці для користувачів ПК. Визначено, що для забезпечення безпеки при роботі з комп'ютерами необхідно дотримуватися ряду правил. Також наголошено на важливості ознайомлення працівників з діями у критичних або аварійних ситуаціях для забезпечення їхньої безпеки та зниження ризиків.

Таким чином, розділ надає комплексне уявлення про значення адаптації в трудовому процесі та вимоги з охорони праці для користувачів ПК, що сприяє покращенню умов праці та підвищенню безпеки на робочих місцях.

ВИСНОВКИ

В процесі виконання кваліфікаційної роботи освітнього рівня «Бакалавр» з розробки соціальної мережі Connectify створено функціональний прототип соціальної мережі, яка здатна задовольнити потреби користувачів у спілкуванні та взаємодії. Використання сучасних технологій, таких як PHP Laravel, MySQL, Bootstrap та JavaScript, дозволило забезпечити високу продуктивність та зручність використання платформи.

В першому розділі кваліфікаційної роботи:

– Подано аналіз сучасних соціальних мереж та їх функціональних можливостей.

– Розглянуто вимоги та постановку завдання для розробки соціальної мережі Connectify.

– Висвітлено пошук актантів та можливі варіанти використання соціальної мережі Connectify.

В другому розділі кваліфікаційної роботи:

– Досліджено та вибрано оптимальний інформаційно-технологічний стек для розробки соціальної мережі Connectify, який включає в себе PHP Laravel, MySQL, Bootstrap та JavaScript.

– Обґрунтовано архітектурні рішення та моделі даних, що використовуються в Connectify.

– Сформовано концептуальну модель даних та перелік інформаційних сутностей.

В третьому розділі кваліфікаційної роботи:

– Розроблено інтерфейс користувача та функціональні можливості соціальної мережі Connectify.

– Запропоновано методи валідації даних та тестування функціональних можливостей.

– Спроєктовано механізми забезпечення безпеки даних користувачів.

– Протестовано функціональні можливості соціальної мережі для забезпечення її стабільної роботи.

У розділі «Безпека життєдіяльності, основи охорони праці»:

– Висвітлено значення адаптації в трудовому процесі.

– Розглянуто загальні вимоги безпеки з охорони праці для користувачів ПК.

Завдяки проведеній роботі було досягнуто поставленої мети – створення сучасної соціальної мережі Connectify, яка відповідає сучасним вимогам та стандартам у сфері веб-розробки.

ПЕРЕЛІК ДЖЕРЕЛ

1 We are social 2023: соціальні мережі, інтернет та тенденції електронної комерції. Elitweb. URL: https://elit-web.ua/ua/blog/we-are-social-2023/ (дата звернення: 04.04.2024).

2 Вплив соціальних мереж на суспільство і бізнес: переваги, недоліки та майбутні перспективи. InProject – IT компанія, яка створює неймовірні digital продукти. URL: https://inproject.org/vplyv-soczialnuh-merezh-na-suspilstvo-i-biznes/ (дата звернення: 04.04.2024).

3 Учасники проектів Вікімедіа. Facebook – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Facebook (дата звернення: 04.04.2024).

4 Вплив Instagram на бренд вашого бізнесу: все, що потрібно знати - Genius.Space. Genius.Space. URL: https://genius.space/lab/vpliv-instagram-na-brend-vashogo-biznesu-vse-shho-potribno-znati/ (дата звернення: 06.04.2024).

5 Учасники проектів Вікімедіа. Твіттер – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Твіттер (дата звернення: 06.04.2024).

6 Особливості хайпової соцмережі TikTok - Press Association. Press Association. URL: https://pressassociation.org.ua/ua/osoblivosti-hajpovoï-soczmerezhi-tiktok/ (дата звернення: 06.04.2024).

7 Функціональні вимоги – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Функціональні_вимоги (дата звернення: 10.10.2023).

8 Нефункціональні вимоги – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Нефункціональні_вимоги (дата звернення: 10.10.2023).

9 Функціональне моделювання. автоматизоване проектування інформаційних систем. українські підручники та статті – бібліотека posibniki.com.ua. Українські підручники, посібники та статті – Бібліотека Posibniki. Електронна бібліотека підручників онлайн. URL:

https://posibniki.com.ua/post-funkcionalne-modelyuvannya (дата звернення: 12.10.2023).

10 Brush K. What is a Use Case?. Software Quality. URL: https://www.techtarget.com/searchsoftwarequality/definition/use-case (дата звернення: 13.10.2023).

11 Contributors to Wikimedia projects. Use case diagram - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Use_case_diagram (дата звернення: 13.10.2023).

12 10 reasons why laravel is the best PHP framework | vilmate. *Nearshore Software Development Company in Ukraine – VILMATE*. URL: https://vilmate.com/blog/why-laravel-is-the-best-php-framework/ (дата звернення: 06.04.2024).

13 Understanding the Laravel MVC Architecture – Everything You Need to know 2024. *Pixlogix*. URL: https://www.pixlogix.com/laravel-mvc-architecture/ (дата звернення: 06.04.2024).

14 Laravel - The PHP Framework For Web Artisans. *Laravel – The PHP Framework For Web Artisans*. URL: https://laravel.com/ (дата звернення: 06.04.2024).

15 Laravel Security Best Practices: Safeguard Your Web Apps. *WPWeb Infotech*. URL: https://wpwebinfotech.com/blog/laravel-security-best-practices/ (дата звернення: 11.04.2024).

16 What is MySQL?. *Oracle | Cloud Applications and Cloud Platform*. URL: https://www.oracle.com/mysql/what-is-mysql/ (дата звернення: 11.04.2024).

17 Neville M. The Advantages and Disadvantages of JavaScript - Softjourn. *Softjourn Inc*. URL: https://softjourn.com/insights/the-advantages-and-disadvantages-of-javascript (дата звернення: 17.04.2024).

18 Understanding the Web Application Architecture Fundamentals. *Cleveroad Inc. – Web and App development company*. URL: https://www.cleveroad.com/blog/web-application-architecture/ (дата звернення: 17.04.2024).

19 How to Call an External API Using Laravel (Tutorial). *Laracoding*. URL: https://laracoding.com/calling-an-external-api-with-laravel/ (дата звернення: 17.04.2024).

20 What is an Entity Relationship Diagram (ERD)?. *Lucidchart*. URL: https://www.lucidchart.com/pages/er-diagrams (дата звернення: 22.04.2024).

21 Olin T. Understanding Data Modeling and Data Model Types. *Agile Data Engine*. URL: https://www.agiledataengine.com/blog/data-modeling-and-data-model-types (дата звернення: 22.04.2024).

22 Laravel's blade template. *Stack Overflow*. URL:https://stackoverflow.com/questions/60562634/laravels-blade-template (дата звернення: 28.04.2024).

23 Hibernate - Pagination - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/hibernate-pagination/ (дата звернення: 28.04.2024).

24 How does laravel finds the connection between models and its table in database. *Stack Overflow*. URL: https://stackoverflow.com/questions/38730360/how-does-laravel-finds-the-connection-between-models-and-its-table-in-database (дата звернення: 02.05.2024).

25 Merchant A. Attach, detach and sync many-to-many relationships in Laravel. *Amit Merchant*. URL: https://www.amitmerchant.com/attach-detach-sync-laravel/ (дата звернення: 02.05.2024).

26 How does laravel finds the connection between models and its table in database. *Stack Overflow*. URL: https://stackoverflow.com/questions/38730360/how-does-laravel-finds-the-connection-between-models-and-its-table-in-database (дата звернення: 23.05.2024).

27 Mastering Laravel Routing: A Comprehensive Guide to Building Robust Web Applications. *Work Done Right | Software Development for the Web | Laravel, PHP, Shopify, TailwindCSS, Amazon Webservices DevOps*. URL: https://workdoneright.co/blog/mastering-laravel-routing-a-comprehensive-guide-to-building-robust-web-applications (дата звернення: 23.05.2024).

28 Data Validation In Laravel: A Comprehensive Guide. *WPWeb Infotech*. URL: https://wpwebinfotech.com/blog/data-validation-in-laravel/ (дата звернення: 29.05.2024).

29 *Laracasts*. URL: https://laracasts.com/discuss/channels/requests/keep-inputs-after-failed-validation?page=1&replyId=127259 (дата звернення: 29.05.2024).

30 Fryz M., Mlynko B. Property Analysis of Conditional Linear Random Process as a Mathematical Model of Cyclostationary Signal // 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2022). Ternopil, Ukraine: CEUR Workshop Proceedings, 2022. Vol. 3309. P. 77–82.

31 Fryz M., Mlynko B. Determination of the characteristic function of discrete-time conditional linear random process and its application // Sci. J. TNTU. 2023. Vol. 109, № 1. P. 16–23.

32 Fryz M., Mlynko B. Property analysis of multivariate conditional linear random processes in the problems of mathematical modelling of signals // Technol. Audit Prod. Reserv. 2022. Vol. 3, № 2(65). P. 29–32.

33 Фриз М.Є., Млинко Б.Б. Умовні лінійні випадкові процеси з дискретним часом та їх властивості // Вісник Хмельницького національного університету. Серія: Технічні науки. 2022 (309), № 3. С. 7–12.

34 Fryz M., Mlynko B. Properties of Stationarity and Cyclostationarity of Conditional Linear Random Processes // 2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET). Lviv-Slavske, Ukraine: IEEE, 2020. P. 166–170.

35 Fryz M., Scherbak L., Mlynko B., Mykhailovych T. Linear Random Process Model-Based EEG Classification Using Machine Learning Techniques // Proceedings of the 1st International Workshop on Computer Information Technologies in Industry 4.0 (CITI 2023). Ternopil, Ukraine: CEUR Workshop Proceedings, 2023. Vol. 3468. P. 126–132.

36 Бабак В. П., Марченко Б. Г., Фриз М. Є. Теорія ймовірностей, випадкові процеси та математична статистика. К.: Техніка, 2004. 288 с.

37 Адаптація трудова. *BYE*. URL:https://vue.gov.ua/Адаптація_трудова (дата звернення: 12.06.2024).

38 Значення адаптації в трудовому процесі - Studies. *Studies*. URL: https://studies.in.ua/bjd-gandzyuk/923-71-znachennya-adaptacyi-v-trudovomu-proces.html (дата звернення: 12.06.2024).

39 Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин. *Офіційний вебпортал парламенту України*. URL: https://zakon.rada.gov.ua/laws/show/z0293-10#Text (дата звернення: 12.06.2024).

40 Охорона праці при роботі з комп'ютером. *Бухгалтерські послуги. Аутсорсинг, аутстафінг | Компанія Вікторія*. URL: https://www.victorija.ua/dovidnik/osnovni-pravyla-dotrymannya-ohorony-pratsi-pry-roboti-na-personalnyh-eom.html (дата звернення: 12.06.2024).

41 М. П. Купчик / Основи Охорони Праці / М. П. Купчик, М. П. Гандзюк, І. Ф. Степанець та ін.. – Київ, 2000. – 416 с.

42 Інструкція з охорони праці при роботі з комп'ютером, принтером, ксероксом та іншою оргтехнікою | Інструкції для навчальних закладів України. *Інструкції для навчальних закладів України | Інструкції з охорони праці, техніки безпеки і пожежної безпеки*. URL: https://osvita-docs.com/node/41 (дата звернення: 12.06.2024).

ДОДАТКИ

Додаток А

Варіанти використання соціальної мережі Connectify

Таблиця А.1 – Варіанти використання для актанта «Зареєстрований користувач»

Актант	Варіант використання	Формулювання
Зареєстрований користувач	Перегляд головної сторінки	Дозволяє перегляд головної сторінки соціальної мережі
	Редагування імені	Дозволяє редагувати ім’я користувача за необхідності
	Редагування прізвища	Дозволяє редагувати прізвище користувача за необхідності
	Редагування місцезнаходження	Дозволяє редагувати власне місцезнаходження за необхідності
	Редагування дати народження	Дозволяє редагувати дату народження за необхідності
	Редагування зображення профілю	Дозволяє редагувати власне зображення профілю за необхідності
	Перегляд профілів користувачів	Дозволяє переглядати профілі інших користувачів соціальної мережі
	Перегляд власного профілю	Дозволяє переглядати власний профіль користувача
	Надсилання запиту на дружбу	Дозволяє надсилати запит на додавання в друзі іншим користувачам
	Приймання запиту на дружбу	Дозволяє прийняти запит на додавання в друзів від інших користувачів
	Видалення із списку друзів	Дозволяє видалити іншого користувача з друзів
	Створення статусів	Дозволяє писати статуси на сторінці
	Створення коментарів	Дозволяє писати коментарі під постами
	Вподобання статусів	Дозволяє вподобати статуси друзів
	Вподобання коментарів	Дозволяє вподобати коментарі друзів
	Вихід	Дозволяє вийти з свого профілю

Додаток Б

Програмний код моделей соціальної мережі Connectify

User.php:

```
namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
use function Symfony\Component\Translation\t;

class User extends Authenticatable implements MustVerifyEmail
{
    use HasApiTokens, HasFactory, Notifiable;

    protected $table = 'users';
    protected $fillable = [
        'email',
        'username',
        'password',
        'first_name',
        'last_name',
        'location',
        'dateOfBirth',
        'role',
        'is_active',
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];


    protected $casts = [
        'email_verified_at' => 'datetime',
    ];

    public function getName()
    {
        if ($this->first_name && $this->last_name) {
            return "{$this->first_name} {$this->last_name}";
        }

        if ($this->first_name) {
            return $this->first_name;
```

```
        }

        return null;
    }

    public function getNameOrUsername()
    {
        return $this->getName() ?: $this->username;
    }

    public function getFirstNameOrUsername()
    {
        return $this->first_name ?: $this->username;
    }

    public function statuses()
    {
        return $this->hasMany('App\Models\Status', 'user_id');
    }

    public function likes()
    {
        return $this->hasMany('App\Models\Like', 'user_id');
    }

    public function friendsOfMine()
    {
        return  $this->belongsToMany('App\Models\User',  'friends',
'user_id', 'friend_id');

    }

    public function friendOf()
    {
        return  $this->belongsToMany('App\Models\User',  'friends',
'friend_id', 'user_id');
    }

    public function friends()
    {
        return       $this->friendsOfMine()->wherePivot('accepted',
true)->get()
            ->merge($this->friendOf()->wherePivot('accepted',
true)->get());
    }

    public function friendRequests()
    {
        return       $this->friendsOfMine()->wherePivot('accepted',
false)->get();
    }
```

```
    public function friendRequestsPending()
    {
        return   $this->friendOf()->wherePivot('accepted',   false)->get();

    }


    public function hasFriendRequestPending(User $user)
    {
        return    (bool)$this->friendRequestsPending()->where('id', $user->id)->count();

    }

    public function hasFriendRequestReceived(User $user)
    {
        return   (bool)$this->friendRequests()->where('id',   $user->id)->count();

    }


    public function addFriend(User $user)
    {
        $this->friendOf()->attach($user->id);
    }

    public function deleteFriend(User $user)
    {
        $this->friendOf()->detach($user->id);
        $this->friendsOfMine()->detach($user->id);

    }

    public function acceptFriendRequest(User $user)
    {
        $this->friendRequests()->where('id',   $user->id)->first()->pivot->update([
            'accepted' => true
        ]);
    }

    public function isFriendWith(User $user)
    {
        return    (bool)$this->friends()->where('id',    $user->id)->count();
    }

    public function hasLikedStatus(Status $status)
    {
        return (bool)$status->likes
            ->where('likeable_id', $status->id)
```

```
            ->where('likeable_type', get_class($status))
            ->where('user_id', $this->id)
            ->count();
    }

    public function getAvatarsPath($user_id)
    {
        $path = "uploads/avatars/id/{$user_id}";

        if (!file_exists($path)) {
            mkdir($path, 0777, true);
        }

        return "/$path/";
    }

    public function clearAvatars($user_id)
    {
        $path = "uploads/avatars/id/{$user_id}";

        if (file_exists(public_path("/$path"))) {
            foreach (glob(public_path("/$path/*")) as $avatar)
                unlink($avatar);
        }
    }

    public function isAdmin()
    {
        return $this->role === 'admin';
    }

    public function isActive()
    {
        return $this->is_active;
    }
}
```

Status.php:

```
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Status extends Model
{
    use HasFactory;

    protected $fillable = [
        'body',
        'image'
    ];
```

```
    public function user()
    {
        return $this->belongsTo('App\Models\User', 'user_id');
    }

    public function scopeNotReply($query)
    {
        return $query->whereNull('parent_id');
    }

    public function replies()
    {
        return $this->hasMany('App\Models\Status', 'parent_id');
    }

    public function likes()
    {
        return $this->morphMany('App\Models\Like', 'likeable');
    }
}
```

Like.php:

```
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Like extends Model
{
    use HasFactory;
    protected $table = 'likeable';
    protected $fillable = ['user_id'];
    public function likeable()
    {
        return $this->morphTo();
    }
    function user()
    {
        return $this->belongsTo('App\Models\User', 'user_id');
    }
}
```

Додаток В

Програмний код контролерів соціальної мережі Connectify

AuthController.php:

```
namespace App\Http\Controllers;
use Illuminate\Support\Facades\Auth;
use Illuminate\Http\Request;
use App\Models\User;
class AuthController extends Controller
{
    public function getSignup()
    {
        return view('auth.signup');
    }
    public function postSignup(Request $request)
    {
    $this->validate($request, [
        'email' => 'required|unique:users|email|max:255',
        'username' => 'required|unique:users|alpha_dash|max:20',
        'password' => 'required|min:6',
        ]);
    User::create([
        'email' => $request->input('email'),
        'username' => $request->input('username'),
        'password' => bcrypt($request->input('password')),
        ]);
    return redirect()->route('home')->with('info','You are success
register');
    }
    public function getSignin()
    {
       return view('auth.signin');
    }
    public function postSignin(Request $request)
    {
        $this->validate($request,
        [
            'email' => 'required|max:255',
            'password' => 'required|min:6',
        ]);
        $credentials = $request->only('email','password');
$user = User::where('email', $request->input('email'))->first();

        if ($user && !$user->is_active) {
            return redirect()->back()->with('info', 'Your account
has been deactivated');
        }
```

```
        if          (!Auth::attempt($credentials,          $request-
>has('remember')))
        {
                return       redirect()->back()->with('info','Please
enter a correct username and password');
        }
        return redirect()->route('home')->with('info','You are log
in');
    }
    public function getSignout()
    {
        Auth::logout();
        return redirect()->route('home');
    }
}
```

FriendController.php:

```
namespace App\Http\Controllers;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
class FriendController extends Controller {
    public function getIndex()
    {
        $friends = Auth::user()->friends();
        $requests = Auth::user()->friendRequests();
        return view('friends.index',[
            'friends' => $friends,
            'requests' => $requests,
        ]);
    }
    public function getAddFriend($username)
    {
        $user = User::where('username', $username)->first();
        if (!$user) {
            return  redirect()->route('home')->with('info',  'User
not found');
        }
        if (Auth::user()->hasFriendRequestPending($user))
        {
                    return                                redirect()-
>route('profile.index',$username)->with('info', 'You have already
sent a friend request to this user' );
            }
        if (Auth::user()->isFriendWith($user)) {
            return    redirect()->route('profile.index',$username)-
>with('info', 'The user is already your friend' );
        }
        if (Auth::user()->id === $user->id) {
            return    redirect()->route('profile.index',$username)-
>with('info', 'You cannot add yourself as a friend' );
```

```
        }
        Auth::user()->addFriend($user);
        return        redirect()->route('profile.index',$username)-
>with('info', 'A friend request has been sent to this user' );
    }
    public function getAccept($username)
    {
        $user = User::where('username', $username)->first();
        if (!$user) {
            return  redirect()->route('home')->with('info',  'User
not found');
        }
        if (!Auth::user()->hasFriendRequestReceived($user)) {
            return redirect()->route('home');
        }
        Auth::user()->acceptFriendRequest($user);
        return        redirect()->route('profile.index',$username)-
>with('info', 'Now you are friends' );
    }
    public function postDelete($username)
    {
        $user = User::where('username', $username)->first();
        if (!Auth::user()->isFriendWith($user)) {
            return redirect()->back();
        }
        Auth::user()->deleteFriend($user);
        return    redirect()->back()->with('info','Deleted    from
friends');
    }
}
```

HomeController.php:

```
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use App\Models\Status;
class HomeController extends Controller {
    public function index()
    {
        if (Auth::check()) {
            $statuses     =     Status::notReply()->where(function
($query){
                return $query->where('user_id', Auth::user()->id)-
>orWhereIn('user_id', Auth::user()->friends()->pluck('id'));
            })
            ->orderBy('created_at', 'desc')
            ->paginate(4);
            return view('timeline.index', compact('statuses'));
        }
        return view('home');
    }
```

ProfileController.php:

```
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Models\User;
use Illuminate\Support\Facades\Auth;
use Intervention\Image\Facades\Image;
class ProfileController extends Controller
{
public function getProfile($username)
    {
        $user = User::where('username', $username)->first();

        if (!$user)
        {
            abort(404);
        }

        $statuses          =          $user->statuses()->notReply()-
>orderBy('created_at', 'desc')->get();

        $latestPhotos = Status::where('user_id', $user->id)
            ->whereNotNull('image')
            ->orderBy('created_at', 'desc')
            ->take(3)
            ->get();

        $allPhotos = Status::where('user_id', $user->id)
            ->whereNotNull('image')
            ->orderBy('created_at', 'desc')
            ->get();

        return view('profile.index', [
            'user' => $user,
            'statuses' => $statuses,
            'authUserIsFriend'           =>           Auth::user()-
>isFriendWith($user),
            'latestPhotos' => $latestPhotos,
            'allPhotos' => $allPhotos,
        ]);
    }
    public function postUploadAvatar(Request $request, $username)
    {
        $user = User::where('username', $username)->first();
        if ( ! Auth::user()->id === $user->id) {
            return redirect()->route('home');
        }
        if ( $request->hasFile('avatar'))
        {
            $user->clearAvatars($user->id);
            $avatar = $request->file('avatar');
            $filename    =    time()    .    '.'    .    $avatar-
>getClientOriginalExtension();
```

```
            Image::make($avatar)->resize(300,300)
            ->save(public_path(   $user->getAvatarsPath($user->id))
. $filename);
            $user = Auth::user();
            $user->avatar = $filename;
            $user->save();
        }
        return redirect()->back();
    }
}
```

SearchController.php:

```
namespace App\Http\Controllers;
use Illuminate\Support\Facades\DB;
use Illuminate\Http\Request;
use App\Models\User;
class SearchController extends Controller {
    public function getResults(Request $request)
    {
        $query = $request->input('query');
        if (!$query)
        {
            redirect()->route('home');
        }
        $users = User::where('first_name', 'like', "%$query%")
            ->orWhere('last_name', 'like', "%$query%")
            ->orWhere('username', 'like', "%$query%")
            ->get();;
        return view('search.results')->with('users',$users);
    }
}
```

StatusController.php:

```
namespace App\Http\Controllers;
use App\Models\Status;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
class StatusController extends Controller {
public function postStatus(Request $request)
    {
        $this->validate($request, [
            'status' => 'required|max:1000',
            'statusImage' =>
'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
        ]);

        $status = new Status();
        $status->body = $request->input('status');
        $status->user_id = Auth::user()->id;
```

```
        if ($request->hasFile('statusImage')) {
            $imagePath = $request->file('statusImage')-
>store('status_images', 'public');
            $status->image = $imagePath;
        }

        $status->save();

        return redirect()->route('home')->with('info', 'Record
successfully added');
    }
    public function postReply(Request $request, $statusId)
    {
       $this->validate($request, [
           "reply-{$statusId}" => 'required|max:1000'
       ]);
       $status = Status::find($statusId);
       if ( ! $status) redirect()->route('home');
       if ( ! Auth::user()->isFriendWith($status->user)
       && Auth::user()->id !== $status->user->id) {
           redirect()->route('home');
       }
       $reply = new Status();
       $reply->body = $request->input("reply-{$statusId}");
       $reply->user()->associate(Auth::user());
       $status->replies()->save($reply);
       return redirect()->back();
    }
    public function getLike($statusId)
    {
        $status = Status::find($statusId);
        if ( ! $status) redirect()->route('home');
        if ( ! Auth::user()->isFriendWith($status->user) ) {
            return redirect()->route('home');
        }
        if ( Auth::user()->hasLikedStatus($status) ) {
            return redirect()->back();
        }
        $status->likes()->create(['user_id' => Auth::user()->id]);
        return redirect()->back();
    }
}
```

AdminController.php:

```
namespace App\Http\Controllers;

use App\Models\Status;
use App\Models\User;
use Illuminate\Http\Request;
```

```
class AdminController extends Controller
{
    public function index()
    {
        $users = User::where('role', 'user')->paginate(5);
        $statuses = Status::paginate(5);
        return view('admin.index', compact('users', 'statuses'));
    }

    public function deactivateUser($id)
    {
        $user = User::findOrFail($id);
        $user->is_active = false;
        $user->save();
        return redirect('/admin')->with('success', 'The user has
been deactivated');
    }

    public function activateUser($id)
    {
        $user = User::findOrFail($id);
        $user->is_active = true;
        $user->save();
        return redirect('/admin')->with('success', 'The user has
been activated');
    }

    public function deleteStatus($id)
    {
        $status = Status::findOrFail($id);
        $status->delete();
        return     redirect('/admin')->with('success',     'Status
deleted');
    }
```

Додаток Д

Програмний код файлу web.php соціальної мережі Connectify

```
use App\Http\Controllers\ProfileController;
use App\Http\Controllers\StatusController;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\HomeController;
use App\Http\Controllers\AuthController;
use App\Http\Controllers\SearchController;
use App\Http\Controllers\FriendController;

Route::get('/', [HomeController::class, 'index'])->name('home');

Route::get('/alert', function () {
   return redirect()->route('home')->with('info', 'You can login!');
});

Route::get('/signup', [AuthController::class, 'getSignup'])->middleware('guest')->name('auth.signup');
Route::post('/signup', [AuthController::class, 'postSignup'])->middleware('guest');
Route::get('/signin', [AuthController::class, 'getSignin'])->middleware('guest')->name('auth.signin');
Route::post('/signin', [AuthController::class, 'postSignin'])->middleware('guest');
Route::get('/signout', [AuthController::class, 'getSignout'])->name('auth.signout');

Route::get('/search', [SearchController::class, 'getResults'])->name('search.results');

Route::get('/user/{username}', [ProfileController::class, 'getProfile'])->name('profile.index');
```

```
Route::get('/profile/edit', [ProfileController::class, 'getEdit'])->middleware('auth')->name('profile.edit');
Route::post('/profile/edit', [ProfileController::class, 'postEdit'])->middleware('auth')->name('profile.edit');
Route::post('/upload-avatar/{username}', [ProfileController::class, 'postUploadAvatar'])->middleware('auth')->name('profile.upload-avatar');

Route::get('/friends', [FriendController::class, 'getIndex'])->middleware('auth')->name('friend.index');
Route::get('/friends/add/{username}', [FriendController::class, 'getAddFriend'])->middleware('auth')->name('friend.add');
Route::get('/friends/accept/{username}', [FriendController::class, 'getAccept'])->middleware('auth')->name('friend.accept');
Route::post('/friends/delete/{username}', [FriendController::class, 'postDelete'])->middleware('auth')->name('friend.delete');

Route::post('/status', [StatusController::class, 'postStatus'])->middleware('auth')->name('post.status');
Route::post('/status/{statusId}/reply', [StatusController::class, 'postReply'])->middleware('auth')->name('status.reply');

Route::get('/status/{statusId}/like', [StatusController::class, 'getLike'])->middleware('auth')->name('status.like');

Route::middleware(['auth', 'admin'])->group(function () {
    Route::get('/admin', [AdminController::class, 'index']);
    Route::post('/admin/deactivate-user/{id}', [AdminController::class, 'deactivateUser']);
    Route::post('/admin/activate-user/{id}', [AdminController::class, 'activateUser']);
    Route::post('/admin/delete-status/{id}', [AdminController::class, 'deleteStatus']);
});
```