Національний університет «Полтавська політехніка імені Юрія Кондратюка»
(повне найменування вищого навчального закладу)
Навчально-науковий інститут інформаційних технологій та робототехніки
(повна назва факультету)
Кафедра комп'ютерних та інформаційних технологій і систем
(повна назва кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

_____магістр_____
(рівень вищої освіти)
на тему
_____Розроблення інтелектуальної соціальної мережі_____ __
_____
_____

Виконав: студент 6 курсу, групи 601-ТН
спеціальності
122 Комп'ютерні науки._____
(шифр і назва напряму)
_____Зоураб Ю.Р._____
(прізвище та ініціали)
Керівник _____Дмитренко Т.А._____
(прізвище та ініціали)
Рецензент _____
(прізвище та ініціали)

Полтава – 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

«ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ

КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

спеціальність 122 «Комп'ютерні науки»

на тему

«Розроблення інтелектуальної соціальної мережі»

Студента групи 601-ТН Зоураба Юсіфа Рафат

Керівник роботи
кандидат технічних наук,
доцент Дмитренко Т.А.

Завідувач кафедри
кандидат технічних наук,
доцент Головко Г.В.

Полтава – 2021

РЕФЕРАТ

Кваліфікаційна робота магістра: 83 с., 34 малюнків, 2 додатки, 13 джерел.

Об’єкт дослідження: соціальна мережа.

Мета роботи: розроблення соціальної мережі

Методи: проектування та розробка бази даних, шаблону та функціоналу інформаційної системи.

Ключові слова: соціальна мережа, чат бот.

SUMMERY

Qualifying work of the master: 83 pages, 34 drawings, 2 appendices, 13 sources.

Object of study: social network.

The goal of the work: social network development

Methods: design and development of a database, template and functionality.

Keywords: social network, chat bot.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

SRP – Single Responsibility Principle

CLI – Command-line Interface

JS – JavaScript

PHP – hypertext Preprocessor

SQL – Structured Query Language

ER-діаграма – Entity-relationship діаграма

БД – базаданих

СУБД – Система управління базами даних

ОС – операційна система

ORM – Object-Relational Mapping

CSS – Cascading Style Sheets

MVC – Model View Controller

CMS – Control Management System

TS - TypeScript

ВСТУП

Соціальнімережі – цечудоваплатформадлялюдей, щобспілкуватисязісвоїмиблизькими. Це допомагає розширити спілкування та встановити зв'язки з людьми по всьому світу. Хоча люди вважають, що соціальні мережі шкідливі, вони також дуже корисні.

Крім того, ми можемо класифікувати сайти соціальних мереж за веденням блогів, відеоблогів, подкастингів тощо. Ми використовуємо соціальні мережі для різних цілей. Це нам дуже допомагає; однак це також дуже небезпечно. Ми повинні контролювати використання сайтів соціальних мереж і обмежувати їх використання, щоб вони не захопили наше життя.

Соціальні мережі зараз повсюдно. Іншими словами, вони захопили майже всі сфери життя. Вони мають як переваги, так і недоліки. Якщо говорити про освітню сферу, то ці сайти покращують освіту, впливаючи на учнів. Вони можуть досліджувати різні теми для своїх проектів.

Ми не уявляємо свого життя без Інтернету. Там ми можемо знайти всю необхідну інформацію. Ми щодня використовуємо популярні сервіси та джерела. Одним з найпопулярніших джерел є соціальні мережі. Соціальна мережа – дуже корисний винахід 21 століття. Сьогодні соціальними мережами користуються мільйони людей. Це інтернет-сайт, який дає нам можливість підтримувати різноманітні контакти, підтримувати дружні стосунки з однокласниками, знайомими. Більш функціональні соціальні мережі дозволяють переглядати фотографії та відеофільми, слухати музику тощо. Є можливість входити в різні цікаві групи. Існують соціальні мережі, що об'єднують людей відповідно до загального інтересу, наприклад, гравці комп'ютерних ігор об'єднані в соціальну мережу.

Люди звикли до серфінгу в Інтернеті, тому що знайти будь-яке бажання швидко і дуже легко. Вам не потрібно думати про пропозиції та ідеї.

Принаймні, що вам потрібно, це вміти правильно друкувати. Мені здається, що соціальні мережі дуже допомагають інвалідам та самотнім людям. Вони можуть створити обліковий запис і почати спілкуватися з іншими ідентичними партнерами. Більшість соціальних мереж використовують люди для зустрічі з друзями зі схожими інтересами та хобі. Дуже часто близькі люди знаходять один одного в Інтернеті. Хоча люди живуть далеко один від одного, будь-яка мережа може їх об'єднати. Але чому соціальні мережі настільки поширені та широко використовуються? Оскільки ми живемо в епоху, коли технології знаходяться на піку і постійно розвиваються, такі речі, як читання книг в Інтернеті, покупки в Інтернеті та спілкування з контактами по всьому світу, які були неможливі сто років тому, тепер є частиною повсякденного життя. . Найважливішою частиною сучасного світу є швидкість – швидкість отримання інформації.

Привабливість соціальних мереж полягає насамперед у їх доступності – реєстрація в більшості популярних соціальних мереж безкоштовна. Доступність і простота використання робить такі сайти, як Facebook і MySpace, привабливими. По-друге, соціальні мережі створюють можливість спілкуватися з друзями, знайомими чи навіть незнайомими людьми по всьому світу, що є життєво важливим у сучасному світі. Якщо в добу до Інтернету комунікація вимагала незліченної кількості часу, енергії та грошей, то сьогодні це лише один клік. Крім того, соціальні мережі дозволяють людям стежити за соціальними подіями зі свого «списку друзів». Нарешті, окрім виконання завдання соціальної мережі, такі сайти, як Facebook, надають своїм користувачам розвагу. Візьмемо, наприклад, програми Facebook, які містять усі види вікторин та ігор для користувачів. Таким чином, кількість користувачів Facebook – мільйони.

Написання та отримання кореспонденції за допомогою пошти – тривалий процес. Це потребує часу та грошей. Більше того, лист може бути втрачений. Але електронну пошту неможливо втратити і не займає багато часу. Соціальна

мережа – дуже корисний винахід 21 століття. Завдяки сучасним технологіям люди можуть не тільки писати різного роду листи своїм партнерам по всьому світу, а й розмовляти та бачити їх за допомогою веб-камери. Але з іншого боку, соціальні мережі – велика біда. Вони змусили людей забути традиційну розмову по душам. Іноді ви не можете побачити очі своїх партнерів і не відчути їх емоції. Крім того, соцмережі рясніють агресивними, жорстокими, жорстокими та небезпечними відео для дітей.

Багато невпевнених людей намагаються підняти впевненість у собі за допомогою соціальних мереж. Можна сказати, що в Інтернеті, а саме на сайтах соціальних мереж, ви ніколи не можете бути впевнені в тому, хто «слухає». Я вважаю, що мережами можуть користуватися дорослі люди та діти віком від чотирнадцяти років. Щодо мене, то я з великим задоволенням користуюся соціальними мережами, адже вони мені допомагають у самоосвіті та спілкуванні.

РОЗДІЛ 1

АНАЛІТИЧНИЙ ОГЛЯД ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1 Опис предметної області

Термін соціальна мережа відноситься до використання веб-сайтів соціальних мереж, щоб залишатися на зв’язку з друзями, родиною, колегами, клієнтами або клієнтами. Соціальні мережі можуть мати соціальну мету, ділову мету або обидві цілі через такі сайти, як Facebook, Twitter, LinkedIn та Instagram. Соціальні мережі також є важливою базою для маркетологів, які прагнуть залучити клієнтів. Facebook залишається найбільшою та найпопулярнішою соціальною мережею: 2,8 мільярда людей користуються платформою щомісяця, станом на 31 грудня 2020 року. За даними Statista наступними за популярністю є Instagram, Facebook Messenger, Twitter і Pinterest.

Послуги соціальних мереж відрізняються за форматом і кількістю функцій. Вони можуть включати цілий ряд нових інформаційних та комунікаційних інструментів, які працюють на настільних комп’ютерах і ноутбуках, на мобільних пристроях, таких як планшетні комп’ютери та смартфони. Це може включати цифрові фото/відео/обмін і щоденникові записи в Інтернеті (блог). Розробники та користувачі інколи вважають послуги спільноти в Інтернеті послугами соціальних мереж, хоча в ширшому сенсі послуги соціальної мережі зазвичай надають послуги, орієнтовані на індивіда, тоді як послуги онлайн спільноти орієнтовані на групи. Загалом визначені як «веб-сайти, які сприяють побудові мережі контактів для обміну різними типами контенту в Інтернеті», сайти соціальних мереж надають простір для взаємодії, яка може продовжитися за межами особистої взаємодії. Ці комп’ютерні взаємодії зв’язують учасників різних мереж і можуть допомогти у створенні, підтримці та розвитку нових соціальних і професійних стосунків.

1.2 Аналіз області застосування об’єкту проєктування

Соціальні мережі передбачають розвиток і підтримку особистих і ділових відносин за допомогою технологій. Це робиться за допомогою сайтів соціальних мереж, таких як Facebook, Instagram та Twitter. Ці сайти дозволяють людям і корпораціям спілкуватися один з одним, щоб вони могли розвивати відносини та ділитися інформацією, ідеями та повідомленнями.

Члени сім’ї, які знаходяться далеко один від одного, можуть залишатися на зв’язку через особисті соціальні мережі, як-от Facebook. Вони можуть ділитися фотографіями та новинами про те, що відбувається в їхньому житті. Люди також можуть спілкуватися з іншими (зокрема, незнайомцями), які поділяють ті ж інтереси. Люди можуть знайти один одного за допомогою груп, списків і використання хештегів.

Соціальні мережі зазвичай використовуються маркетологами для підвищення впізнаваності бренду та заохочення лояльності до бренду. Оскільки це робить компанію більш доступною для нових клієнтів і більш впізнаваною для наявних, маркетинг у соціальних мережах допомагає просувати голос і контент бренду.

Наприклад, постійний користувач Twitter може вперше дізнатись про компанію через стрічку новин і вирішити купити товар чи послугу. Чим більше людей знайомі з брендом компанії, тим більше у компанії шансів знайти та утримати нових клієнтів.

Маркетологи використовують соціальні мережі, щоб підвищити коефіцієнт конверсії. Створення підписки надає доступ до нових, недавніх і старих клієнтів і взаємодію з ними. Публікація публікацій у блозі, зображень, відео чи коментарів у соціальних мережах дозволяє читачам відреагувати, відвідати веб-сайт компанії та стати клієнтами.

Соціальні мережі важливі, оскільки вони дозволяють людям розвивати стосунки з іншими, з якими вони не могли б зв’язатися інакше. Вони також допомагають підвищити продуктивність бізнесу, якщо використовуються для зв’язків з громадськістю, маркетингу та реклами.

1.3 Функціональні вимоги

Для того, щоб визначити функціонал модулів розглянемо функції, які можуть виконувати користувачі системи.

Таблиця 1.1 – Функції користувача.

№ з/п	**Функція**	**Виконавець**
1	2	3
	ПЕРЕГЛЯД. Перегляд списку публікацій.	Користувач
	РЕДАГУВАННЯ. Додавання, редагування та видалення публікацій.	Користувач
	ПЕРЕГЛЯД. Перегляд даних профілю.	Користувач
	РЕДАГУВАННЯ. Редагування даних профілю.	Користувач
	ПЕРЕГЛЯД. Перегляд списку діалогів.	Користувач
	РЕДАГУВАННЯ. Додавання повідомлення в діалоги.	Користувач
	ПЕРЕГЛЯД. Перегляд кількості лайків в публікаціях.	Користувач
	РЕДАГУВАННЯ. Додавання лайків в публікаціях.	Користувач
	ПЕРЕГЛЯД. Перегляд кількості переглядів в публікаціях.	Користувач
	РЕДАГУВАННЯ. Додавання переглядів в публікаціях.	Користувач
	ПЕРЕГЛЯД. Перегляд коментарів.	Користувач
	РЕДАГУВАННЯ. Додавання коментарів.	Користувач
	ПЕРЕГЛЯД. Перегляд друзів.	Користувач
	РЕДАГУВАННЯ. Додавання списку друзів.	Користувач

1.4. Порівняння з існуючими проєктами

Тепер виведемо та порівняємо ехнічний стек схожих проєктів.

Таблиця 1.2 – Порівняння стеків.

Характеристика	**Instagram**	**Facebook**	**Мій проєкт**
1	2	3	4
Python	+	-	-
PHP	-	+	+
jQuery	+	?	-
PostgreSQL	+	?	-
Redis	+	?	+
React	+	+	-
Vue	-	-	+
GraphQL	+	+	-
Java	+	+	-
C	+	+	-
Webpack	+	-	+
Babel	+	-	+
Jest	+	-	+
Sentry	+	-	-
AWS	+	-	-

1.5. Огляд існуючих соціальних мереж

Соціальна мережа Instagram.Instagram вперше був наданий публіці у 2010 році. Його авторами є два розробники з Сан-Франциско – Кевін Систром і Майк Крігер. Додаток був призначений лише для пристроїв компанії Apple і мав

обмежений функціонал, а саме дозволяв ділитися невеликими фотознімками на своїй персональній сторінці.

З розвитком технологій і виходом нових версій гаджетів на IOS та Android, продовжував збільшуватися і функціонал.

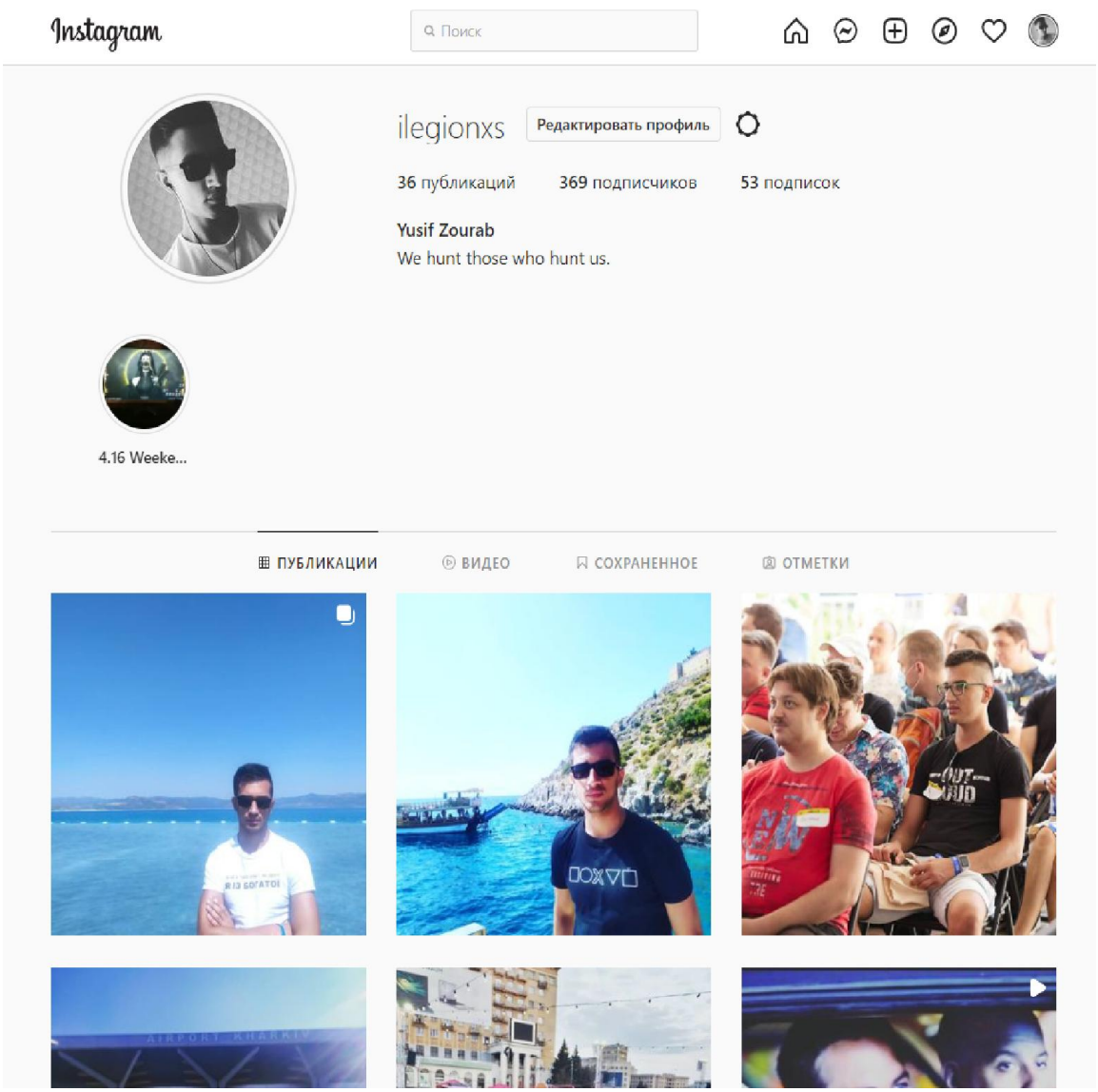


Рисунок 1.1 – Сторінка користувача у системі Instagram.

З часом багато людей змінило свою думку стосовно цієї мережі й активно користуються всіма можливостями Instagram. Зараз Instagram надає наступні функції:

- можливість ділитися фото й відеозаписами;
- додавати опис до зображень;

- знімати короткі відео та фото в режимі реального;
- редагувати зображення, додавати текст, наліпки, графічні об’єкти;
- проводити прямі ефіри;
- слідкувати за новинами та публікаціями;
- спілкування за допомогою коментарів в публікаціях або в персональних чатах;
- розвивати свій бізнес.

Соціальна мережа Facebook. Facebook – найпопулярніша соціальна мережа на сьогодні. Вона була заснована в далекому 2004 році Марком Цукербергом. З часом ім’я було змінене, а доступ до мережі поширився на весь світ.

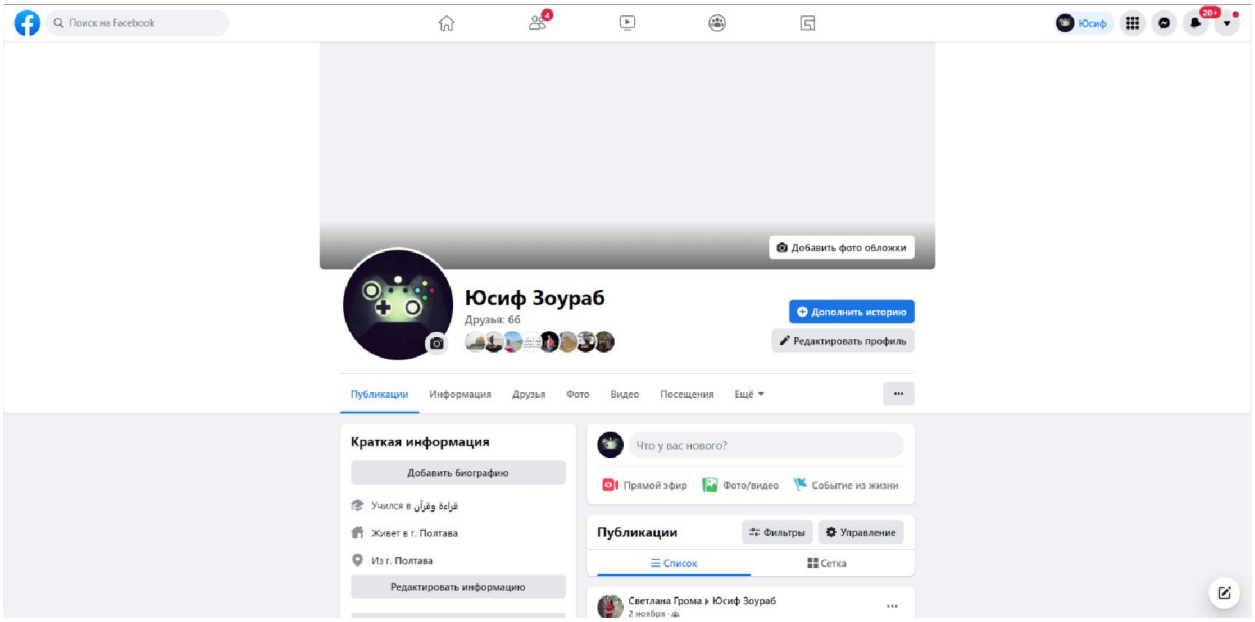


Рисунок 1.2 – Сторінка користувача у системі Facebook.

Facebook дозволяє користувачам створювати профілі з персональними даними, обмінюватися текстовими повідомленнями, організовувати спільноти, запрошуючи друзів. На той час це було своєрідним проривом у можливостях, що і дозволило мережі завоювати велику кількість прихильників.

У 2010 році Марк Цукерберг зробив ще один крок з метою розширення та надання нових послуг мережі, а саме підписав угоду про впровадження в мережу сервісу Skype. В тому ж році була запущена власна поштова служба.

У 2011 р. була запущена україномовна версія сайту. З цього часу користувачі могли змінювати мову системи на українську, а при запуску сайту з України, система автоматично її використовувала.

На сьогодні Facebook є головною соціальною мережею у світі з величезною аудиторією.

Соціальна мережа «ВКонтакте».Структура і функціонал соціальної мережі «ВКонтакте» досить схожі на Facebook. А це майже гарант успіху, що робить цю мережу найпопулярнішою в країнах СНД.

Перші версії соціальної мережі «ВКонтакті» з'явилися влітку 2006 року. Її автором був 22-річний Павло Дуров. Він досить довго формулював ідею для своєї системи, однак коли зустрів свого друга з Америки, який познайомив його з мережею Facebook, створення системи було лише питанням часу. Більшість ідей «ВКонтакті» піддивилося у свого більш відомого аналога, однак головною метою у процесі розроблення було задоволення потреб користувачів.

З перших днів сайт позиціонував себе як система орієнтована на користувача і утримувалась від реклами. Однак з часом з'явилася проблема у фінансах. Затрати на хостинг фото та відео безупинно зростали. Потреби у доходах збільшувалися і до функціонала додалися платні подарунки іншим користувачам.

Цей варіант розв'язання проблеми хоч і був дуже цікавий і сподобався багатьом прихильникам. І все ж виручка від нього була недостатньою і тому у 2008 році у системі з'явилася платна реклама, а також можливість додавання різник додатків, таких як: ігри та платні сервіси. У цих додатках розробники заробляють гроші від продажу якихось атрибутів чи можливостей, і частину прибутку віддають соціальній мережі.

З 2014 року інтернет-компанія Mail.Ru Group стала єдиним власником соціальної мережі «ВКонтакті».

За час володіння мережею іншою компанією, відбулися деякі зміни в її роботі. Однією з головних була зміна дизайну соціальної мережі. Багатьом користувачам не сподобався новий дизайн і вони, навіть, почали збирати петиції на повернення до старого.

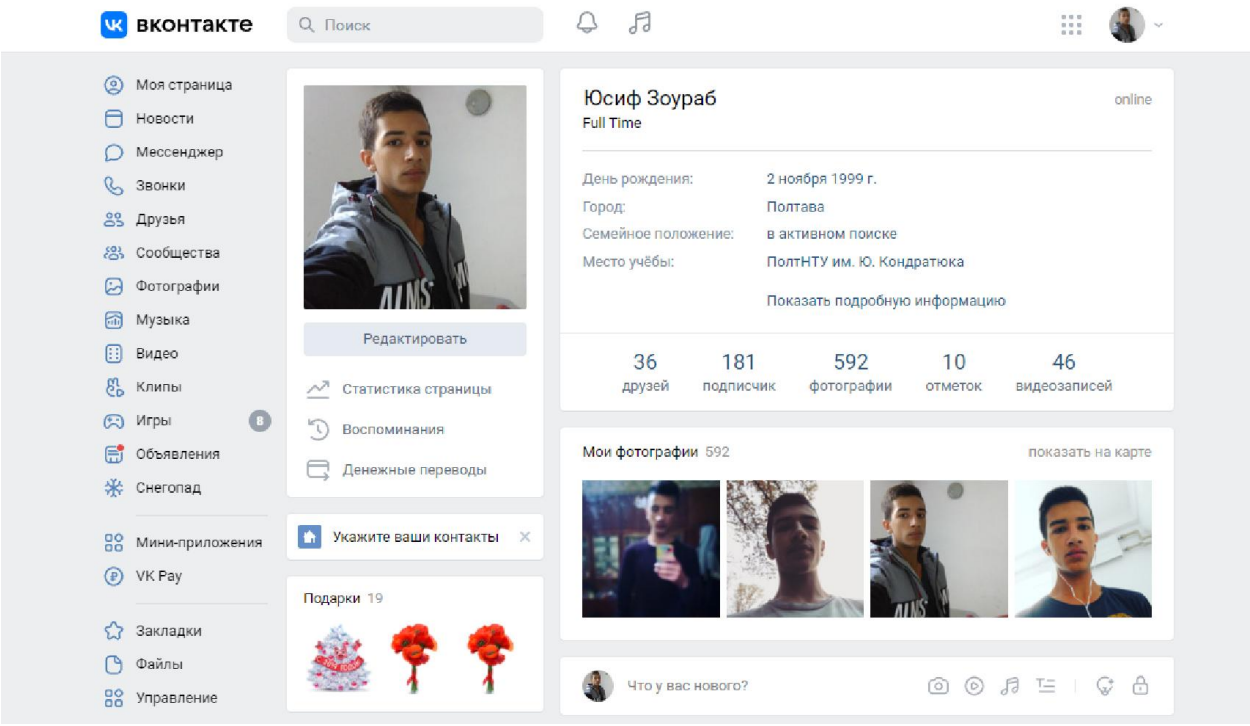


Рисунок 1.4 – Сторінка користувача у системі «Вконтакте».

Зараз «ВКонтакті» є лідером у Білорусі, Росії серед соціальних мереж за відвідуваністю. В Україні ця система була б також найпопулярнішою, якби не заборона російських ресурсів.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Функціонал та структура системи

Система, що розроблюється, була розбита на модулі. Такий підхід дозволяє значно зменшити час для розробки та підтримки, необхідний на впровадження системи.

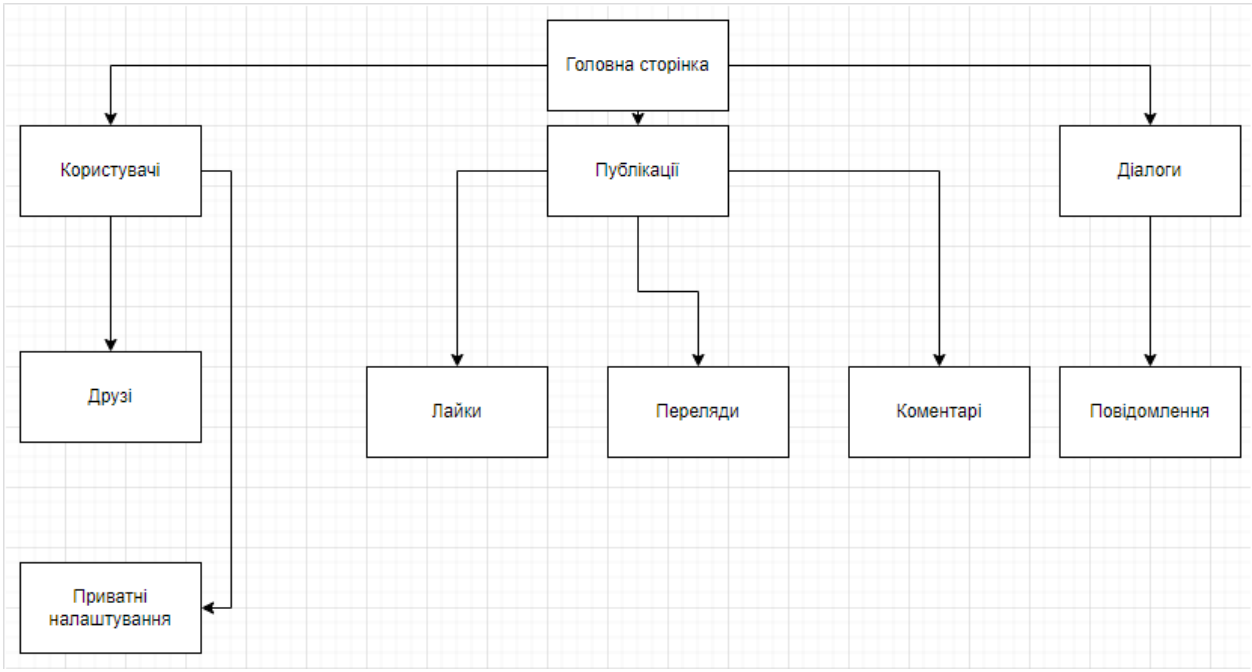


Рисунок 2.1 – Структура системи.

Розглянемо структуру сайту детальніше.

Почнемо ми з модуля «Користувачі» (Рисунок 2.2). Цей модуль включає в себе можливість перегляду списку користувачів данної системи, створення та редагування та інші підмодулі, а саме: Друзі та приватні налаштування.

Модуль «Користувачі» являється невідємною частиною підмодулів. Підмодуль «Друзі» ідентифікує користувачів як друзів для певних користувачів. Підмодуль «Приватні налаштування» має інформацію про приватні налаштування користувача, такі як, можливість проглядати іншими користувачами інформацію про ті інші дані на строрінці такого користувача,

додавання в друзі такого користувача та можливість написання повідомлень такому користувачеві.

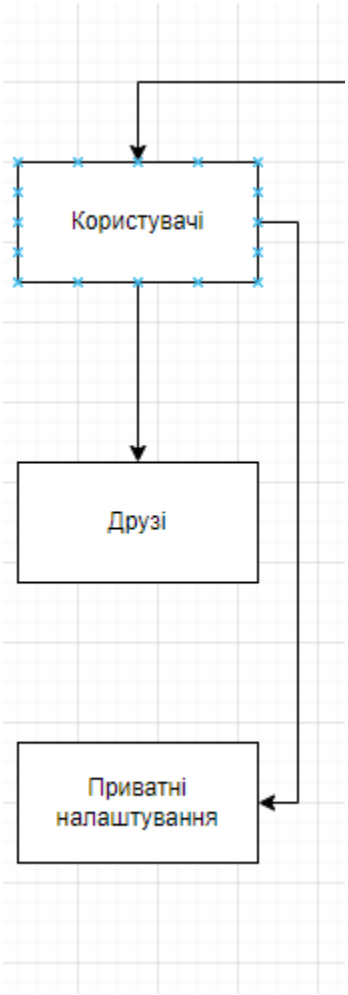


Рисунок 2.2 – Модуль «Користувачі».

Видалення користувачів в цей модуль не включено, так як ця функція тут не потрібна.

Наступній модуль про який я хочу розповісти це модуль «Публікації» (Рисунок 2.3). В цьому модулі реалізовані перегляд списку публікацій, їх створення, редагування та видалення. Також цей модуль включає три підмодулі. Ці підмодулі не являются невідємною частиною модуля «Публікації». Їх можна використовувати для будь якої сутності, але треба мати на увазі, что ці підмодулі не можуть бути незалежними від всього.

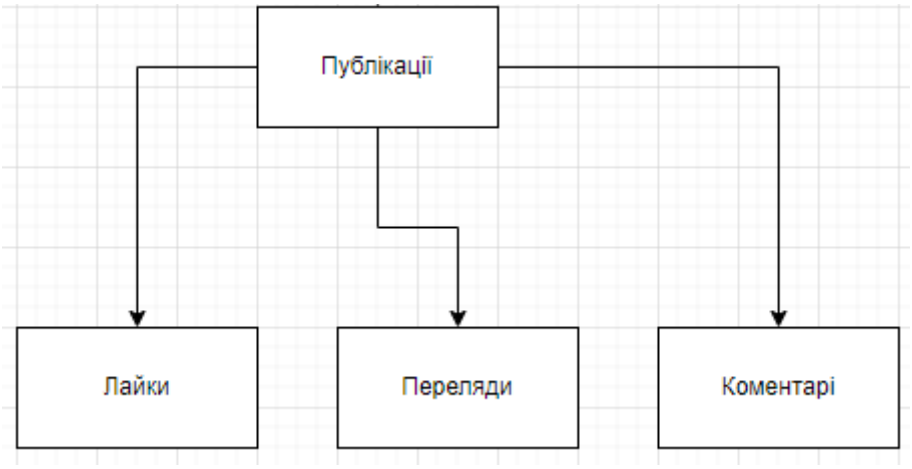


Рисунок 2.3 – Модуль «Публікації».

Модуль «Діалоги» (Рисунок 2.4). Цей модуль має також один підмодуль, а саме «Повідомлення». Модуль «Діалог» має інформацію про назву діалога, дату його створення та інше. А от підмодуль «Повідомлення», має зв’язок з модулем «Діалог» та включає інформацію про повідомлення від користувачів.

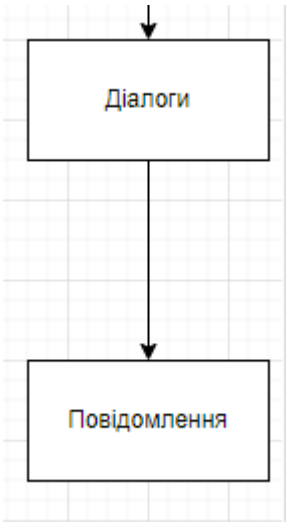


Рисунок 2.4 – Модуль «Діалоги».

2.2 Проектування бази даних

База даних містить масову інформацію, що зберігається в структурі, що полегшує пошук та дослідження відповідної інформації. Добре розроблена база даних містить точну й актуальну інформацію для аналізу та звітності.

Проектування бази даних — це набір кроків, які допомагають створювати, впроваджувати та підтримувати системи управління даними підприємства. Основною метою проектування бази даних є створення фізичних та логічних моделей проектів для запропонованої системи баз даних.

Хороший процес проектування бази даних регулюється певними правилами. Перше правило при створенні дизайну бази даних — уникати надмірності даних. Це витрачає простір і збільшує ймовірність помилок і розбіжностей у базі даних. Друге правило полягає в тому, що точність і вичерпність інформації є обов'язковими. База даних, що містить помилкову інформацію, призведе до неправильного аналізу та звітності. Отже, це може ввести в оману осіб, які приймають рішення, і негативно вплинути на результати діяльності компанії. Тому при розробці бази даних для організації важливо пам'ятати про правила.

Добре розроблена база даних – це та, яка:

- розподіляє дані в таблиці на основі конкретних предметних областей, щоб зменшити надмірність даних;
- надає базі даних інформацію, необхідну для зв'язування даних у таблицях;
- забезпечує підтримку та гарантує точність і надійність даних;
- забезпечує ваші вимоги до обробки інформації та звітності;
- функціонує інтерактивно з операторами бази даних.

Проектування бази даних визначає структуру бази даних, яка використовується для планування, зберігання та керування інформацією. Щоб забезпечити точність даних, ви повинні створити базу даних, яка зберігає лише релевантну та цінну інформацію.

Добре розроблена база даних необхідна для забезпечення узгодженості інформації, усунення зайвих даних, ефективного виконання запитів і

підвищення продуктивності бази даних. Методологічний підхід до проектування бази даних заощадить вам час на етапі розробки бази даних.

Надійність даних залежить від структури таблиці, тоді як створення первинного та унікального ключів гарантує однорідність інформації, що зберігається. Ви можете уникнути реплікації даних, сформувавши таблицю ймовірних значень і використавши ключ для позначення значення. Таким чином, щоразу, коли значення змінюється, зміна відбувається лише один раз в основній таблиці.

Оскільки загальна продуктивність бази даних залежить від її дизайну, хороший дизайн бази даних використовує прості запити та швидшу реалізацію. Крім того, його легко підтримувати та оновлювати. З іншого боку, коли база даних погано спроектована, навіть незначні переривання можуть зашкодити збереженим подіям, представленням та утилітам.

Проектування бази даних зазвичай починається з визначення мети вашої бази даних. Відповідні дані потім збираються і систематизуються в таблиці. Далі ви вказуєте первинні ключі та аналізуєте зв’язки між різними таблицями для ефективного проектування даних. Після уточнення таблиць останнім кроком є застосування правил нормалізації для стандартизації таблиць.

Розглянемо ці етапи проектування бази даних:

- визначити метубази даних;
- знайдти і об’єднайти необхідні дані;
- розподілити дані в таблиці;
- змінити елементи даних на стовпці;
- визначити первинні ключі;
- визначити зв’язок між таблицями;
- покращити дизайн бази даних;
- виконати правила нормалізації.

Тепер перейдемо до проектування бази даних, а саме визначимо структуру бази та логічні зв'язки і зобразимо усе це на ER-діаграмі.

Визначимо основні сутності ІС:

Сутність «Користувачі» містить інформацію про користувачів системи, вона має такі атрибути:

- id – ідентифікатор користувача, має тип bigInt(20), є первинним ключем та не може бути NULL;
- name – ім’я користувача, має тип varchar(255) та не може бути NULL;
- email – електронна адреса користувача, має тип varchar(255) та не може бути NULL;
- email_verified_at – час підтвердження електронної адреси користувача, має тип timestamp та може бути NULL;
- password – зашифрована версія паролю користувача, має тип varchar(255) та не може бути NULL;
- avatar – посилання на картинку користувача, має тип varchar(255) та може бути NULL;
- remember_token – довготривалий токен користувача, має тип varchar(100) та може бути NULL;
- created_at – час створення запису, має тип timestamp та може бути NULL;
- updated_at – час оновлення запису, має тип timestamp та може бути NULL.

Сутність «Коментарі» містить інформацію про коментарі до публікацій і т.д., вона має такі атрибути:

- id – ідентифікатор коментаря, має тип bigInt(20), є первинним ключем та не може бути NULL;
- user_id – ідентифікатор користувача, має тип bigInt(20), є вторинним ключем та не може бути NULL;

- commentable_id – ідентифікатор сутності, має тип bigInt(20) та не може бути NULL;
- commentable_type – тип сутності, має тип varchar(255) та не може бути NULL;
- text – текст коментарія, має тип text та не може бути NULL;
- created_at – час створення запису, має тип timestamp та може бути NULL;
- updated_at – час оновлення запису, має тип timestamp та може бути NULL.

Сутність «Повідомлення діалогів» містить інформацію про повідомлення які відносяться до діалогів, вона має такі атрибути:

- id – ідентифікатор повідомлення, має тип bigInt(20), є первинним ключем та не може бути NULL;
- dialog_id – ідентифікатор діалога, має тип bigInt(20), є вторинним ключем та не може бути NULL;
- user_id – ідентифікатор користувача, має тип bigInt(20), є вторинним ключем та не може бути NULL;
- text – текст коментаря, має тип text та не може бути NULL;
- read_at – час коли користувач переглянув повідомлення, має тип timestamp та може бути NULL;
- created_at – час створення запису, має тип timestamp та може бути NULL;
- updated_at – час оновлення запису, має тип timestamp та може бути NULL.

Сутність «Діалоги» містить інформацію про діалоги, вона має такі атрибути:

- id – ідентифікатор повідомлення, має тип bigInt(20), є первинним ключем та не може бути NULL;

- owner_id – ідентифікатор користувача, має відношення до ідентифікації користувача який створив діалог, має тип bigInt(20), є вторинним ключем та не може бути NULL;
- title – назва діалогу, має тип varchar(255) та не може бути NULL;
- created_at – час створення запису, має тип timestamp та може бути NULL;
- updated_at – час оновлення запису, має тип timestamp та може бути NULL.

Сутність «Друзі» містить інформацію про зв’язок між користувачами в якості друзів, вона має такі атрибути:

- id – ідентифікатор, має тип bigInt(20), є первинним ключем та не може бути NULL;
- user_id – ідентифікатор користувача, має тип bigInt(20), є вторинним ключем та не може бути NULL;
- friend_id – ідентифікатор користувача який вважається другом, має тип bigInt(20), є вторинним ключем та не може бути NULL;
- created_at – час створення запису, має тип timestamp та може бути NULL;
- updated_at – час оновлення запису, має тип timestamp та може бути NULL.

Сутність «Лайки» містить інформацію про лайки користувачів відповідно до вказаних сутностей, вона має такі атрибути:

- id – ідентифікатор, має тип bigInt(20), є первинним ключем та не може бути NULL;
- user_id – ідентифікатор користувача, має тип bigInt(20), є вторинним ключем та не може бути NULL;
- likeable_id – ідентифікатор сутності, має тип bigInt(20) та не може бути NULL;

- likeable_type – тип сутності, має тип varchar(255) та не може бути NULL;
- created_at – час створення запису, має тип timestamp та може бути NULL;
- updated_at – час оновлення запису, має тип timestamp та може бути NULL.

Сутність «Публікації» містить інформацію про публікації користувачів, вона має такі атрибути:

- id – ідентифікатор, має тип bigInt(20), є первинним ключем та не може бути NULL;
- author_id – ідентифікатор користувача, має тип bigInt(20), є вторинним ключем та не може бути NULL;
- title – назва публікації, має тип varchar(255) та не може бути NULL;
- text – текст публікації, має тип text та не може бути NULL;
- created_at – час створення запису, має тип timestamp та може бути NULL;
- updated_at – час оновлення запису, має тип timestamp та може бути NULL.

Сутність «Приватні налаштування» містить інформацію про приватні налаштування користувачів, вона має такі атрибути:

- id – ідентифікатор, має тип bigInt(20), є первинним ключем та не може бути NULL;
- user_id – ідентифікатор користувача, має тип bigInt(20), є вторинним ключем та не може бути NULL;
- profile_display_mode – параметр котрий визначає доступність відображення даних користувача іншим користувачам котрі не являються друзями, має тип smallint та не може бути NULL;

- add_friends_mode – параметр котрий визначає доступність в функціоналі додавання користувача в друзі, має тип smallint та не може бути NULL;
- message_writing_mode – параметр котрий визначає доступність в функціоналі написання повідомлень користувачу котрий не являється другом, має тип smallint та не може бути NULL;
- created_at – час створення запису, має тип timestamp та може бути NULL;
- updated_at – час оновлення запису, має тип timestamp та може бути NULL.

Сутність «Ролі» містить інформацію про ролі доступні для користувачів, вона має такі атрибути:

- id – ідентифікатор, має тип bigInt(20), є первинним ключем та не може бути NULL;
- title – назва ролі, має тип varchar(255) та не може бути NULL;
- created_at – час створення запису, має тип timestamp та може бути NULL;
- updated_at – час оновлення запису, має тип timestamp та може бути NULL.

Сутність «Перегляди» містить інформацію про перегляди користувачів відповідно до вказаних сутностей, вона має такі атрибути:

- id – ідентифікатор, має тип bigInt(20), є первинним ключем та не може бути NULL;
- user_id – ідентифікатор користувача, має тип bigInt(20), є вторинним ключем та не може бути NULL;
- viewable_id – ідентифікатор сутності, має тип bigInt(20) та не може бути NULL;

- viewable_type – тип сутності, має тип varchar(255) та не може бути NULL;
- created_at – час створення запису, має тип timestamp та може бути NULL;
- updated_at – час оновлення запису, має тип timestamp та може бути NULL.

Також використовуються ще декілька таблиць, але вони не являються окремими сутностями. Такі таблиці являються допоміжними для основних сутностей.

РОЗДІЛ 3

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Вибір платформи

Комп'ютерна платформа – це будь-яка існуюча виконавча система, в якій може виповняться якесь програмне забезпечення.

В якості платформи були обрані всі системи які можуть отримати доступ до інтернету. Так як середовище виконання був обраний браузер, то будь-яка платформа в якій є доступ до мережі інтернет, може використовувати цей продукт.

Через таку середовища виконання ми не будемо обмежувати користувачів у використанні даного продукту, так як не у всіх користувачів є можливість використовувати ту чи іншу платформу.

Комп’ютерна платформа — це система, яка складається з апаратного пристрою та операційної системи, на якій працює програма, програма чи процес. Прикладом комп’ютерної платформи є настільний комп’ютер, на якому встановлено Microsoft Windows. Робочий стіл — це апаратний пристрій, а Windows — операційна система.

Операційна система діє як інтерфейс між комп'ютером і користувачем, а також між комп'ютером і програмою. Тому для того, щоб мати функціональний пристрій, вам потрібне обладнання та операційна система разом, щоб створити придатну комп’ютерну платформу для роботи програми.

Апаратна частина комп’ютерної платформи складається з процесора, пам’яті та сховища. Процесор трохи схожий на ваш мозок, а пам’ять — як блокнот, яким ваш мозок може користуватися під час вирішення проблеми.

Раніше люди називали різні комп’ютерні платформи за їхнім фізичним розміром, від найменших до найбільших – мікрокомп’ютери (найменші), мінікомп’ютери (середнього розміру) та мейнфрейми (найбільші). Термін

мікрокомп'ютер дещо втратив популярність - тепер більшість людей просто називають ці машини комп'ютерами або персональними комп'ютерами.

При розробці програмного забезпечення платформа може бути використана як засіб забезпечення постійної продуктивності продукту під час роботи платформи. Це може призвести до створення програмної програми, яка може виконуватися незалежно від операційної системи, встановленої на апаратному забезпеченні. Прикладами таких типів програмних додатків є пакети на основі Java та QuickTime. Деякі компанії, що розробляють програмне забезпечення в Інтернеті, використовують ці платформи, щоб дозволити онлайн-іграм без обмежень встановленої операційної системи, що впливають на роботу програмного забезпечення. Використовуючи обчислювальну платформу для емуляції старої операційної системи, ці служби дозволяють грати в ігри на сучасному обладнанні, яке зазвичай було б несумісним із програмним забезпеченням.

Деякі будинки програмного забезпечення використовували вбудовані програмні платформи, щоб дозволити їхньому програмному забезпеченню бути сумісним з двома операційними системами з одного носія. Прикладом цього може бути програмна програма або гра, яку користувач може встановити як на персональний комп’ютер (ПК) під керуванням операційної системи Microsoft Windows, так і на комп’ютер Apple Macintosh. Це називається кросплатформним додатком.

Рівень сумісності, який демонструє кросплатформна програма, досягається за допомогою мови програмування, яка також виступає в якості платформи. Використовуючи цю вбудовану обчислювальну платформу, програмні додатки можна запрограмувати на запуск на обладнанні без робочої операційної системи. Це використовується для надання графічного інтерфейсу користувача під час встановлення операційних систем на порожній диск. Як приклад незалежності від платформи мова програмування Java виступає і як мова програмування, і як обчислювальна платформа. Програми, запрограмовані

цією мовою, будуть успішно запускатися на будь-якій операційній системі або обладнанні.

Типи обчислювальних платформ, засновані на програмному забезпеченні, є поширеними і включають комп'ютерні та мобільні операційні системи, такі як системи на базі Linux і Unix, Google Chrome, Android і Palm OS. Апаратні платформи включають великі мейнфрейми та суперкомп'ютери аж до домашніх ігрових консолей. Апаратні форми обчислювальної платформи також включають удосконалену комп'ютерну машину зі скороченим набором інструкцій (Advanced RISC Machine або ARM) для архітектур мобільних систем, систем Unix та систем на базі Intel x86.

3.2 Вибір системного ПЗ

В якості веб сервера був обраний nginx. nginx - це HTTP-сервер і зворотний проксі-сервер, поштовий проксі-сервер, а також TCP/UDP проксі-сервер загального призначення.

Основною функціональностью цього серверу являється: обслуговування статичних запитів, розподіл навантаження і відмовостійкість, модульність, фільтри, в тому числі стиснення (gzip), підтримка протоколу HTTP/2.Крім цієї функціональності nginx має багато додаткової, а саме: віртуальні сервери, обмеження доступу, стрімінг, геолокація по IP-адресу, віддзеркалення запитів, підтримка SSL, проксінг TCP і UDP.

В якості порівняння ми візьмемо веб сервер Apache.

Таблиця 3.1 Особливості веб серверів Apache та Nginx.

Особливість	**Apache**	**Nginx**
1	2	3
Простота	Легкий тому що використовується модель	Складний, оскільки він використовує модель "кілька

	"одне з'єднання на процес".	з'єднань на процес".

Продовження табл. 3.1

1	2	3
Продуктивність (статика)	Повільно	В 2,5 рази швидше.
Продуктивність (динаміка)	Відмінно.	Відмінно.
Підтримка	Всі ОС.	Всі ОС, в Windows менш стабільна.

Nginx виконує важку роботу, пов'язану з HTTP - обслуговує статичні файли, кеширує вміст - так що сервер Apache може виконувати код програми в безпечному та захищеному середовищі.

В якості системи управління базами даних для режиму «продакшен» було обрано MySQL.

MySQL - найпопулярніша база даних з відкритим кодом у світі. Завдяки своїй перевіреній продуктивності, надійності та простоті у використанні MySQL став провідним вибором бази даних для веб-додатків, які використовуються Facebook, Twitter, YouTube, Yahoo! та багато інших.

Чому варто використовувати MySQL:

- масштабованість та гнучкість;
- висока продуктивність;
- висока доступність;
- надійна підтримка транзакцій;
- надійний захист даних;
- комплексна розробка додатків;
- легкість управління.

Тепер давайте розглянемо різницю між PostgreSQL и MySQL.

Таблиця 3.2 Особливості PostgreSQL та MySQL.

Назва	**MySQL**	**PostgreSQL**
Основа	Система управління реляційними базами даних.	Об'єктно-реляційна система управління базами даних.
Розширюваність	Не розширюється.	Розширюється.
Резервне копіювання	Mysqldump і XtraBackup забезпечує резервне копіювання в MySQL.	Забезпечує онлайн резервне копіювання.
Об'єкт домену даних	Не забезпечує об'єкт домену даних.	Надає об'єкт домену даних.

Різниця кількості запитів MySQL и PostgreSQL в пошуковій системі Google. Синій колір це MySQL, червоний – PostgreSQL.

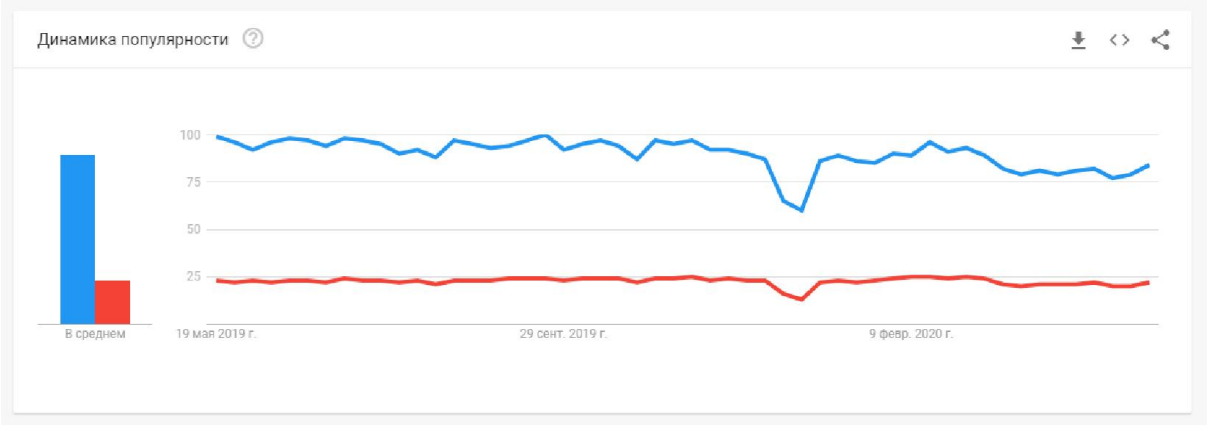


Рисунок 3.1 – Різниця кількості запитів MySQL и PostgreSQL.

Виходячи із інформації на рисунку ми можемо побачити, що система управління базами даних MySQL популярніша чим PostgreSQL.

3.3 Обгрунтування вибору фреймворків, бібліотек, засобів розробки

Так як проєкт повинен бути реалізований на web технологіях, то в якості фронт частини був обранний фреймворк Nuxt, а на серверній частині фреймворк Laravel.

Vue - прогресивна основа для побудови інтерфейсів користувача. Він розроблений з нуля, щоб бути поступовим для прийняття, і може легко змінювати масштаб між бібліотекою та рамкою залежно від різних випадків використання. Він складається з доступної основної бібліотеки, яка зосереджена лише на рівні перегляду, та екосистеми бібліотек, що допомагають вирішити складність у великих додатках на одній сторінці.

Функції Vue.js:

- реактивні інтерфейси;
- декларативний рендерінг;
- зв'язування даних;
- директиви (всі директиви містять префікс «v-»).
- логіка шаблонов;
- компоненти;
- обробка подій;
- властивості;
- переходи та анімація CSS;
- фільтри.

Основними концепціями даного фреймворку являються: конструктор, компоненти, директиви та переходи.

Тепер порівняння фреймворку Vue з іншими популярними фреймворками.

Таблиця 3.3 Порівняння Vue, Angular и React фреймворків.

Назва	**Angular**	**React**	**Vue**

1	2	3	4
Рівень складності	Високий	Середній	Низький

Продовження табл. 3.3

1	2	3	4
Розмір	500 КВ	100 КВ	80 КВ
Час завантаження	Довго	Не довго	Швидко
Результати на Google Trend	~50	~90	~45
Результати на Stackoverflow Survey	~74	~203	~43
Результати на Stackoverflow Job	~12	~35	~3
Результати на Upwork	~728	~816	~257

Nuxt.js - це безкоштовне середовище веб-додатків з відкритим вихідним кодом, засноване на Vue.js, Node.js, Webpack та Babel.js.

Мета Nuxt — зробити веб-розробку інтуїтивно зрозумілою та продуктивною, маючи на увазі чудовий досвід розробника.

Щоб зрозуміти, що таке Nuxt, потрібно зрозуміти, що потрібно для створення сучасного додатка:

- Vue.js як фреймворк JavaScript, що забезпечує реактивність і веб-компоненти;
- пакет для підтримки гарячої заміни модулів під час розробки та комплектування коду для виробництва, підтримка як Webpack 5, так і Vite;

- транспілер для написання найновішого синтаксису JavaScript і підтримки застарілих браузерів, використовується для цього esbuild;
- сервер для обслуговування програми в процесі розробки, а також для підтримки рендерингу на стороні сервера або маршрутів API, Nuxt використовує h3 для універсальності розгортання, наприклад безсерверного, робочого, Node.js і неперевершеної продуктивності;
- бібліотеку маршрутизації для обробки навігації на стороні клієнта в якості vue-router.

Nuxt подбає про все це, щоб можливо було зосередитися на важливому: створенні веб-додатка.

На додаток до цього налаштування Nuxt надає структуру каталогів, зосереджену на певних функціях, щоб зосередитися на створенні, а не налаштуванні.

TypeScript — це наднабір JavaScript, що означає, що він містить усі функції JavaScript, а потім деякі. Таким чином, будь-яка програма, написана на допустимому JavaScript, також працюватиме, як очікується, у TypeScript. Фактично, TypeScript компілюється просто у звичайний JavaScript. Отже, яка різниця? TypeScript пропонує нам більше контролю над нашим кодом за допомогою анотацій типів, інтерфейсів і класів.

TypeScript був створений Microsoft і був випущений у 2012 році після двох років розробки. Він був створений, щоб дозволити додаткову статичну перевірку типів, що було б особливо корисно при розробці великомасштабних програм. Цікаво, що одна з причин, чому Microsoft розробила TypeScript, полягала в тому, що їхні внутрішні команди мали проблеми з масштабуванням JavaScript для власних проектів Microsoft, зокрема команда, яка працювала над Bing Maps.

На додаток до відображення помилок під час компіляції, певні IDE, такі як Visual Studio Code або PHPStorm, відображатимуть помилки розробнику під

час введення коду. Це робить виправлення простих необережних помилок набагато швидшим і легшим.

TypeScript також може виводити типи, які явно не оголошені розробником, наприклад, визначення типу значення функції. Якщо функція додає два параметри, які були оголошені як числові, TypeScript визначить, що значення має бути типу number.

Хто використовує TypeScript? Команди AngularJS, Ionic, NativeScript, Aurelia, SitePen і Epic використовують TypeScript, щоб зробити свій код більш керованим, покращити робочий процес розробки та допомогти в процесі налагодження.

Laravel - це фреймворк для web-додатків з виразним і елегантним синтаксисом. Він дозволить спростити вирішення основних завдань, таких як аутентифікація, маршрутизація, сесії і кешування. Laravel - це спроба об'єднати все найкраще, що є в інших PHP фреймворках.

Наведені нижче функції служать ключовими особливостями фреймворку Laravel:

Пакети забезпечують модульну систему упаковки;

- Eloquent ORM*;*
- будівник запитів;
- логіка програми;
- зворотна маршрутизація;
- контейнери IoC дозволяють генерувати нові об'єкти за принципом інверсії управління (IoC);
- юніт-тести;
- колекції;
- інтерфейс командного рядка (CLI), який називається Artisan.

MVC (Model-View-Controller) — це архітектурний шаблон, який використовується для розбиття програми на три основні частини: дані (модель),

інтерфейс для перегляду та зміни даних (View) та операції, які можна виконувати з даними (контролер).

Подумайте про це як про замовлення піци. Ви телефонуєте із запитом користувача на піцу Pepperoni. Щойно ваш запит зареєстровано, людина, яка готує піцу (Контролер), розкладає її на ряд кроків: бере тісто, розпалює духовку, посипає тертим сиром. Контролер може використовувати лише обмежені ресурси, які він має у своєму розпорядженні, цим обмеженим набором інструментів є модель: руки, духовка, піднос для піци тощо. Нарешті, ви отримуєте піцу, яка є чудовим виглядом.

Такий спосіб структурування програми корисний, оскільки він розділяє речі на логічні області. Це робить ваш код більш організованим, менш крихким і його легше налагоджувати. Laravel реалізує архітектуру MVC як частину свого дизайну.

На відміну від CMS, таких як Drupal або Wordpress, Laravel дає вам повний контроль над вашим додатком. У Laravel все робиться в коді, на відміну від Drupal або Joomla, наприклад, де ви можете створювати функціональні веб-сайти, не писуючи жодного рядка коду або навіть не знаючи, що таке PHP.

Простіше кажучи, CMS – це програма, яка постачається з основними функціональними можливостями і побудована на основі фреймворка. Laravel — це фреймворк, який використовується для створення додатків, включаючи платформи CMS.

Laravel і Symfony – основна інформація про обидві фреймворки

Laravel — це фреймворк з відкритим вихідним кодом, який дотримується шаблону проектування модель-погляд-контролер. Він повторно використовує існуючі компоненти різних фреймворків для створення веб-додатків. Він також складається з основних функцій PHP-фреймворків, таких як Yii, CodeIgniter або Ruby on Rails. Якщо ви добре знаєте Core PHP і Advanced PHP, Laravel стане для вас набагато легше. Він добре відомий тим, що просте кодування наближається та скорочує рамки часу розробки, що чудово підходить для розробки простого додатка PHP.

Symfony – цей фреймворк також базується на проектах PHP з відкритим кодом, таких як Propel, Doctrine, PHPUnit, Twig і Swift Mailer. Незважаючи на те, що він має такі компоненти, як Symfony YAML, Symfony Event Dispatcher, Symfony Dependency Injector і Symfony Templating. З 2005 року Symfony став все більш надійним і зрілим фреймворком. В основному використовується для складних корпоративних проектів.

Перший є найбільш очевидним – вони обидва використовують PHP як мову програмування.

Вони є кросплатформними, що означає, що вони є комп'ютерним програмним забезпеченням, яке реалізується на кількох обчислювальних платформах.

Варто зазначити, що обидва є багатокористувацькими та багатомовними. Обидва забезпечують структуру програми, шаблон для інтерфейсів і підтримують текстовий пошук.

Незважаючи на деяку схожість між цими двома структурами, ми, безперечно, можемо вказати і на відмінності.

Symfony можна назвати звичайною мовою PHP – її можна змінити на C# або Java, але, звичайно, вона складається з унікальних і єдиних у своєму роді елементів, які роблять його видатним. У 2021 році Laravel з'явився як найпопулярніший PHP фреймворк. Він більше покладається на магічні методи та риси. Це робить код коротшим, а всю структуру легшою для розуміння. Symfony розроблено для більш масштабних або складніших проектів, які містять величезні можливості та використовуються значною кількістю клієнтів. У той же час Laravel пов'язаний із шаблоном проектування MVC, про який згадувалося вище. Коли мова йде про масштабованість, якщо ви виберете Laravel, ви повинні знати про необхідність написання коду для обробки цього. Symfony надає кілька платформ для підтримки масштабованості – це отримує бал.

Механізм шаблонів — Symfony надає Twig, але Laravel надає Blade, що має велику перевагу — ви можете повторно використовувати код. Ця опція не існує в Twig.

Якщо вас цікавить підтримка бази даних – обидва надають об’єктно-реляційне відображення для доступу до даних. Symfony використовує Doctrine, Laravel – Eloquent.

І останнє, але не менш важливе – швидкість. У Laravel швидкість програми подібна до іншої програми PHP. Він забезпечує належну систему контролю версій, яка допомагає пізніше переносити програми. Якщо Symfony реалізовано належним чином, швидкість програми покращується. Він налаштовує швидкість окремих основних функцій, тому вся програма може легко вирішити, які функції потрібні на даний момент.

Зобразимо таблицю з різницею між Laravelта Symfony.

Таблиця 3.4. Порівняння Laravelта Symfony.

Характеристика	Laravel	Symfony
1	2	3
Модульністьі масштабування	Використовує програми на основі MVC з низкою попередньо створених залежностей. Це робить його трохи менш гнучким, але більш зручним, якщо ви використовуєте програми MVC.	Використовує різноманітні компоненти, що забезпечує більш надійну модульність. Код організований краще.
Механізм шаблонів	Механізмом шаблонів за замовчуванням є Blade, що дозволяє	Використовує Twig як механізм шаблонів за замовчуванням.

	повторно використовувати код, чого ви не отримуєте з Twig.	

Продовження табл. 3.4

1	2	3
Підтримка бази даних	Використовує об'єктно-реляційне відображення (ORM) для доступу до даних через Eloquent. Laravel з коробки підтримує такі бази даних: MySQL, PostgreSQL, SQLite та SQLServer.	Використовує об'єктно-реляційне відображення (ORM) для доступу до даних через Doctrine. Symfony з коробки підтримує такі бази даних: Drizzle, MySQL, Oracle, PostgreSQL, SAP Sybase SQL Anywhere, SQLite, SQLServer.
Міграції баз даних	Міграції бази даних здійснюються вручну, але не вимагають визначення полів.	Переміщення даних відбувається автоматично, вимагаючи лише простих визначень для полів у моделі.
Моделі	Потрібні значні знання SQL. Eloquent також зазвичай пов'язує вашу	Не вимагає значних знань SQL, хоча ви повинні створювати

	програму з дизайном схеми БД, роблячи її набагато менш гнучким у цьому відношенні.	репозиторій для кожного виклику.

3.4 Розробка дизайну користувацького інтерфейсу

В результаті розробки програмних модулів було використано вільний підхід та CSS фреймворк (Рисунок 3.2 – 3.16).

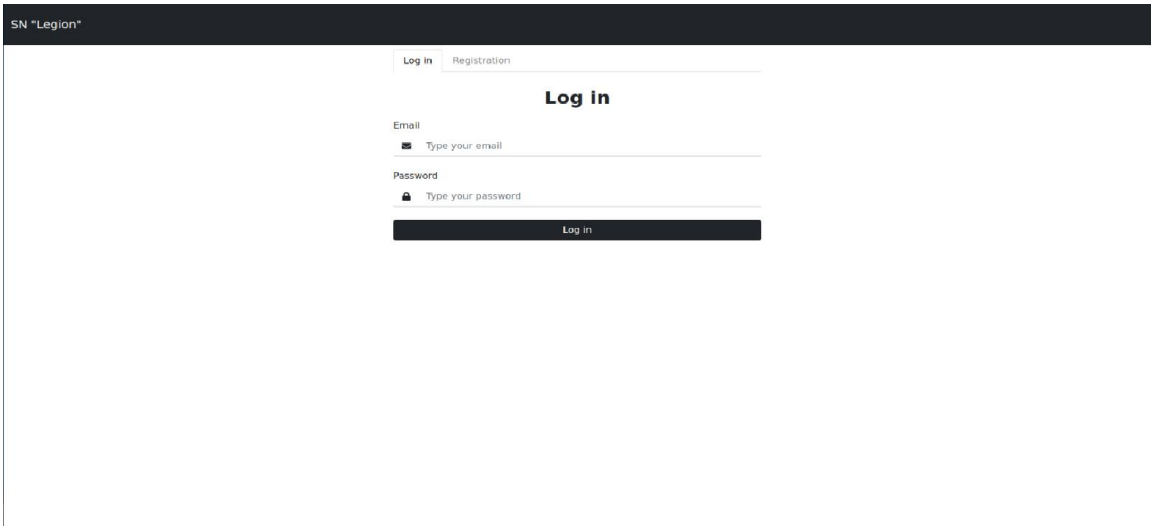


Рисунок 3.2 – Сторінка входу в систему.

На цій сторінці користувачі можуть ввійти в систему за допомогою email та паролю котрі були вказані ним при попередній реєстранції.

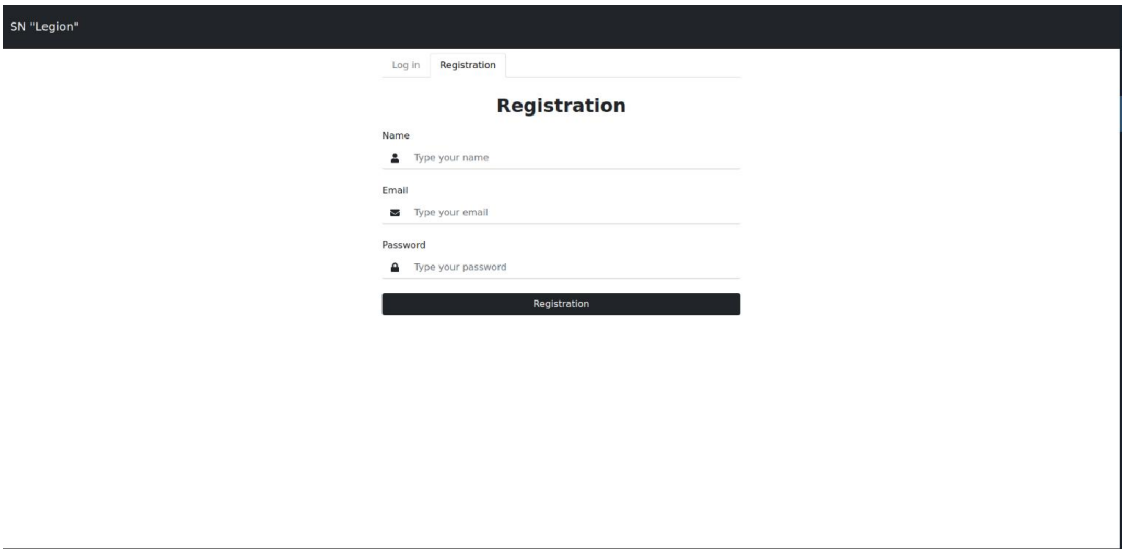


Рисунок 3.3 – Сторінка реєстрації в системі.

На цій сторінці користувачі зареєструвати свій аккаунт в сервісі.

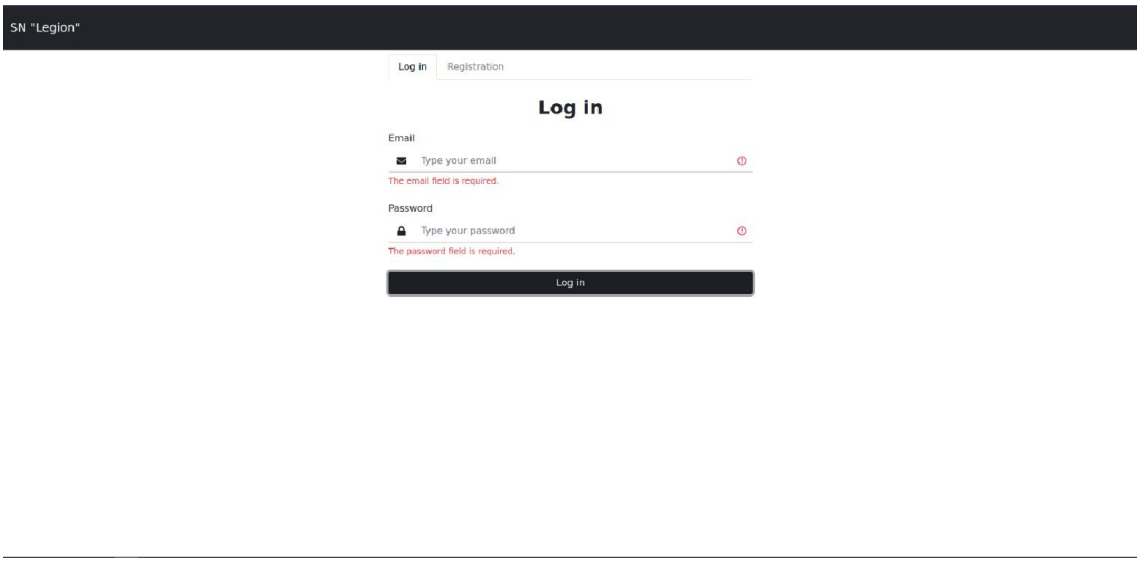


Рисунок 3.4 – Сторінка входу в систему із відображенням помилок.

На даному рисунку було зображено ефетивність роботи системи валідації для форми входу в систему.

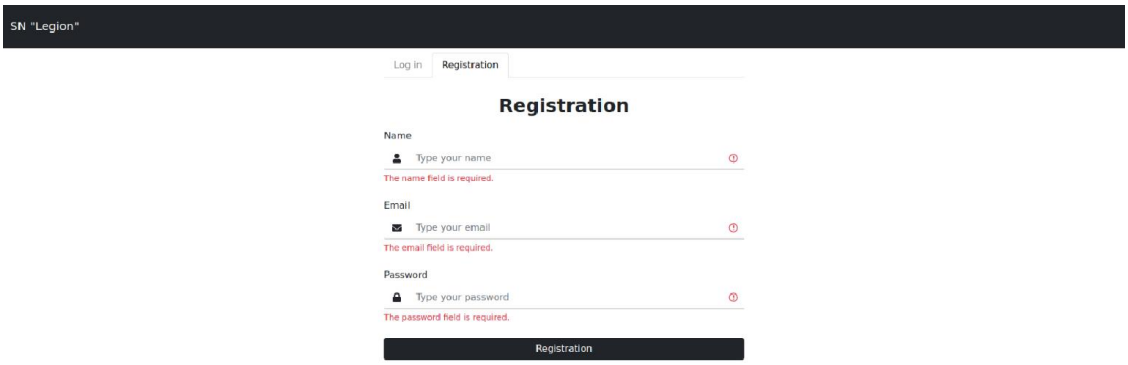


Рисунок 3.5 – Сторінка реєстрації в системі із відображенням помилок.

На даному рисунку було зображено ефективність роботи системи валідації для форми реєстрації в системі.

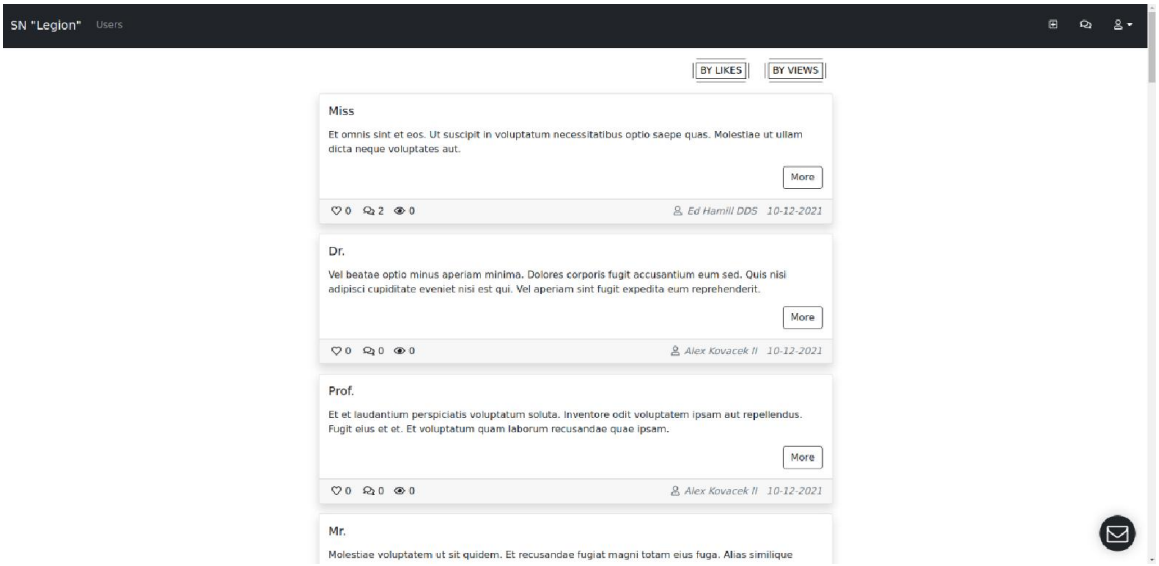


Рисунок 3.6 –Сторінка публікацій.

На даній сторінці користувач може переглядати, лайкати, коментувати публікації друзів.

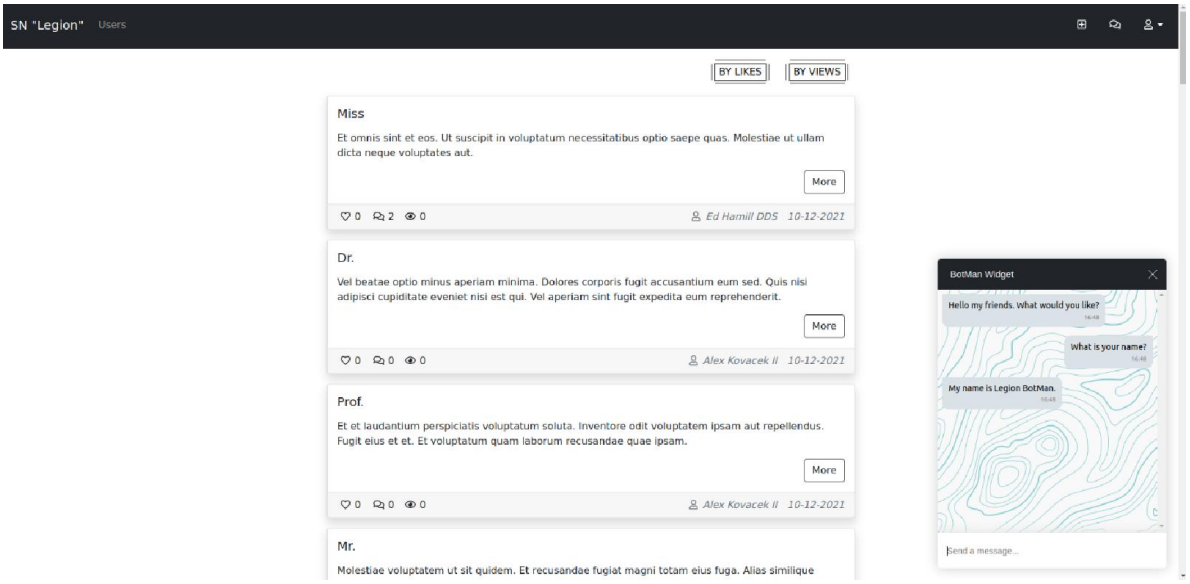


Рисунок 3.7 – Сторінка публікацій та відображення чат боту.

Також на даній сторінці користувач може використовувати чат бота для отримання певної інформації.

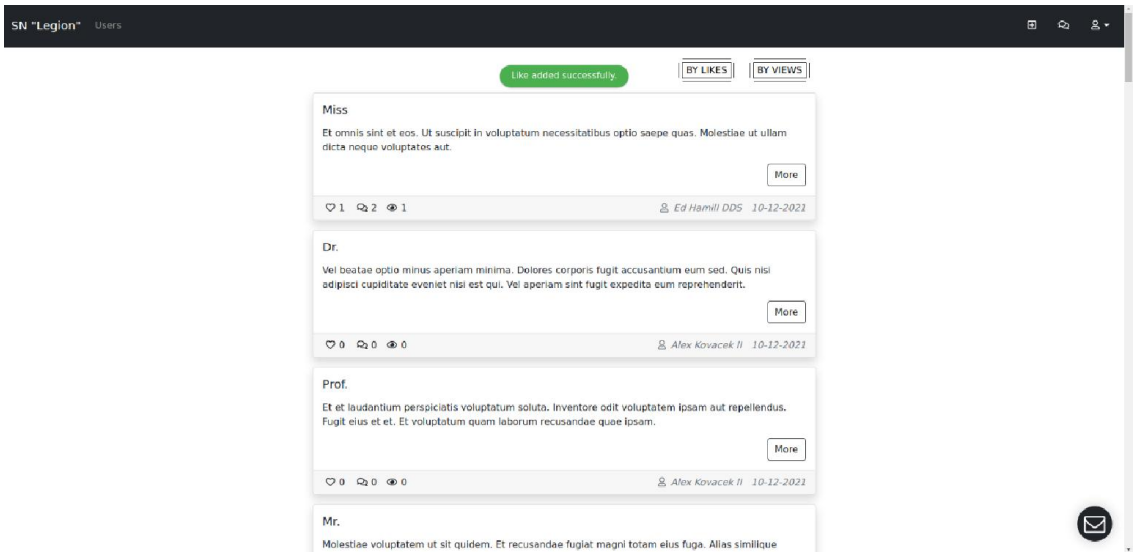


Рисунок 3.8 – Сторінка публікацій та відображення ефективності функціоналу лайків.

На даному рисунку зображено результат лайку публікації.

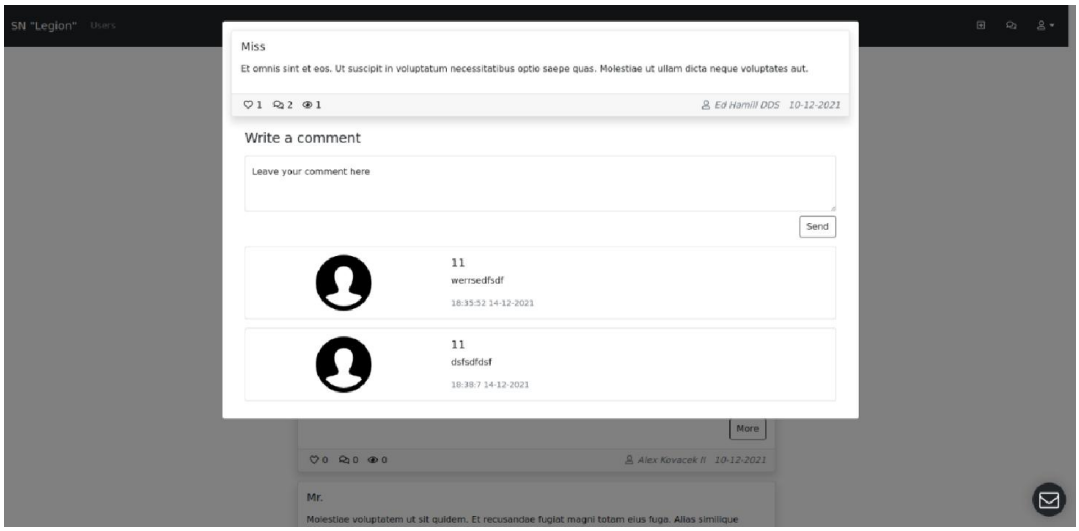


Рисунок 3.9 – Сторінка публікацій та відображення модального вікна публікації.

На даному рисунку зображено повний опис публікації в модальному вікні та коментарі.

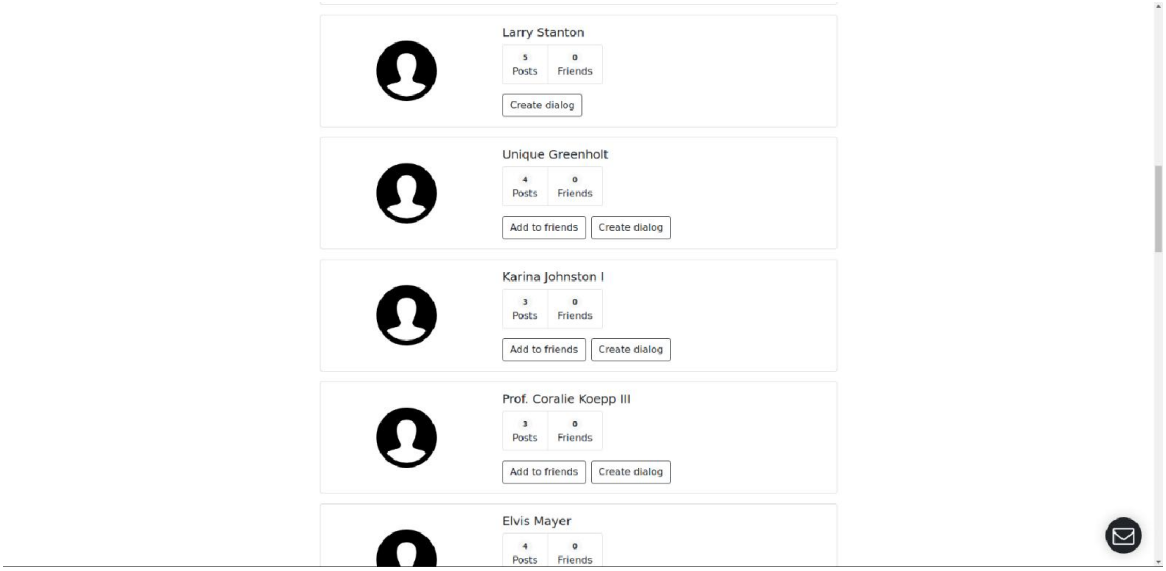


Рисунок 3.10 – Сторінка користувачів системи.

На даній сторінці зображено список користувачів системи. Користувач може створити діалог, додати в друзі та перейти до сторінки користувача.

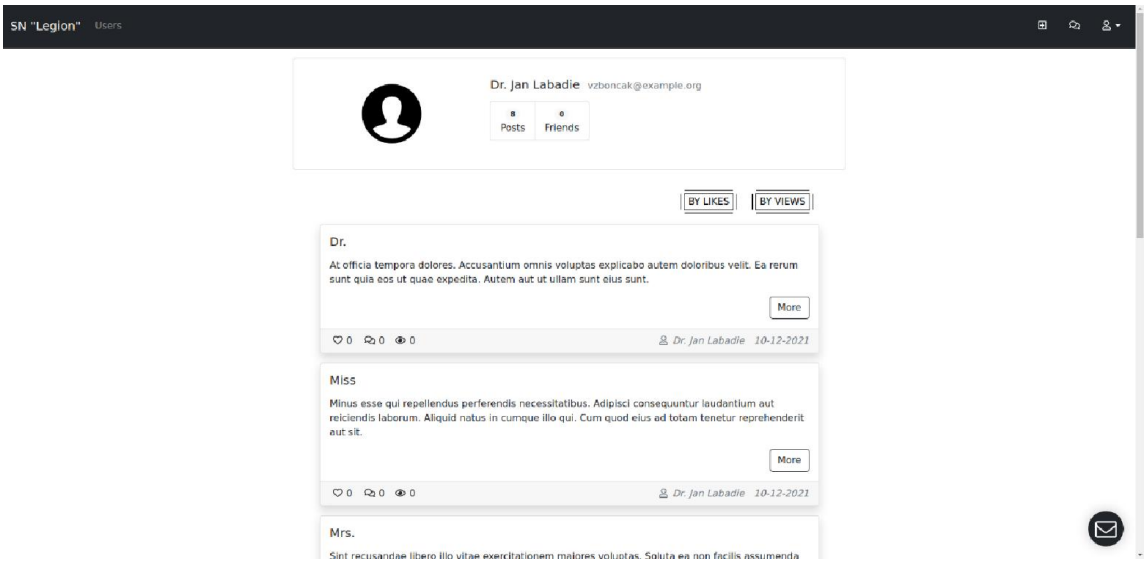


Рисунок 3.11 – Сторінка профілю користувача.

На даній сторінці зображено профіль користувача системи. Користувач може створити діалог, додати в друзі, переглядати публікації іншого користувача.

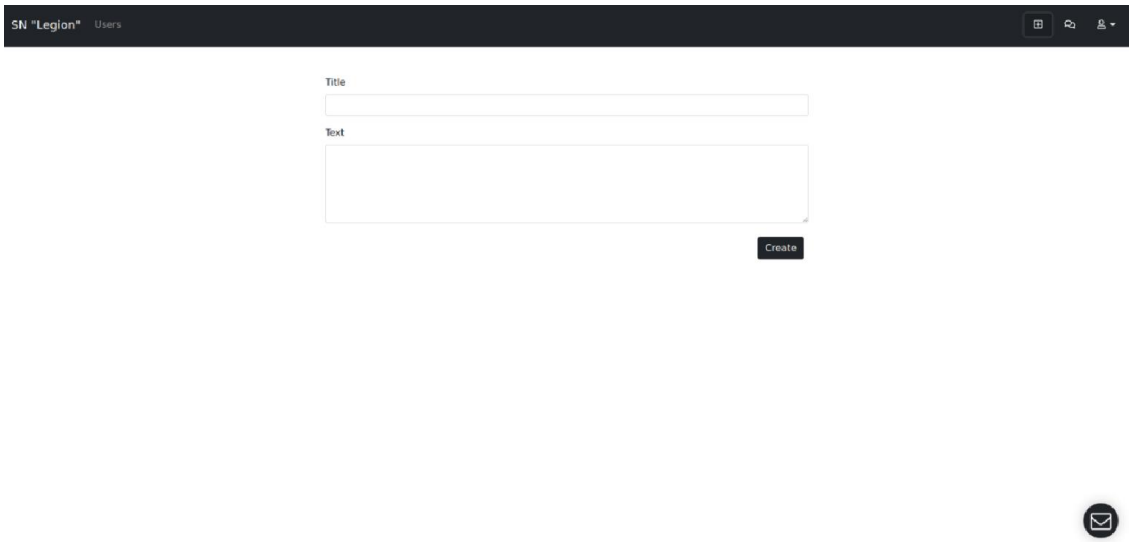


Рисунок 3.12 – Сторінка створення публікації.

На даній сторінці зображено форму створення публікації.

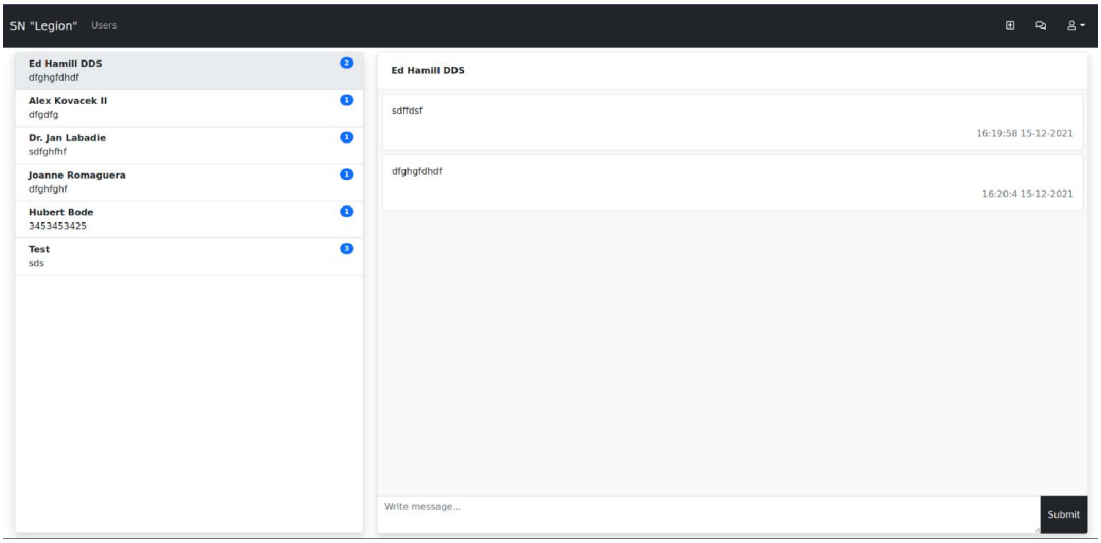


Рисунок 3.13 – Сторінка чату.

На даній сторінці користувач може переглядати та відправляти повідомлення іншим користувачам.

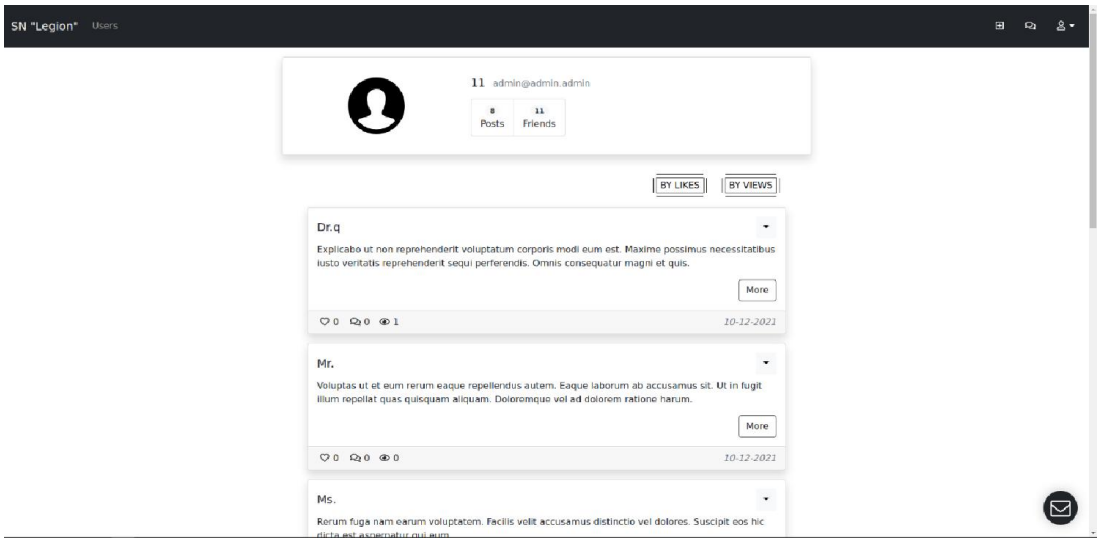


Рисунок 3.14 – Сторінка особистого профілю.

На даній сторінці користувач може переглядати дані свого аккаунту, тіж самі публікації і т.д.

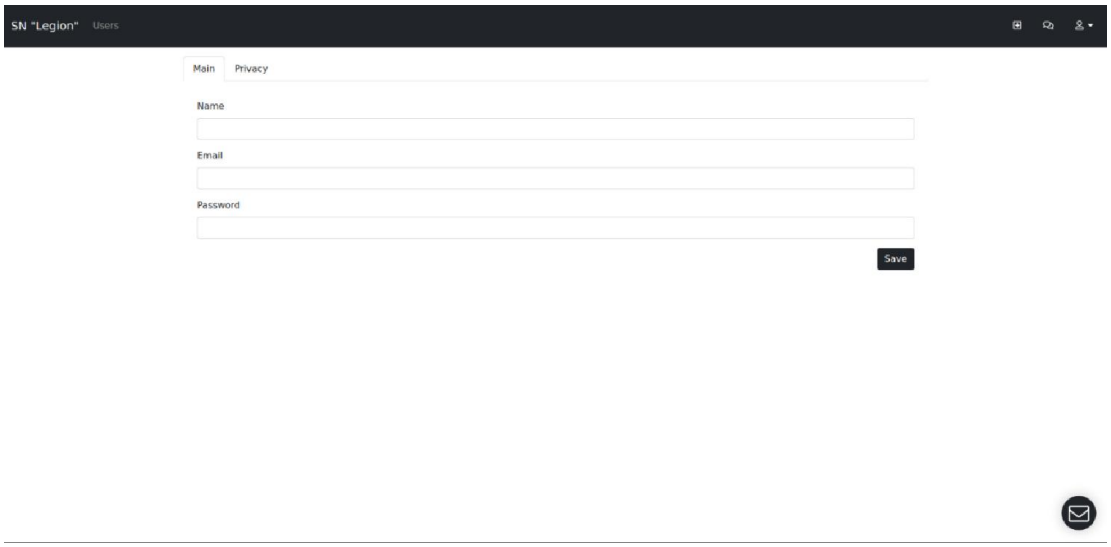


Рисунок 3.15 – Сторінка головних налаштувань.

Сторінка на котрій користувач може змінити ім’я, пошту та пароль до акаунту.

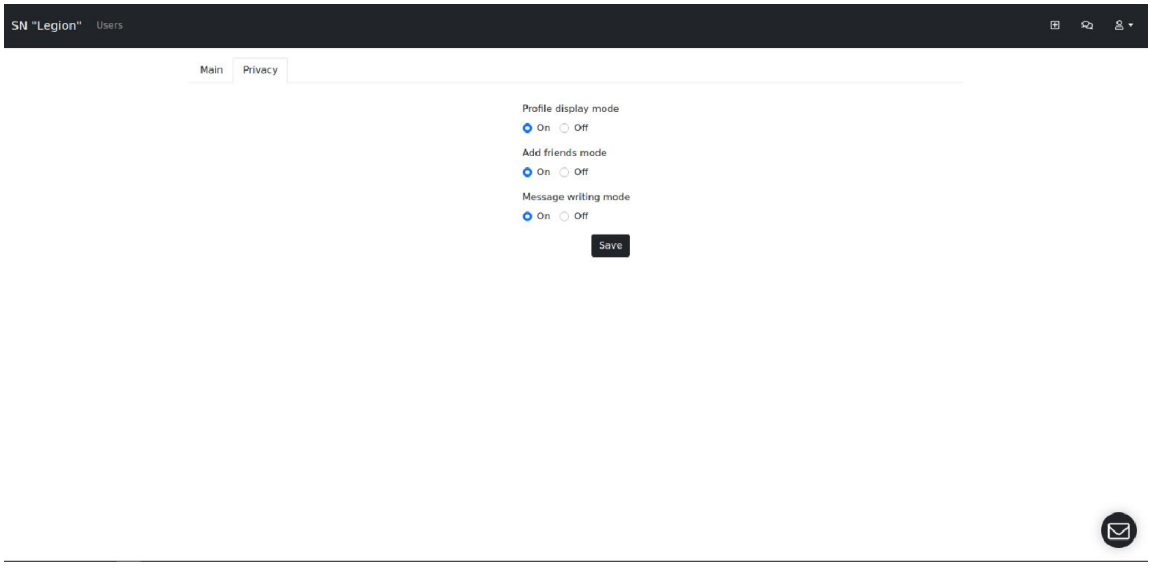


Рисунок 3.16 – Сторінка допоміжних налаштувань.

Це друга чатина сторінки на котрій користувач може змінити параметри відображення свого акканту для других користувачів системи.

РОЗДІЛ 4

ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЮ

4.1 Вибір виду тестування

Тестування – заключний етап розробки модулів, що відіграє важливу роль у процесі створення якісного програмного забезпечення.

Веб-тестування - це тестування веб-сайтів чи веб-додатків на потенційні помилки. Веб-систему потрібно повністю перевірити від кінця до кінця, перш ніж вона перейде до кінцевого користувача. Виконуючи тестування веб-сайтів, можна переконатися, що веб-система працює належним чином і може бути прийнята користувачами в режимі реального часу.

Чеклист веб-тестування:

- тестування функціональності;
- тестування на корисність;
- тестування інтерфейсу;
- тестування на сумісність;
- тестування працездатності;
- тестування безпеки.

Перехресне тестування браузера дає змогу зробити кожен веб-переглядач ідеальним у будь-якому веб-переглядачі чи мобільному пристрої за допомогою хмарної лабораторії реальних пристроїв.

LambdaTest — це масштабована хмарна платформа для міжбраузерного тестування, розроблена для того, щоб запропонувати тестування всіх веб-сайтів і веб-програм, необхідних для хмарної інфраструктури.

Платформа LambdaTest допомагає забезпечити безперебійне відображення елементів вашого веб-додатка (наприклад, JavaScript, CSS, HTLM5, відео тощо) у кожному веб-переглядачі для настільних комп'ютерів і мобільних пристроїв з підтримкою ручного, візуального та автоматичного тестування. За допомогою LambdaTest ви можете отримати доступ до понад 2000 комбінацій настільних і мобільних браузерів у хмарі.

Test for – усі посилання на веб-сторінках, підключення до бази даних, форми, які використовуються для подання або отримання інформації від користувача на веб-сторінках, тестування файлів cookie тощо.

Перегляньте всі посилання:

Перевірте вихідні посилання з усіх сторінок на певний домен, який тестується.

Перевірте всі внутрішні посилання.

Тестування посилань, які стрибають на одній сторінці.

Тестові посилання використовуються для надсилання електронних листів адміністратору або іншим користувачам із веб-сторінок.

Перевірте, чи є сторінки-сироти.

Нарешті, перевірка посилань включає в себе перевірку непрацюючих посилань у всіх вищезгаданих посиланнях.

Тестові форми на всіх сторінках:

Форми є невід'ємною частиною будь-якого веб-сайту. Форми використовуються для отримання інформації від користувачів і для взаємодії з ними. Отже, що потрібно перевірити в цих формах?

- спочатку перевірте всі підтвердження кожного поля;
- перевірте значення за замовчуванням у полях;
- неправильні введення у формах до полів у формах;

- параметри створення форм, якщо такі є, видалення форми, перегляд або змінення форм.

Тестування файлів cookie:

Файли cookie – це невеликі файли, які зберігаються на машині користувача. Вони в основному використовуються для підтримки сеансу – в основному, сеансів входу. Перевірте програму, увімкнувши або вимкнувши файли cookie в параметрах вашого браузера.

Перевірте, чи зашифровані файли cookie, перш ніж записувати їх на машину користувача. Якщо ви тестуєте файли cookie сеансу (тобто файли cookie, термін дії яких закінчується після закінчення сеансу), перевірте сеанси входу та статистику користувачів після закінчення сеансу. Перевірте вплив на безпеку програми, видаливши файли cookie. (Незабаром я також напишу окрему статтю про тестування cookies)

Перевірте HTML/CSS:

Якщо оптимізуєте свій для пошукових систем, то перевірка HTML/CSS є найважливішою. В основному перевіряйте сайт на наявність синтаксичних помилок HTML. Перевірте, чи можна сканувати сайт різними пошуковими системами.

Тестування бази даних:

Узгодженість даних також дуже важлива у веб-додатку. Перевіряйте цілісність даних і помилки під час редагування, видалення, зміни форм або виконання будь-яких функцій, пов’язаних з БД.

Перевірте, чи всі запити до бази даних виконуються правильно, дані витягуються та оновлюються правильно. Більше про тестування бази даних може бути навантаженням на БД, ми розглянемо це у веб-завантаженні або тестуванні продуктивності нижче.

Для тестування розроблених модулів скористаємося ручним тестуванням.

4.2 Тест план

План тестування – це детальний документ, який описує стратегію тестування, цілі, графік, оцінку, результати та ресурси, необхідні для проведення тестування програмного продукту. План тестування допомагає визначити зусилля, необхідні для перевірки якості програми, що тестується. План тестування служить планом для проведення заходів з тестування програмного забезпечення як визначеного процесу, який щохвилино відстежується та контролюється менеджером тестування.

Відповідно до визначення ISTQB: «План тестування — це документ, що описує обсяг, підхід, ресурси та графік запланованих тестових заходів».

Створення документа плану тестування має ряд переваг:

- допомога людям за межами тестової групи, таким як розробники, бізнес-менеджери, клієнти, зрозуміти деталі тестування;
- план тестування керує мисленням. Це як збірник правил, якого потрібно дотримуватися;
- важливі аспекти, такі як оцінка тесту, обсяг тестування, стратегія тестування, задокументовані в Плані тестування, тому його можна переглянути командою керівництва та повторно використати для інших проектів.Для того, щоб провести тестування, потрібно скласти тест-плани та визначити функції, що будуть протестовані.

Ви вже знаєте, що складання плану тестування є найважливішим завданням процесу управління тестуванням. Описано сім кроків нижче, щоб створити план тестування відповідно до IEEE 829:

- проаналізуйте продукт;
- розробити стратегію тестування;
- визначте цілі тесту;
- визначте критерії тестування;
- планування ресурсів;
- план тестового середовища;
- розклад і оцінка;

- визначте результати тесту.

Головні функції для котрих потрібно провести тестування:

- вхід в систему за допомогою даних;
- реєстрація в системі за допомогою даних;
- редагування даних користувача системи;
- створення/редагування/видалення публікацій;
- додавання в друзі;
- діалог із чат ботом.

Розробимо тест-кейси, які необхідно перевірити. Враховуючи те, що розроблені модулі мають однаковий базовий функціонал вони будуть мати ряд однакових тест-кейсів, тож позначимо їх як загальні.

Таблиця 4.1 – Тест-кейси загальні для розроблених модулів.

Опис	Вхід в систему за допомогою даних.	
Передумови	Відкрито програмний модуль.	
№	**Дія**	**Очікуваний результат**
1	Заповнюємо форму вірними даними.	Перенаправлення на сторінку публікацій.
Опис	Реєстрація в системі за допомогою даних.	
Передумови	Відкрито таблицю з загальною кількістю записів більшою за 15.	
№	**Дія**	**Очікуваний результат**
1	Заповнюємо форму вірними даними.	Перенаправлення на сторінку профілю.
Передумови	Редагування даних користувача системи	
№	**Дія**	**Очікуваний результат**
1	Заповнюємо певний набір даних та виповняємо	Повідомлення про успішність зберігання.

	спробу зберегти.	
Передумови	Створення публікації	
№	**Дія**	**Очікуваний результат**
1	Заповнюємо певний набір даних та виповняємо спробу зберегти.	Повідомлення про успішність зберігання та перенаправлення на сторінку профілю.
Передумови	Оновлення публікації	

Продовження табл. 4.1

№	**Дія**	**Очікуваний результат**
1	Заповнюємо певний набір даних та виповняємо спробу зберегти.	Повідомлення про успішність оновлення та перенаправлення на сторінку профілю.
Передумови	Видалення публікації	
№	**Дія**	**Очікуваний результат**
1	Тиснемо на кнопку «Delete»	Повідомлення про успішність видалення.
Передумови	Додавання в друзі	
№	**Дія**	**Очікуваний результат**
1	Тиснемо на кнопку «Addfriend»	Повідомлення про успішність операції.
Передумови	Діалог із чат ботом	
№	**Дія**	**Очікуваний результат**
1	Написання якогось тексту та підтверження відправки його боту.	Отримати будь який успішний результат від чат боту.

4.3. Введення в експлуатацію

Щоб забезпечити стабільність на всіх етапах збірки, ви можете використовувати інструменти автоматизації, такі як Gitlab CI і Jenkins.

На етапі збірки – код повинен бути встановлений, залежно від бібліотек PHP та активів CSS/JS, які потрібно підготувати. Нарешті, контейнер Docker з тегом версії має бути створений і вставлений в реєстр Docker. Також буде можливість виконувати модульні тести та аналізувати код.

На етапі випуску – об'єднати контейнер Docker (вироблений на етапі збірки) з конфігурацією середовища (постановка та/або виробнича), де буде запущена збірка.

На етапі запуску – запустіть програму у вибраному середовищі з належною конфігурацією середовища. Це буде залежати від стадії випуску.

Провідні практики веб-розробки рекомендують зберігати постійні дані в окремих сервісах. Це може бути будь-яка реляційна база даних, наприклад, як Redis, Amazon S3 тощо.

Кожен веб-розробник повинен знати чотирирівневу модель розгортання розробки, тестування, стадії та виробництва. У більшості місць це «стандарт» для створення, тестування та обслуговування веб-додатків і виглядає так:

- розробка: тут розробники вносять зміни в код, і зазвичай це локальне середовище з одним клієнтом (наприклад, ноутбук розробника);
- тестування: це середовище інтеграції, де розробники об'єднують зміни, щоб перевірити, чи вони працюють разом. Це також може бути середовище забезпечення якості або UAT;
- проведення: тут тестовані зміни виконуються з інфраструктурою та даними, еквівалентними для виробництва, щоб забезпечити належну роботу після випуску;

- виробництво: Це середовище живого виробництва.

Ця модель існує вже деякий час, і її часто вважають найкращою практикою для архітектур розгортання. Але це має ряд проблем...

Чотирирівнева модель виникла внаслідок певного історичного злиття зростаючої складності в дизайні, тестуванні та упаковці веб-додатків, а також фізичних обмежень обчислювальної інфраструктури. Оскільки програмне забезпечення ставало складним, розробники почали використовувати більш складні методи пакування для розгортання цього програмного забезпечення. Це дозволило нам почати розбивати модель розгортання на серію кроків, які більш точно відповідали видам тестування, які потрібні для складних додатків. Ці кроки стали нашим реальним середовищем. Ми почали переміщати код через ці рівні, при цьому кожен рівень стверджував, що пропонує певну гарантію щодо підвищення узгодженості даних і середовища, а також якості коду.

У той же час, однак, здатність керувати розгортанням того самого програмного забезпечення була обмежена вартістю та складністю придбання обчислювальних ресурсів (тобто серверів) і керування ними для обслуговування середовищ. Якщо ви хотіли, щоб нове середовище тестувало код, ви повинні були його купити, створити, підтримувати та знайти спосіб розгортання в ньому. У результаті більшість команд розробників підтримували абсолютну мінімальну кількість середовищ або серверів, необхідних для задоволення їхніх власних вимог робочого процесу. У багатьох випадках насправді це було менше чотирьох, а іноді всього двох (або одного, якщо ви робили розробку безпосередньо на робочому сервері).

Очевидно, скорочення використання середовищ дуже ускладнює визначення того, чи ви тестуєте та розгортаєте код безпечно та надійно, але навіть із чотирма середовищами виникнуть проблеми:

Об’єднання коду має здійснюватися на рівні розробки, що призводить до конфліктів через відсутність видимості.

Зміни не можна легко перевірити ізольовано, що ускладнює відстеження та перевірку цих змін (це проблема, коли у вас більше змін, ніж середовищ інтеграції, тому зазвичай починається з двох змін!)

Якщо ті, хто займається QA, не володіють технічною підготовкою, зміни повинні проходити тестування в спільному середовищі. Це може призвести до того, що ці середовища дуже легко заблокуються та створить проблеми з переробкою.

Порушені тестові середовища або зміни вимагають тестування всіх змін, а не лише тих, які потребують доопрацювання.

Неспроможність підтримувати середовище в актуальному стані за допомогою Production призводить до застарілих даних тестування, неправильних залежностей операційної системи та інших факторів середовища, що може призвести до збою розгортання Production, що вимагає дорогого та неприємного відкату.

Контроль версій дає нам можливість ізолювати зміни, однак обмежене середовище для тестування зводить нанівець цю перевагу і змушує нас завчасно об’єднувати зміни, що спричиняє конфлікти та вузькі місця.

Зрештою, основна проблема полягає в тому, що коли у вас є обмежена кількість серверів для розгортання, ймовірність того, що будь-який із них буде заблоковано зламаним кодом, значно зростає.

Для того щоб користуватися даною системою, потрібно мати 1 або 2 веб сервера та 2 домена.

Системні вимоги:

- операційна система: Windows або Unix системи;
- процесор: 1ГГц або вище;
- рперативна пам’ять: 256 МБ;
- жорсткий диск: 60 МБ вільного місця;
- база данних: MySQL 5.7 або вище;

- PHP: версії 8.0 або вище;
- NPM: версії 16.14.5 або вище;
- Composer: версії 1.9.3 або вище.

Процес деплою проєкту (Frontend):

- встановити js залежності за допомогою npm;
- налаштувати файл конфігурації проєкту;
- компіляція всіх js та css залежностей.

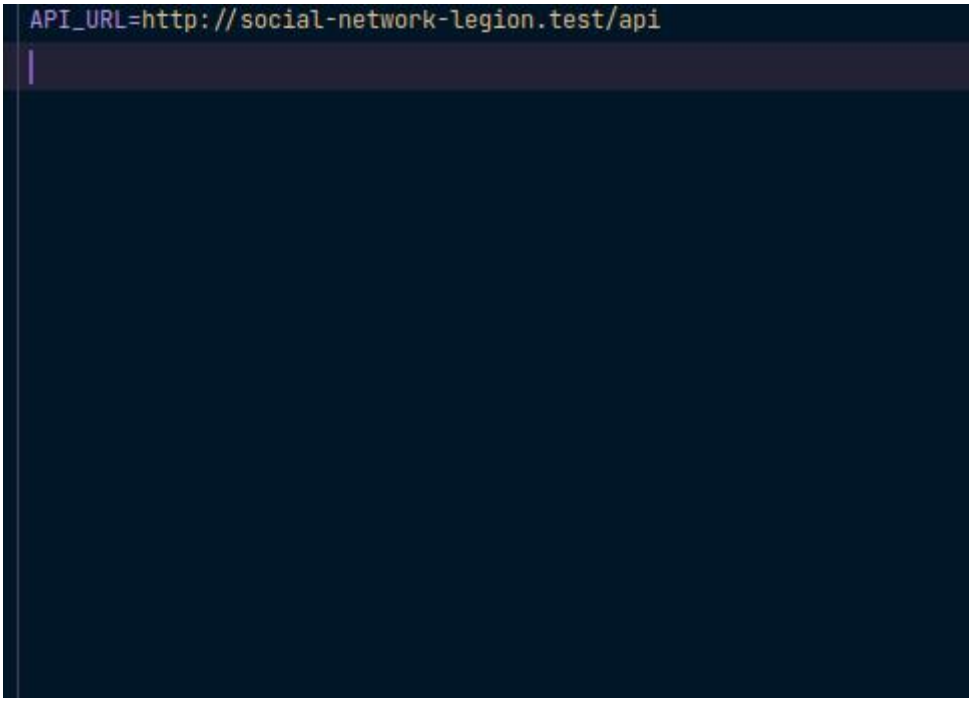


Рисунок 4.1 – Конфігурація проєкту.

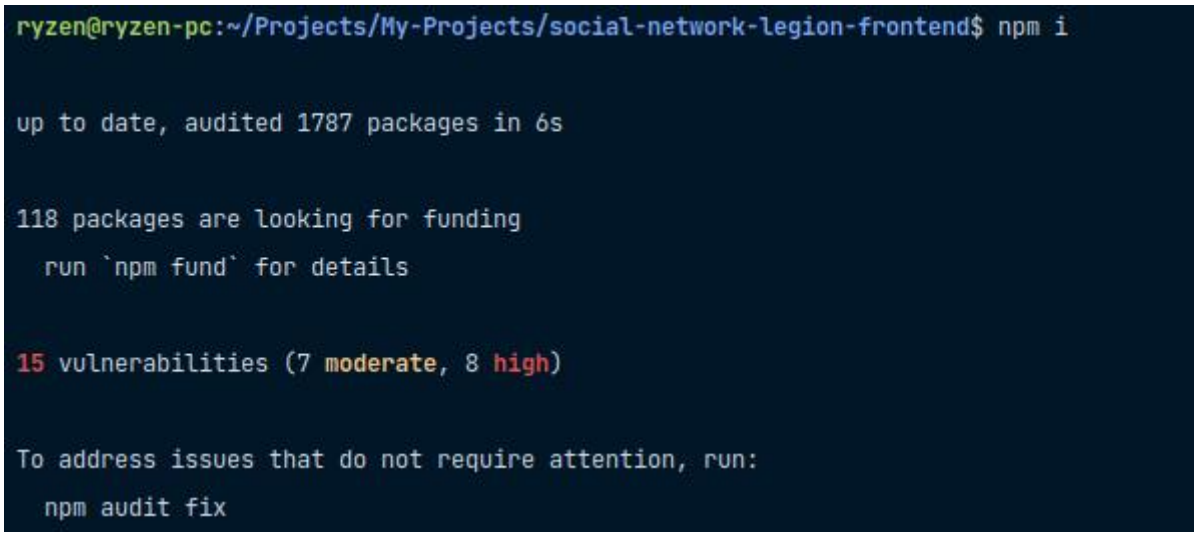
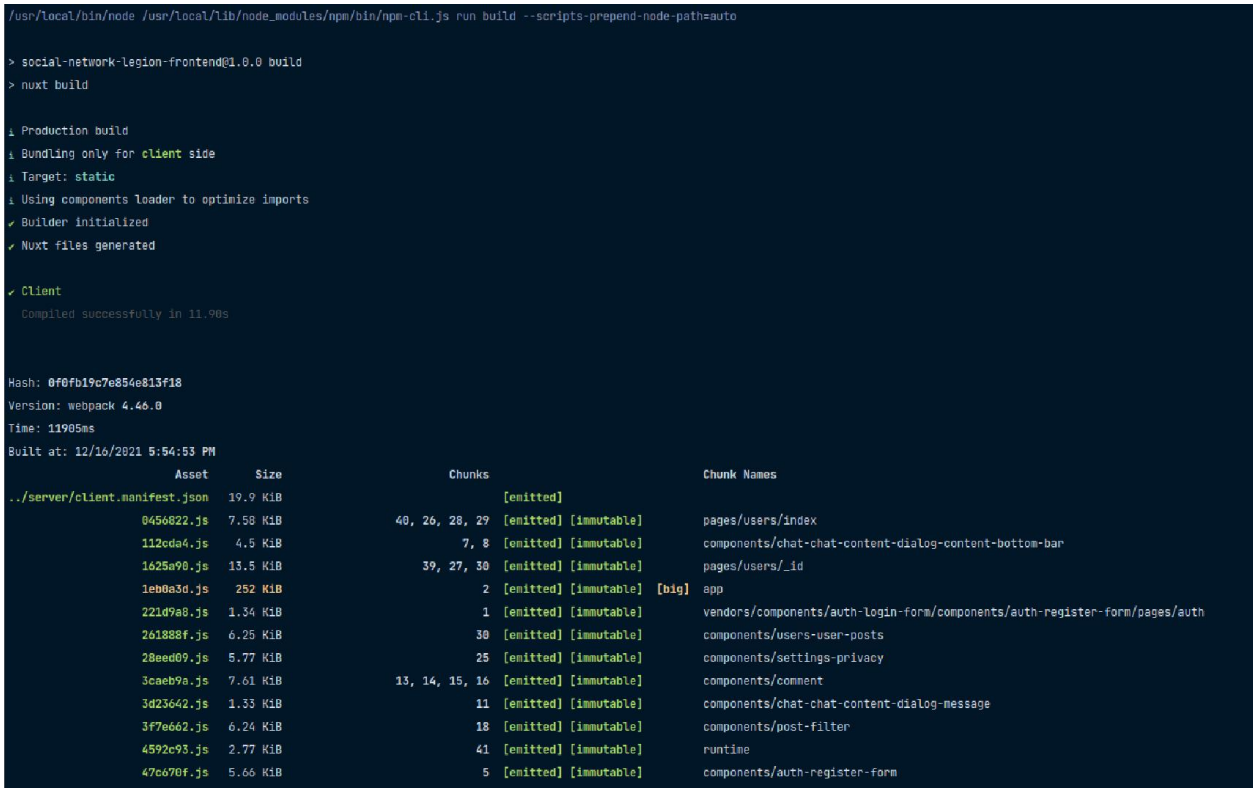
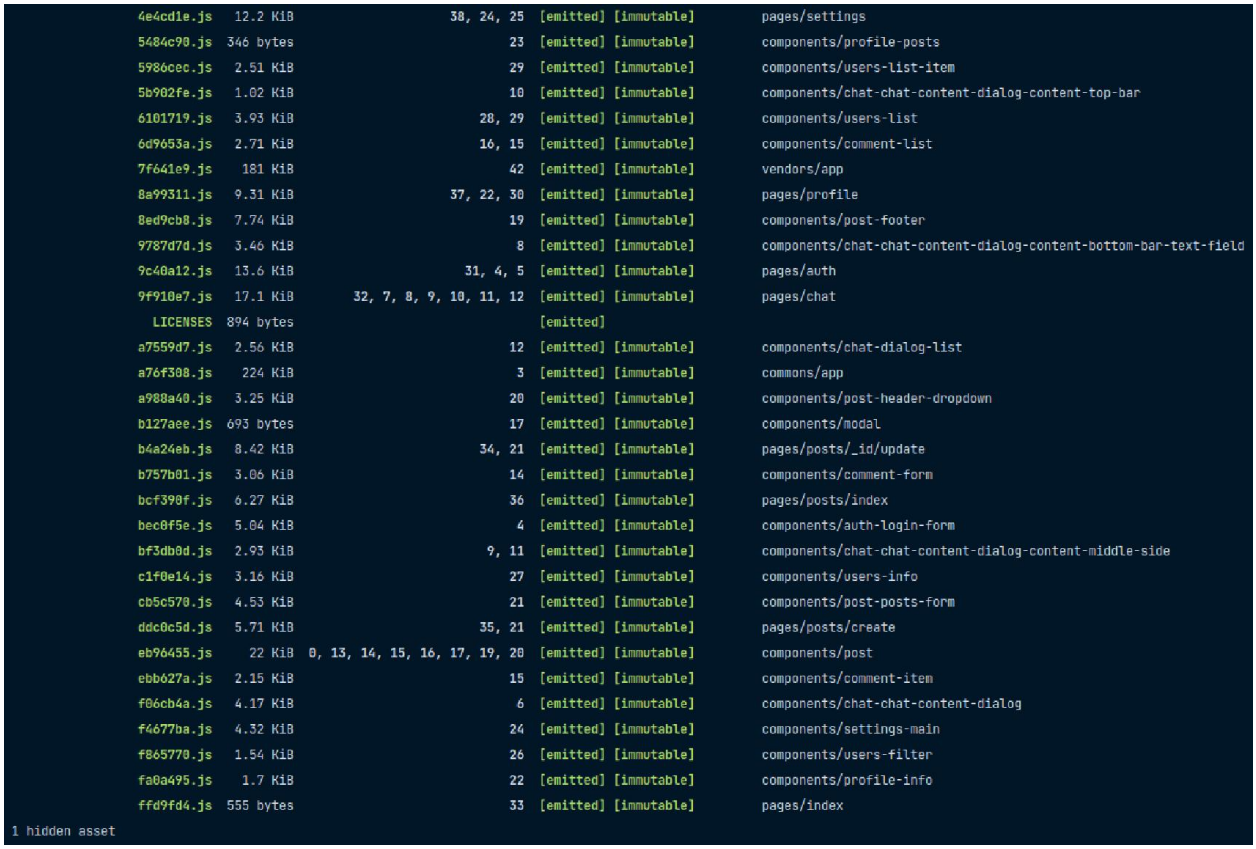


Рисунок 4.2 – Встановлення залежностей.



```
/usr/local/bin/node /usr/local/lib/node_modules/npm/bin/npm-cli.js run build --scripts-prepend-node-path=auto

> social-network-legion-frontend@1.0.0 build
> nuxt build

ℹ Production build
ℹ Bundling only for client side
ℹ Target: static
ℹ Using components loader to optimize imports
✔ Builder initialized
✔ Nuxt files generated

✔ Client
  Compiled successfully in 11.90s


Hash: 0f0fb19c7e854e813f18
Version: webpack 4.46.0
Time: 11905ms
Built at: 12/16/2021 5:54:53 PM
                  Asset       Size                  Chunks                         Chunk Names
../server/client.manifest.json  19.9 KiB                       [emitted]
                0456822.js  7.58 KiB  40, 26, 28, 29  [emitted] [immutable]         pages/users/index
                112cda4.js   4.5 KiB            7, 8  [emitted] [immutable]         components/chat-chat-content-dialog-content-bottom-bar
                1625a90.js  13.5 KiB      39, 27, 30  [emitted] [immutable]         pages/users/_id
                1eb0a3d.js   252 KiB               2  [emitted] [immutable]  [big]  app
                221d9a8.js  1.34 KiB               1  [emitted] [immutable]         vendors/components/auth-login-form/components/auth-register-form/pages/auth
                261888f.js  6.25 KiB              30  [emitted] [immutable]         components/users-user-posts
                28eed09.js  5.77 KiB              25  [emitted] [immutable]         components/settings-privacy
                3caeb9a.js  7.61 KiB  13, 14, 15, 16  [emitted] [immutable]         components/comment
                3d23642.js  1.33 KiB              11  [emitted] [immutable]         components/chat-chat-content-dialog-message
                3f7e662.js  6.24 KiB              18  [emitted] [immutable]         components/post-filter
                4592c93.js  2.77 KiB              41  [emitted] [immutable]         runtime
                47c670f.js  5.66 KiB               5  [emitted] [immutable]         components/auth-register-form
```

Рисунок 4.3 – Компіляція файлів (частина 1).



```
                4e4cd1e.js  12.2 KiB      38, 24, 25  [emitted] [immutable]         pages/settings
                5484c90.js  346 bytes              23  [emitted] [immutable]         components/profile-posts
                5986cec.js  2.51 KiB               29  [emitted] [immutable]         components/users-list-item
                5b902fe.js  1.02 KiB               10  [emitted] [immutable]         components/chat-chat-content-dialog-content-top-bar
                6101719.js  3.93 KiB           28, 29  [emitted] [immutable]         components/users-list
                6d9653a.js  2.71 KiB           16, 15  [emitted] [immutable]         components/comment-list
                7f641e9.js   181 KiB               42  [emitted] [immutable]         vendors/app
                8a99311.js  9.31 KiB      37, 22, 30  [emitted] [immutable]         pages/profile
                8ed9cb8.js  7.74 KiB               19  [emitted] [immutable]         components/post-footer
                9787d7d.js  3.46 KiB                8  [emitted] [immutable]         components/chat-chat-content-dialog-content-bottom-bar-text-field
                9c40a12.js  13.6 KiB        31, 4, 5  [emitted] [immutable]         pages/auth
                9f910e7.js  17.1 KiB  32, 7, 8, 9, 10, 11, 12  [emitted] [immutable]  pages/chat
                  LICENSES  894 bytes                 [emitted]
                a7559d7.js  2.56 KiB               12  [emitted] [immutable]         components/chat-dialog-list
                a76f308.js   224 KiB                3  [emitted] [immutable]         commons/app
                a988a40.js  3.25 KiB               20  [emitted] [immutable]         components/post-header-dropdown
                b127aee.js  693 bytes              17  [emitted] [immutable]         components/modal
                b4a24eb.js  8.42 KiB           34, 21  [emitted] [immutable]         pages/posts/_id/update
                b757b01.js  3.06 KiB               14  [emitted] [immutable]         components/comment-form
                bcf390f.js  6.27 KiB               36  [emitted] [immutable]         pages/posts/index
                bec0f5e.js  5.04 KiB                4  [emitted] [immutable]         components/auth-login-form
                bf3db0d.js  2.93 KiB            9, 11  [emitted] [immutable]         components/chat-chat-content-dialog-content-middle-side
                c1f0e14.js  3.16 KiB               27  [emitted] [immutable]         components/users-info
                cb5c570.js  4.53 KiB               21  [emitted] [immutable]         components/post-posts-form
                ddc0c5d.js  5.71 KiB           35, 21  [emitted] [immutable]         pages/posts/create
                eb96455.js    22 KiB  0, 13, 14, 15, 16, 17, 19, 20  [emitted] [immutable]  components/post
                ebb627a.js  2.15 KiB               15  [emitted] [immutable]         components/comment-item
                f06cb4a.js  4.17 KiB                6  [emitted] [immutable]         components/chat-chat-content-dialog
                f4677ba.js  4.32 KiB               24  [emitted] [immutable]         components/settings-main
                f865770.js  1.54 KiB               26  [emitted] [immutable]         components/users-filter
                fa0a495.js   1.7 KiB               22  [emitted] [immutable]         components/profile-info
                ffd9fd4.js  555 bytes              33  [emitted] [immutable]         pages/index
1 hidden asset
```

Рисунок 4.4. – Компіляція файлів (частина 2).

```
Entrypoint app = 4592c93.js a76f308.js 7f641e9.js 1eb0a3d.js

WARNING in asset size limit: The following asset(s) exceed the recommended size limit (244 KiB).
This can impact web performance.
Assets:
  1eb0a3d.js (252 KiB)
i Ready to run nuxt generate

Process finished with exit code 0
```

Рисунок 4.5 – Компіляція файлів (частина 3).

Процес деплою проєкту (Backend):

- встановити php залежності за допомогою composer;
- налаштувати файл конфігурації проєкту;
- запустити міграції;
- запустити сіди.

```
laradock@518a1aa6a057:/var/www$ composer install
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
Package operations: 1 install, 45 updates, 15 removals
  - Removing tightenco/collect (v8.76.1)
  - Removing spatie/macroable (1.0.1)
  - Removing react/stream (v1.2.0)
  - Removing react/socket (v1.10.0)
  - Removing react/promise-timer (v1.8.0)
  - Removing react/event-loop (v1.2.0)
  - Removing react/dns (v1.8.0)
  - Removing react/cache (v1.1.1)
  - Removing psr/cache (3.0.0)
```

Рисунок 4.6 – Встановлення залежностей.

```
APP_NAME=Legion
APP_ENV=local
APP_KEY=base64:20tP9NfOZQV97PYBi+28tiejfMWhC5/lnXTbmckboLM=
APP_DEBUG=true
APP_URL=http://social-network-legion.test

LOG_CHANNEL=stack
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=mysql
DB_PORT=3306
DB_DATABASE=socialNetworkLegion
DB_USERNAME=root
DB_PASSWORD=root

BROADCAST_DRIVER=log
CACHE_DRIVER=redis
FILESYSTEM_DRIVER=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=redis
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=redis
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailhog
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
```

Рисунок 4.7 – Файл конфігурації (частина 1).

```
MAIL_FROM_ADDRESS=null
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

PUSHER_APP_ID=1317688
PUSHER_APP_KEY=375cd93cd6ef5b27fc74
PUSHER_APP_SECRET=ad83a3260d4240382555
PUSHER_APP_CLUSTER=eu

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

Рисунок 4.8 – Файл конфігурації (частина 2).

```
laradock@518a1aa6a057:/var/www$ php artisan migrate
Nothing to migrate.
```

Рисунок 4.9 – Запуск міграцій.

```
laradock@518a1aa6a057:/var/www$ php artisan db:seed
Seeding: Database\Seeders\RoleSeeder
Seeded:  Database\Seeders\RoleSeeder (22.75ms)
Seeding: Database\Seeders\UserSeeder
```

Рисунок 4.10 – Запуск сідів.

ВИСНОВОК

Мета дипломної роботи полягала в розробці модуля «Розроблення інтелектуальної соціальної мережі».

Цей модуль дозволяє користувачам системи мати необхідну інформацію для написання публікацій, коментарів, повідомлень, чат боту, а також встановлення лайків, додавання в друзі і т.д.

При створенні проєкту був використаний такий технологічний стек:

- PHP 8.0;
- JS;
- Vue;
- Nuxt;
- TypeScript;
- Laravel;
- MySQL.

Такий стек технологій дозволив швидко розробити велику частину функціонала, комфортну сборку проєкту та швидку роботу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. JavaScript | MDN [Електронний ресурс]: [Вебсайт]. – Електронні дані. – Режим доступу: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата звернення 10.05.2020). – Назва з екрана.
2. Request lifecycle [Електронний ресурс]: [Вебсайт]. – Електронні дані. – Режим доступу: https://laravel.com/docs/7.x/lifecycle (дата звернення 12.05.2020). – Назва з екрана.
3. Laravel — экосистема, а не просто PHP-фреймворк[Електронний ресурс]: [Вебсайт]. – Електронні дані. – Режим доступу: https://habr.com/ru/post/334776/ (дата звернення 20.05.2020). – Назва з екрана.
4. Введение – Vue.js [Електронний ресурс]: [Вебсайт]. – Електронні дані. – Режим доступу: https://ru.vuejs.org/v2/guide/index.html (дата звернення 20.05.2020). – Назва з екрана.
5. Vue.js для сомневающихся. Все, что нужно знать [Електронний ресурс]: [Вебсайт]. – Електронні дані. – Режим доступу: https://habr.com/ru/post/329452/ (дата звернення 25.05.2020). – Назва з екрана.
6. Why MySQL? [Електронний ресурс]: [Вебсайт]. – Електронні дані. – Режим доступу: https://www.mysql.com/why-mysql/ (дата звернення 26.05.2020). – Назва з екрана.
7. Руководство по MySQL [Електронний ресурс]: [Вебсайт]. – Електронні дані. – Режим доступу https://metanit.com/sql/mysql/ (дата звернення 30.05.2020). – Назва з екрана.
8. Гид по ручному тестированию приложений: преимущества, этапы и методологии [Електронний ресурс]: [Вебсайт]. – Електронні дані. – Режим доступу https://habr.com/ru/company/skillbox/blog/418889/ (дата звернення 01.06.2020). – Назва з екрана.

9. Getting started – What is Git? [Електронний ресурс]: [Вебсайт]. – Електронні дані. – Режим доступу https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F (дата звернення 01.06.2020). – Назва з екрана.
10. Что такое PHP? [Електронний ресурс]: [Вебсайт]. – Електронні дані. – Режим доступу https://www.php.net/manual/ru/intro-whatis.php (дата звернення 01.06.2020). – Назва з екрана.
11. Getting started [Електронний ресурс]: [Вебсайт]. – Електронні дані. – Режим доступу https://docs.npmjs.com/getting-started/ (дата звернення 01.06.2020). – Назва з екрана.
12. Введение: знакомство с React [Електронний ресурс]: [Вебсайт]. – Електронні дані. – Режим доступу https://ru.reactjs.org/tutorial/tutorial.html (дата звернення 01.06.2020). – Назва з екрана.
13. Introduction to Angular concepts Електронний ресурс]: [Вебсайт]. – Електронні дані. – Режим доступу https://angular.io/guide/architecture (дата звернення 01.06.2020). – Назва з екрана.

ДОДАТОК А

ER-ДІАГРАМА БАЗИ ДАНИХ

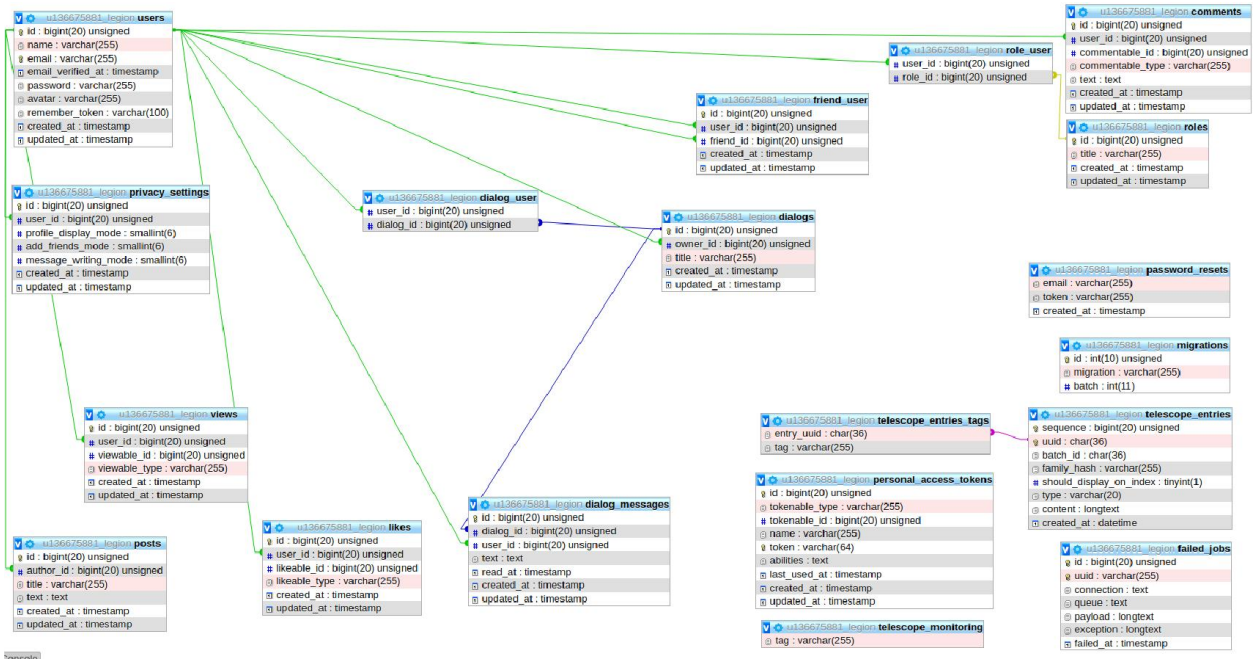


Рисунок А.1 – ER-діаграма

ДОДАТОК В

ВИХІДНИЙ КОД ОСНОВНИХ КОМПОНЕНТІВ

1. Файл components/Auth/LoginForm.vue

```
<template>
<div class="login-form-component">
<div class="fs-2 mt-4 fw-bold text-center mb-3">Log in</div>
<form @submit.prevent="send">
<div class="form-group mb-4">
<label for="email" class="form-label">Email</label>
<div class="input-group">
<div class="input-group-text">
<font-awesome-icon :icon="faEnvelope" fixed-width />
</div>
<input type="text"
        class="form-control"
        :class="{ 'is-invalid': errors.password }"
        id="email"
        placeholder="Type your email"
        v-model="form.email"
        @input="handleInput('email')">
<div class="invalid-feedback">
{{ errors.email }}
</div>
</div>
</div>
<div class="form-group mb-4">
<label for="password" class="form-label">Password</label>
<div class="input-group">
<div class="input-group-text">
```

```
<font-awesome-icon :icon="faLock" fixed-width />
</div>
<input type="password"
        class="form-control"
        :class="{ 'is-invalid': errors.password }"
        id="password"
        placeholder="Type your password"
        v-model="form.password"
        @input="handleInput('password')">
<div class="invalid-feedback">
{{ errors.password }}
</div>
</div>
</div>
<!--<div class="text-end text-secondary mb-4">-->
<!--<span>Forgot password?</span>-->
<!--</div>-->
<button type="submit"
    class="btn btn-dark w-100">Log in</button>
</form>
</div>
</template>

<script lang="ts">
import Vue from 'vue';
import { mapActions } from "vuex";
import { faEnvelope } from '@fortawesome/free-solid-svg-icons'
import { faLock } from '@fortawesome/free-solid-svg-icons'

import ValidationError from "~/classes/Errors/ValidationError";
import LocalStorageService from "~/services/LocalStorageService";
```

```
export default Vue.extend({
  computed: {
faEnvelope() {
      return faEnvelope;
    },
faLock() {
      return faLock;
    }
  },
  data: () => {
    return {
      form: {
        // email: '', TODO: Need to fix.
        // password: '',
        email: 'admin@admin.admin',
        password: '12345678'
      },
      errors: {
        email: '',
        password: ''
      }
    }
  },
  methods: {
    ...mapActions("auth", ["setUser"]),

setErrors(errors: { email?: string[], password?: string[] }): void {
      if (errors.email) {
this.errors.email = errors.email[0];
      }
```

```
      if (errors.password) {
this.errors.password = errors.password[0];
      }
    },
    // TODO: Need to fix it.
removeError(type: string = 'all'): void {
      if (type === 'all') {
this.errors.email = '';
this.errors.password = '';
      } else if (type === 'email') {
this.errors.email = '';
      } else if (type === 'password') {
this.errors.password = '';
      }
    },
handleInput(type: string): void {
this.removeError(type);
    },
    async send(): Promise<void> {
      const email = this.form.email;
      const password = this.form.password;
this.removeError('all');
      try {
        const response = await this.$api.login({ email, password });
        const token = response.data.type + ' ' + response.data.token;

        LocalStorageService.setToken(token);
this.$axios.setToken(token);

        await this.getUser();
```

```
        await this.$router.push('/');
      } catch (e) {
        if (e instanceof ValidationError) {
this.setErrors(e.errors);
        }
      }
    },
    async getUser(): Promise<void> {
      try {
        const response = await this.$api.user.me();

this.setUser(response.data)
      } catch (e) {}
    }
  }
})
</script>
<style lang="scss">
  .login-form-component {
    form {
.form-group {
.input-group {
.input-group-text {
            background-color: transparent;
            border: none;
            border-bottom: 2px solid #d9d9d9;
          }

          input {
            border: none;
            outline: none;
```

```
            box-shadow: none;
            border-bottom: 2px solid #d9d9d9;
}
        }
                }
  }
</style>
```

2. Файл components/Post/Post.vue

```
<template>
<div class="card mb-3 shadow">
<!--<img src="..." class="card-img-top" alt="...">-->
<div class="card-body">
<div class="d-flex justify-content-between align-items-end">
<h5 class="card-title">{{ post.title }}</h5>
<PostHeaderDropdown v-if="post.author.id === user.id"
:post="post"
                    @onDelete="handleDelete" />
</div>
<p class="card-text mt-2">{{ isSimple ? shortDescription : post.text }}</p>
<div class="d-flex justify-content-end">
<button v-if="isSimple"
              type="button
              class="btn btn-outline-dark"
              @click="handleOpenModal">
          More
</button>
</div>
</div>
<div class="card-footer">
<PostFooter ref="PostFooter"
:post="post"
```

```
            @onAddLike="handleAddLike"
            @onAddView="handleAddView"/>
</div>

<Modal v-if="isSimple"
       :id="modalId">
<template v-slot:content>
<Post :post="post" :is-simple="false"></Post>
<Comment></Comment>
</template
</Modal>
</div>
</template>

<script lang="ts">
import Vue from 'vue';
import { mapGetters } from "vuex";
import { Modal as BootstrapModal } from 'bootstrap';
import { faComments } from "@fortawesome/free-regular-svg-icons/faComments";

import PostModel from "~/classes/Models/PostModel";
import PostHeaderDropdown from "~/components/Post/PostHeaderDropdown.vue";
import PostFooter from "~/components/Post/PostFooter.vue";
import Modal from "~/components/Modal/Modal.vue";
import Comment from "~/components/Comment/Comment.vue";

export default Vue.extend({
  name: 'Post',
  components: {
    PostHeaderDropdown,
    PostFooter,
```

```
    Modal,
    Comment
  },
  props: {
    post: {
      type: Object as () => PostModel,
      required: true
    },
    isSimple: {
      type: Boolean,
      required: false,
      default: true
    }
  },
  computed: {
    ...mapGetters('auth', ['user']),
shortDescription(): string {
      return this.post.text.slice(0, 300) + "...";
    },
modalId(): string {
      return 'modal-post-' + this.post.id;
    },
faComments() {
      return faComments;
    }
  },
  data: (): { modalInstance: BootstrapModal | null} => ({
    modalInstance: null
  }),
  methods: {
initModal(): void {
```

```
    const el = document.getElementById((this.modalId as string));


    if (el) {
this.modalInstance = new BootstrapModal(el     }
   },
handleAddLike(id: number): void {
this.$emit('onAddLike', id);
   },
handleAddView(id: number): void {
this.$emit('onAddView', id);
   },
handleOpenModal(): void {
this.modalInstance?.show();
   },
handleDelete(id: number): void {
this.$emit('onDelete', id);
   }
 },
mounted() {
this.initModal();
 }
}
</script>
<style lang="scss">
.icon {
 margin-left: 3px;
}
</style>
```

3. Файл app/Models/User/User.php

```
<?php
```

```
namespace App\Models\User;


use App\Models\Dialog\Dialog;

use App\Models\Dialog\DialogMessage;

use App\Models\Post;

use App\Traits\Models\User\Friendable;

use Illuminate\Database\Eloquent\Builder;

use Illuminate\Database\Eloquent\Collection;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Relations\BelongsToMany;

use Illuminate\Database\Eloquent\Relations\HasMany;

use Illuminate\Database\Eloquent\Relations\HasOne;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

use Illuminate\Support\Carbon;

use Laravel\Sanctum\HasApiTokens;


/**

 * @property int $id

 * @property string $name

 * @property string $email

 * @property Carbon $email_verified_at

 * @property string $password

 * @property string $avatar

 * @property string $remember_token

 * @property Carbon $created_at

 * @property Carbon $updated_at

 *

 * @property-read PrivacySetting $privacySettings

 * @property-read Collection<int, Role> $roles

 * @property-read Collection<int, DialogMessage> $dialogMessages
```

```
 * @property-read Collection<int, Dialog> $dialogsOwner
 *
 * @property-read int $posts_count
 *
 * @method static User|Builder query()
 */
class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable, Friendable;
    /**
     * @var string[]
     */
    protected $fillable = [
        'name',
        'email',
        'password',
/**
     * @var array
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];
    /**
     * @var array
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
    /**
     * @var string[]
```

```
     */
    protected $with =
        'privacySettings'
    ];
    /**
     * @var string[]
     */
    protected $attributes = [
        'avatar' => 'core/user.svg'
    ];
    public function posts(): HasMany
    {
        return $this->hasMany(Post::class, 'author_id', 'id');
    }
    public function dialogMessages(): HasMany
    {
        return $this->hasMany(DialogMessage::class);
    }
    public function dialogsOwner(): HasMany
    {
        return $this->hasMany(Dialog::class);
    }
    public function privacySettings(): HasOne
    {
        return $this->hasOne(PrivacySetting::class);
    }
    public function roles(): BelongsToMany
    {
        return $this->belongsToMany(Role::class);
    }
    public function dialogs(): BelongsToMany
```

```
    {
        return $this->belongsToMany(Dialog::class);
    }
    public function friends(): BelongsToMany
    {
        return $this->belongsToMany(
User::class,
            'friend_user',
            'user_id',
            'friend_id'
        );
    }
    public function isAdmin(): bool
    {
        return $this->roles()->getAdminQuery()->exists();
    }

    public function isUser(): bool
    {
        return $this->roles()->getUserQuery()->exists();
    }
    public function setUserRole(): void
    {
        $this->roles()->attach($this->roles()->getUserQuery()->select('id')->pluck('id'));
    }
}
```

4. Файл app/Models/Post.php

```
<?php

namespace App\Models;
```

```
use App\Models\User\User;
use App\Traits\Models\Commentable;
use App\Traits\Models\Likeable;
use App\Traits\Models\Viewable;
use Illuminate\Database\Eloquent\Builder;
use Illuminate\Database\Eloquent\Collection;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Support\Carbon;

/**
 * @property int $id
 * @property int|null $author_id
 * @property string $title
 * @property string $text
 * @property Carbon $created_at
 * @property Carbon $updated_at
 *
 * @property-read User|null $author
 * @property-read View[]|Collection $views
 *
 * @property-read int $views_count
 * @property-read int $likes_count
 * @property-read int $comments_count
 *
 * @method static Builder|Post query()
*/
class Post extends Model
{
    use HasFactory, Viewable, Likeable, Commentable;
```

```
    /**
     * @var string[]
     */
    protected $fillable = [
        'title',
        'text'
    ];
    public function author(): BelongsTo
    {
        return $this->belongsTo(User::class, 'author_id', 'id');
    }
}
```