Міністерство освіти і науки України
Державний університет «Одеська Політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра системного програмного забезпечення

Гордієнко Олексій Михайлович
студент групи АС-161

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Веб-застосування з інтегрованою нейронною мережею для підвищення конверсії туристичних місць

Спеціальність:
121 – Інженерія програмного забезпечення

Освітня програма:
Інженерія програмного забезпечення

Керівник:
Писаренко Катерина Олександрівна
канд. техн. наук, доцент

Одеса – 2021

ЗМІСТ

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ 4
АНОТАЦІЯ 6
ВСТУП 7
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 9
1.1 Вступ 9
1.2 Аналіз аналогів 10
1.3 Глосарій 14
2 РОЗРОБКА ВЕБ-СЕРВІСУ 15
2.1 Визначення модулів системи 15
2.2 Вимоги до модулів 17
2.3 Опис нейронної мережі 18
3 СПЕЦИФІКАЦІЯ ВИМОГ 21
3.1 Підстави для розробки 21
3.2 Призначення розробки 22
3.3 Вимоги до програмного продукту 23
3.4 Вимоги до програмної документації 35
4 ПРОЕКТУВАННЯ ВЕБ-СЕРВІСУ 36
4.1 Опис програми 36
4.2 Структура та опис бази даних 39
4.3 Проектування програмних класів 41
4.4 ER-модель розроблюваного продукту 44
4.5 Діаграма активності 46
4.6 Діаграми послідовності 49
4.7 Інтерфейс користувача 51
5 РЕАЛІЗАЦІЯ ТА ВИПРОБУВАННЯ ВЕБ-СЕРВІСУ 54
5.1 Програмна реалізація інтерфейсу 54
5.2 Реалізація нейронної мережі 60
5.3 Керівництво користувача 62

5.4 Інструкція із налаштування програмної системи 62
5.5 Керівництво системного програміста 63
5.6 Керівництво з технічного обслуговування 66
6 ТЕСТУВАННЯ ВЕБ-СЕРВІСУ 68
6.1 Призначення і область використання 68
6.2 Технічні характеристики 68
6.3 Очікувані техніко-економічні показники 69
6.4 Оцінювання якості 71
6.5 Об’єкт випробувань 74
6.6 Мета випробувань 75
6.7 Кошти і порядок випробувань 75
6.8 Методи випробувань 75
6.9 Тестування методом симуляції дій користувача 77
6.10 Тестування моделі нейронної мережі 79
ВИСНОВКИ 81
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 82
ДОДАТОК А. ЛІСТИНГ ТА ТЕСТУВАННЯ ЕЛЕМЕНТІВ ВЕБ-СЕРВІСУ
ДОДАТОК Б. РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Міністерство освіти і науки України
Державний університет «Одеська Політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра системного програмного забезпечення

Рівень вищої освіти: другий (магістерський)

Спеціальність: 121 – Інженерія програмного забезпечення

Спеціалізація: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Любченко В.В.
«___»_____20__р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Гордієнка Олексія Михайловича, група АС-161

1. Тема роботи: *Веб-застосування з інтегрованою нейронною мережею для підвищення конверсії туристичних місць*

Керівник роботи: *Писаренко Катерина Олександрівна, канд. техн. наук, доцент*

Затверджені наказом ректора від «25» жовтня 2021р. № 374-в

2. Зміст роботи: підстави для розробки, опис предметної області, формалізація вимог, проектування системи, програмна реалізація, тестування.

3. Перелік ілюстративного матеріалу:

Згідно зі слайдами презентації

4. Консультанти розділів роботи:

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

5. Дата видачі завдання: «___»_____20__р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	Підстави для розробки	10.09.2021 – 15.09.2021	вик.
2	Опис предметної області	16.09.2021 – 23.09.2021	вик.
3	Формалізація вимог	25.09.2021 – 12.10.2021	вик.
4	Проектування програмної системи	13.10.2021 – 25.10.2021	вик.
5	Програмна реалізація та тестування	26.10.2021 – 21.11.2021	вик.
6	Оформлення пояснювальної записки та графічного матеріалу	22.11.2021 – 29.11.2021	вик.

Здобувач вищої освіти _____ *О.М. Гордієнко*

Керівник роботи _____ *К.О. Писаренко*

АНОТАЦІЯ

Метою роботи є підвищення конверсії серед місць, що обділені увагою серед туристів та місцевих жителів завдяки розробці «Eternal Radiance», що надає зручний та звичний для користувачів інтерфейс, полегшує та прискорює процес вивчення місцевості, спираючись на досвід місцевих жителів, що добре знають ці місця.

Методи розробки базуються на мові програмування TypeScript, середовищі розробки Visual Studio Code з використанням фреймворків React TS, Node JS, веб-sockets та реляційної БД MySQL.

Як результат роботи створено соціальну мережа-путівник з використанням нейронної мережі для аналізу фотографій, що завантажують користувачі.

Ключові слова: веб-сервіс, соціальна мережа, путівник, нейронна мережа, перцептрон, Typescript, MySQL, React TS, Node JS, TensorflowJS web-socket, socket.io.

ABSTRACT

The aim of the work is to increase the conversion of places that are deprived of attention among tourists and locals through the development of "Eternal Radiance", which provides a user-friendly interface, facilitates and accelerates the process of exploring the area, drawing on the experience of locals who know these places.

Development methods are based on the TypeScript programming language, the Visual Studio Code development environment using web-sockets, React TS, Node JS frameworks, and the MySQL relational database.

As a result, a social network-guide was created using a neural network to analyze photos uploaded by users.

Keywords: web-service, social network, guide, neural network, perceptron, Typescript, MySQL, ReactTS, Node JS, TensorFlowJS, web-socket, socket.io.

ВСТУП

Актуальність розробки в сучасних умовах сервісів-путівників визначається тим, що вони надають зручний набір інструментів для пошуку необхідного місця, допомагають у подорожах, відрядженнях, під час пошуку закладів та розваг.

Сучасна людина зіштовхнулася з потребою частих подорожей до інших міст чи держав через роботу, або відпочинок. При цьому виникає проблема географічної адаптації, коли людина, що потрапила у незнайоме місце, не знає, як дістатися до пункту призначення. Тим паче, місцеві жителі можуть розмовляти на мові, котрої людина не знає, тому пошук шляху стає дуже складним.

Також, місцеві путівники можуть показувати обмежену кількість місць, рекомендованих для відвідування, при цьому багато не менш цікавих закладів, парків та ін. залишається поза увагою людини.

Крім того, самі місцеві жителі можуть погано знати місто, в якому живуть, бо, вивчення нових місць вимагає багато часу, що, у рамках сучасності, є дуже цінним ресурсом.

Отже, проблемою роботи виступає відсутність різноманітності сервісів-путівників, що дозволяють швидко знайти бажане місце, а також практично повна відсутність ринкових пропозицій серед соціальних мереж, що дозволяють знаходити компанію по інтересам для відвідування бажаних місць.

Метою роботи є підвищення конверсії серед місць, що обділені увагою серед туристів та місцевих жителів завдяки розробці «Eternal Radiance», що надає зручний та звичний для користувачів інтерфейс, полегшує та прискорює процес вивчення місцевості, спираючись на досвід місцевих жителів, що добре знають ці місця.

Основним завданням роботи є розробка єдиної універсальної системи на базі веб-сервісу з інтегрованою нейронною мережею, що дозволить людині вивчати місцевість на основі досвіду інших людей. Це дозволить користувачам знаходити нові місця, що можна відвідати, будь це який-небудь маловідомий бар з нестандартною тематикою, чи парк, що не потрапив до місцевого путівника через

непопулярність, або навіть схил, що знаходиться далеко від урбаністичної місцевості.

Також, система дозволяє знизити соціальний поріг для пошуку людей зі схожими інтересами, для пошуку компанії, щоб відвідати якесь місце разом.

Для повноцінної реалізації завдання потрібно створення соціальної мережі з інтегрованою нейронною мережею, що включає до себе роботу з Google-картою, можливістю створення групових чатів з користувачами зі схожими інтересами, пошуку компанії та нових місць для відвідання [2].

Робота поділяється на 2 розділи: Front-end та Back-end розробка. Front-end розробка буде проводитися з використанням мови програмування TypeScript на базі фреймворка React JS. Back-end розробка буде проводитися на базі мови програмування JavaScript та SQL з використанням фреймворків Visual Studio Code та MySQL Workbench. Зв’язування розділів front та back відбувається на основі мови JavaScript. Сервер, що організує роботу загальних модулів системи, чату та роботи нейронної мережі – Node JS.

Розробка повинна бути проведена в чотири стадії: розробка технічного завдання; розробка прототипу та мінімального функціоналу; тестування; впровадження.

На стадії «Розробка технічного завдання» повинен бути виконаний етап розробки, узгодження і затвердження цього технічного завдання.

На стадії «Розробка прототипу та мінімального функціоналу» повинна бути виконана робота по програмуванню (кодування) і налагодженні програми.

На стадії «Тестування» повинен бути перевірені усі компоненти веб-сервісу на відповідність вимогам, що зазначені у розділах «Вимоги до функціональних характеристик», та «Вимоги до надійності»

На стадії «Впровадження» розроблений веб-сервіс має бути розміщений на публічному сервері та відкритий доступ користувачам до функцій соціальної мережі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Вступ

В якості програмного продукту обраний веб-сервіс у вигляді соціальної мережі. Умовна назва програмного продукту «Eternal Radiance». Цільовою предметною областю продукту є туристи та люди, що мандрують.

Відповідно до предметної області система будується з урахуванням таких особливостей: користувач реєструється у системі та отримує можливість перегляду постів інших користувачів, а також створення особистих постів. Кількість постів, що може створити користувач одразу після реєстрації – необмежено. Після створення посту, він проходить перевірку – спочатку модераторами системи для створення учбової вибірки, а надалі – за участі нейронної мережі для автоматичної перевірки постів. Нейронна мережа передбачена для автоматизації перевірки постів та коментарів на дотримання правил використання сервісу (відсутність мату, дотримання цензурі, «постів-пустушок», потенційно небезпечних місць у відмітках).

Під час реєстрації система перевіряє вказані дані на валідність. Якщо в системі вже зареєстрований користувач із вказаною поштою, то користувачу пропонується процедура відновлення паролю.

Пошта перевіряється завдяки відсиланню підтверджуючого письма на вказану адресу. При цьому користувачу видається унікальний реєстраційний токен, що записується до БД та дублюється у посилання в листі. Якщо користувач перейшов по посиланню у письмі та токени не співпадають, то дані користувача видаляються.

Для організації контролю контенту в системі, окрім нейронної мережі, передбачена багаторівнева система рейтингу, що включає рейтинг користувача, рейтинг поста та рейтинг коментарів. Рейтинг користувача впливає на функціонал системи, що доступний користувачу: високий рейтинг – повний доступ до функціоналу, середній – обмеження по кількості публікацій, низький – заборона

публікацій та спілкування з іншими користувачами (користувач може тільки переглядати чужі пости). Рейтинг поста впливає на рейтинг автора та показує популярність місця серед інших користувачів; чи правильний опис був опублікований, чи рекомендують це місце для візиту тощо. Рейтинг коментаря впливає на рейтинг поста та підтверджує достовірність відгуку на пост.

Кожний користувач вказує інформацію щодо персональних даних (ПІБ, дата народження, стать, фото), особисті переваги щодо підбору системою рекомендованих для візиту місць. Завдяки рейтинговій системі інші користувачі мають можливість уникати небажаних користувачів, або людей, що порушують основні правила роботи з системою (правила користування, створення поміток-пустушок, або розміщення потенційно небезпечної інформації).

Кожний пост, створений користувачем, включає до себе: координати місця, що відображають мітку на карті та пов’язані з нею, фотографії місця, короткий відгук, а також області для коментарів та оцінки посту. Також пости можуть бути публічними та приватними. Приватні пости не проходять перевірку та доступні лише автору. Приватні пости необхідні у тому разі, якщо користувачу необхідно зберегти якесь місце для швидкого доступу до нього, на випадок регулярного відвідування.

1.2 Аналіз аналогів

На сьогоднішній день існує багато сервісів-путівників, що частково виконують задачі веб-сервісу, що розробляється.

Аналогами веб-сервісу є:

1) Google Maps Local Guides

GoogleMaps - це комплекс програм, створених на базі безкоштовного сервісу картографії та технології, що використовується Google. Цей сервіс використовується для пошуку інформації на карті з відмітками пам'яток, організацій та ін. [2].

1. Користувач відвідує якесь громадське місце (парк, кафе, торговий центр, магазин, кінотеатр, тощо)
2. Система відстежує місця, що відвідав користувач та пробув там більше 10 хвилин та пропонує користувачу написати короткий опис місця та додати фотографії (якщо користувач додатково використовує Google Photoes, то фотографії завантажуються автоматично)
3. Створений відгук проходить перевірку та публікується у системі

Переваги: автоматичне відстеження переміщення користувача (система автоматично пропонує створити пост за простим шаблоном); профілі користувачів (допомагають визначити достовірність даних); інтеграція інших програмних продуктів компанії (Google Maps, Google Photoes).

Недоліки: погана адаптованість для пошуку «місць, що можуть сподобатися» (сервіс лише дає можливість відстежити заздалегідь відомі місця, що відмічені на карті).

2) 2Gis

«2ГІС» - міжнародна картографічна компанія, що випускає однойменні електронні довідники з картами міст з 1999 року. Головний офіс «2ГІС» знаходиться в Новосибірську. [3]

1. Користувач знаходить необхідне місце та відвідує його
2. Система видає опис даного місця

Переваги: адаптивна побудова маршруту на карті; точний опис організації, що відвідав користувач (включаючи контактні дані фірми); просунута картографічна система (включно опису кількості та розташування під'їздів, поверхів, динамічне оновлення даних про громадський транспорт).

Недоліки: обмеженість карт (система містить у собі дуже точні, але уривки карт); сервіс може запропонувати деякі місця для відвідування, але інформація про ці місця дуже суха, іноді неточна та неповна; не всі організації мають опис; відсутність опису для громадських місць; відсутність взаємодії з іншими користувачами.

3) Zenly

Zenly – французький мобільний сервіс, що відстежує друзів та знайомих на карті міста. Влітку 2017 компанія Snap (раніше Snapchat) купила Zenly та розвила його до сучасного виду. [4]

1 Користувач реєструється в системі та повідомляє про це людей, що знаходяться у його контактній книжці
2 Знайомі користувача, що вже зареєстровані в системі, бачать повідомлення та додають користувача у список друзів
3 Користувачі, що відмічені як «Друзі» відтепер відстежують геопозицію одне одного

Переваги: оновлення геопозиції користувачів у реальному часі; сучасний, молодіжний інтерфейс; просунута система взаємодії користувачів (чати, відстеження «вечірок» -- груп друзів від 2 чоловік, можливість приховати своє місце розташування від конкретного користувача, або від всіх).

Недоліки: відсутність головної (у рамках розробки) функції – створення постів (система лише відстежує геопозицію); як наслідок -- неможливість прокладання маршруту, опису місць, відсутність рейтингової системи

4) Instagram

Instagram – соціальна мережа, що використовує в якості публікацій фотографії або відеозаписи. Розроблена 2010 року, у 2013 році викуплена Facebook. Дозволяє розповісти про себе та оточення. Слова підкріплені цікавою картинкою, що зроблена на камеру смартфона. [5]

1. Користувач створює пост (завантажує фото, редагує його та додає опис)
2. Інші користувачі, що підписані на нашого користувача, бачать пост та можуть його оцінювати та залишати коментарі.

Переваги: просунута система створення постів (мінімалістичний інтерфейс, широкий вибір функціоналу); автоматична модерація постів (з використанням нейронної мережі); взаємодія користувачів (чати, підписки)

Недоліки: погано розвинена функція створення гео-поміток (система погано виконує функцію довідника та більше схожа на блог [6]).

Аналіз сервісів-аналогів, порівняно з розроблюваним сервісом, представлено у таблиці 1.3.

Таблиця 1.3 – Порівняльна характеристика аналогів

	Google Maps local guides	2Gis	Zenly	Instagram	**Eternal Radiance**
Профіль користувача	+	+	+	+	+
Фільтр постів	+/-	+/-	-	+/-	+
Інтерактивний підбір постів	+/-	-	-	+	+/-
Розміщення постів користувачами	+	-	-	+	+
Відгуки	+	+	-	+	+
Спілкування з іншими користувачами	+	-	+	+	+
Створення індивідуальних гео-поміток	-	-	+/-	+/-	+
Можливість створення групових чатів	-	-	+	+	+

Як видно, існуючі путівники мають певні обмеження у використанні – вони дозволяють знайти шлях до вже заздалегідь відомого місця, прокласти найшвидший маршрут, спираючись на щільність заторів на дорогах тощо. Але якщо людина ще не знає, куди можна піти, не знає, з ким можна піти, то вказані путівники вже не можуть вирішити дану задачу. Існує багато сервісів, що, навпаки, можуть підказати місце для відвідування, але не можуть виконати наступних вимог – прокладання шляху, демонстрація карти тощо.

1.3 Глосарій

Глосарій – це словник значень предметної області для налагодження спілкування між замовником та розробником сервісу.

Розглянемо та визначимо основні терміни предметної області.

Путівник – друкований, електронний або аудіовізуальний довідник про якесь місто, історичне місці, музеї, туристичний маршрут. Використовуються туристами для кращого орієнтування в незнайомій місцевості. Композиція путівника часто підпорядкована рекомендованим маршрутами огляду визначних пам'яток описуваної місцевості, що містить відомості про країну, регіоні, місті, туристському маршруті [1].

Гео-помітка – це мітки, які використовуються на картах, частіше - електронних або GPS-навігаторах. Можуть виглядати по різному, частіше зображуються у вигляді невеликих кіл з точкою або символом всередині. Від кола відходить невеликий стрілкоподібний відросток, який вказує конкретне розташування гео-помітки. У цифровому вигляді – кортеж, що містить координати місця на карті.

Пост – цифровий блок даних, створений користувачем. Може містити у собі фотографії, текстовий опис, відео- чи аудіо- контент.

Стрічка постів – формат даних, що використовується для подачі користувачам інформації, що швидко оновлюється. Складається з постів інших користувачів.

Онлайн-путівник потребує використання цифрового гаджету – смартфон, ноутбук чи настільний комп’ютер, що має стабільне підключення до мережі Інтернет. Також потребує мінімальних знань у використанні цифрових карт а також бажанням допомогти іншим користувачам краще вивчити місцевість.

2 РОЗРОБКА ВЕБ-СЕРВІСУ

2.1 Визначення модулів системи

Архітектура проекту продемонстрована на рис. 2.1. Вона умовно поділена на три частини:

1) уявлення зі сторони користувача – як він бачить сервіс своїми очима;

2) алгоритми – внутрішня будова сервісу, що прихована від очей користувачів; блоки методів по обробці запитів;

3) збереження – методи по збереженню інформації на сервері;

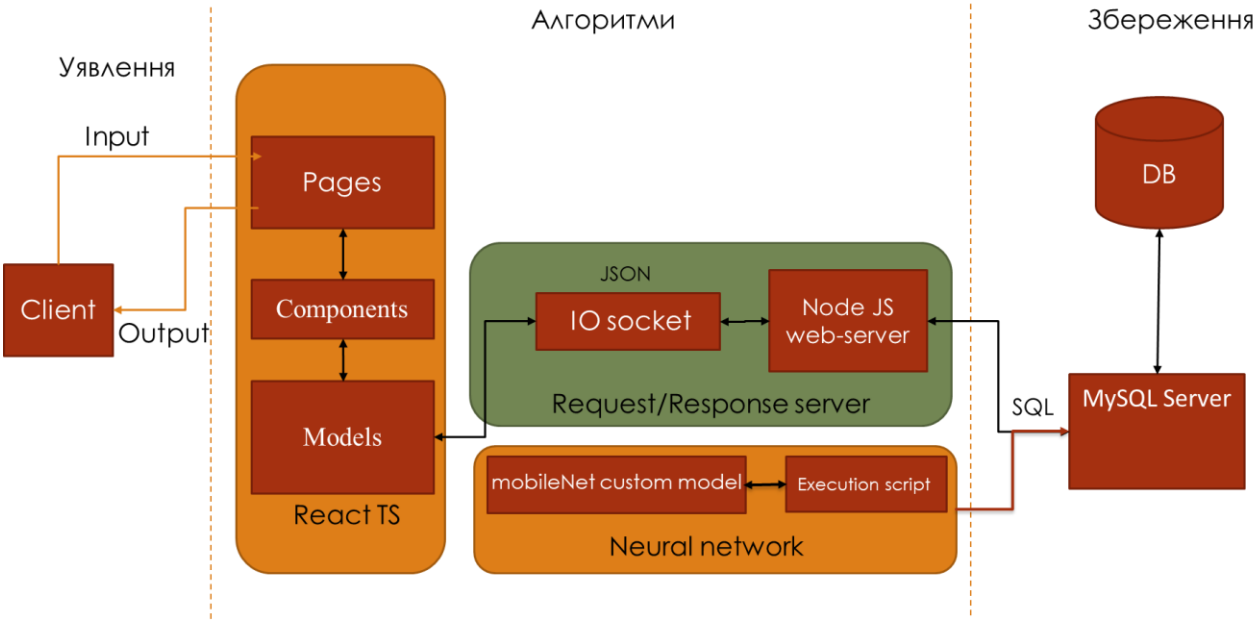

Рисунок 2.1 – Архітектура системи

Нижче приведений розбір архітектури, вказаної на рисунку. Особливості методів та обраних рішень приведені в кінці опису кожної із частин.

Front-end частина системи складається з таких модулів:

1) Pages – сторінки сайту, які бачить користувач. Це сукупність компонентів, які використовуються повторно. За рахунок цього сторінки мають високу продуктивність та гнучкість.

2) Components – складові частини, що в своїй сукупності складають сторінку. Компоненти дозволяють розділяти інтерфейс на самостійні повторно застосовані частини і аналізувати кожну з них окремо. За ідеєю, компоненти це функції JavaScript. Вони приймають довільну властивість і віддають назад React елементи, що описують те, що повинно відображатися на екрані.

3) Models – моделі даних, що застосовуються у роботі компонентів при прийомі та передачі даних.

Усі модулі Front-end частини розроблені з використанням технології React TS. Завдяки цьому реалізується:

1) Безпечність – кожний компонент, що написаний на TypeScript компілюється у html з використанням унікальних ідентифікаторів, тим самим приховуючи введені користувачем дані та зв’язки з сервером;

2) Надійність – у випадку помилок зі сторони серверу, клієнт містить спеціальні методи по обробці помилок з подальшим повідомленням користувача та спробою повторного підключення;

3) Оптимізація – клієнт компілюється за спеціальними алгоритмами, що мінімізують html-код і на виході повністю скомпільована html-сторінка не перевищує об’єм у 2 МБ (без урахування контенту, отриманого з БД).

Back-end частина системи складається з наступних модулів:

1) Request/Response Server – серверна частина проекту, що виконує запити клієнта. Виконана на основі Node.JS. Включає до себе:

а *Input-Output socket* – комплекс методів, реалізованих на основі Веб-сокетів. Організовують двосторонню, незалежну передачу даних між клієнтами та сервером. Головною перевагою даного рішення є повна незалежність клієнтів одне від одного. Створені сокети працюють паралельно та контактують лише під час прямого підключення на сервері. В усі інші випадки сервер не оброблює запити, що добре впливає на оптимізацію [21];

b *Node JS server* – сервер, що займає виконуючу роль для взаємозв'зку клієнту з даними БД, передачі та захисту інформації та обробці запитів чату. Не зберігає у собі даних користувачів, а лише організовує зв'язок між різними сокетами та сокетом і БД.

2) Neural network – нейронна мережа, написана на JavaScript. Основою для нейронної мережі виступає фреймворк TensorFlowJS, що включає до себе основні методи по створенню, налаштуванню та компіляції моделей штучних нейронних мереж. Включає до себе:

a *Execution script* – скрипт контролю роботи моделі. Виконує запуск аналізу, навчання моделі, відправку результатів до БД. Працює незалежно від клієнту та серверів, запуск виконується за таймером, або вручну;

b *MoileNet custom model* – алгоритм опису моделі нейронної мережі, автором якої є компанія Google. Особливістю даного алгоритму є добра оптимізація, маленькі розміри самої моделі та добра оптимізація. Завдяки цьому дану модель здатні обробляти навіть смартфони. Модель, список об'єктів та ваги нейронних зв'язків зберігаються у різних файлах, що підвищує читаємість.

3) MySQL Server – сервер БД. Виконує запити, отримані від серверів та нейронної мережі. Запити, отримані від інших джерел, ігноруються.

4) SQL DB – сама база даних.

2.2 Вимоги до модулів

Для стабільної роботи веб-сервісу, потрібне постійне підключення до сервісу керування базами даних, тому інтернет з'єднання повинно бути швидким та стабільним.

Веб-сервер повинен підтримувати підключення до бази даних, та контролювати помилки у системі, щоб зменшити ризик критичної помилки та втрати даних користувача.

Основні три модулі клієнтської частини тісно взаємозв’язані. Якщо виникне помилка в одному з них, перестануть працювати два інші. Саме тому всі модулі повинні вміти реагувати на помилки будь-якого характеру, відповідно до функціоналу, та протестовані на працездатність.

2.3 Опис нейронної мережі

У проекті використовується метод автоматичної перевірки контенту, що завантажується, за допомогою штучної нейронної мережі. Оскільки постів буде викладатись багато, одна людина фізично не зможе перевіряти усі нові пости. Рішення – використання методики машинного навчання для автоматичної перевірки інформації.

В сучасних умовах класифікації та розпізнавання образів активно використовуються штучні нейронні мережі – математична модель, що побудована за принципом біологічної нейронної мережі. Існує багато методів реалізації штучних мереж – від багатошарових перцептронів до рекурентних та згорточних мереж.

Штучні нейронні мережі не програмуються у звичному плані, вони навчаються. Задача розробника – розробити модель та навчити виконувати поставлені задачі, показуючи очікувані результати (метод навчання з учителем).

За основу була обрана нейронна мережа типу згорточної нейронної мережі (рис. 2.2), оскільки даний вид мережі має просту будову, гнучке налаштування та гарну швидкодію. Особливістю таких мереж є розподілення нейронів на, так звані, шари, за аналогією до перцептрону Розенблатта. Кожен нейрон поточного шару пов’язаний з усіма нейронами попереднього шару, але в самому шарі нейрони не пов’язані. Єдина відмінність від перцептрону – особливий вхідний шар, що уявляє

собою матрицю згортки. Фотографія розбивається на три канали за кольорами: червоний, синій та зелений. Після чого виконується згортка – стискання групи пікселів (5x5, 2x2 тощо) до одного єдиного пікселю, тим самим, стискаючи фотографію до менших розмірів. В результаті отримуємо матрицю, що придатна до аналізу штучними нейронами.

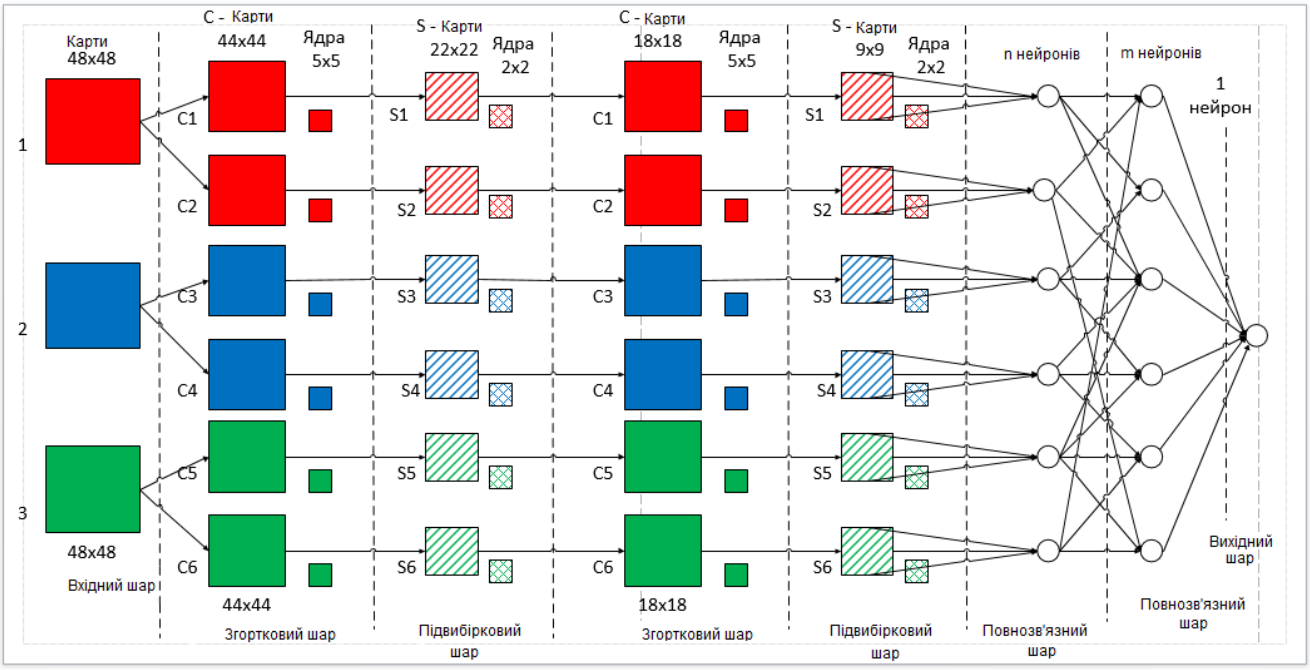

Рисунок 2.2 – Модель згорточної нейронної мережі

Завдання для мережі:

1) Мережа повинна читати фотографії, що завантажують користувачі та аналізувати об’єкти, що розміщені там. В залежності від результату, фотографії копіюються до навчальної вибірки за критеріями, для послідуючого навчання моделі;

2) Модель повинна розпізнавати образи на фотографіях, що відповідають параметрам навчальної вибірки. Навчальна вибірка – збірник стокових фотографій, розподілених по папкам по спільному критерію, наприклад, папка «Театр» та перелік фотографій театру;

3) Через деякий проміжок перевірок модель нейронної мережі проходить перенавчання, використовуючи як навчальну вибірку збірник фотографій, поповнений новими, відсортованими фотографіями користувачів;

4) Результат роботи нейронної мережі – визначення недопустимих образів на фотографіях (сцени насилля, порнографія, заборонена символіка тощо) та блокування постів, до яких ці фотографії належать;

Особливістю такого підходу до навчання мережі є відсутність необхідності редагувати саму модель та код її опису під час кожного повтору циклу навчання. Мережа буде видавати тим точніший результат, чим більше та якісніше буде навчальна вибірка.

Параметри мережі:

1) 1 вхід, що приймає спеціальну матрицю – тенсор (оброблену картинку за спеціальними алгоритмами);

2) Внутрішні шари – кількість шарів, нейронів та функції активації визначаються емпіричним методом;

3) Вихід – кількість виходів визначається кількістю образів, що здатна розпізнати мережа, функція активації «SoftMax» (особливістю цієї функції є посилення вхідного сигналу з минулого шару нейронів для підвищення чутливості штучного нейрону);

Виконання розпізнавання образів виконується в автоматичному режимі, незалежно від роботи серверів.

3 СПЕЦИФІКАЦІЯ ВИМОГ

3.1 Підстави для розробки

В рамках виконання дипломного проекту було вивчено вже існуючі ринкові пропозиції в сфері веб-сервісів путівників та соціальних мереж з функцією гео-локації, а також розроблено програмний продукт, що поєднує у собі одразу путівник та соціальну мережу [21].

Веб-сервіс буде створений на базі інформації, що будуть додавати самі користувачі, що дозволить краще вивчити міста, а елементи соціальної мережі дозволять шукати людей зі схожими інтересами для спільних прогулянок. Тим самим знижуючи соціальний поріг пошуку нових знайомств.

Інтегрована нейронна мережа буде виконувати перевірку контенту, що користувачі додають до мережі, що дозволить прискорити поповнення бази з дотриманням правил використання веб-сервісу.

Розроблюваний сервіс міститиме кращі функції сайтів-аналогів, а також додатковий функціонал, що значно покращить знання про місцевість та зручність комунікації між туристами чи звичними жителями, що шукають собі компанію для відпочинку.

Отже, необхідно створити Веб-сервіс «Eternal Radiance», який здатний об’єднати в собі елементи соціальної мережі та цифрового довідника, та призначений для полегшення вивчення місцевості на основі досвіду місцевих жителів. Продукт повинен бути розроблений за допомогою мов програмування MySQL та TypeScript.

Програмний продукт можна уявно поділити на декілька підсистем, що реалізовані наступним чином:

1) Front-end технології – розроблена на мові TypeScript у середовищі програмування React.JS. В якості патерну розробки візуальної складової була використана методологія BEM [1] для підвищення розширюваності та універсальності коду.

2) Back-end технології – поділені на дві підсистеми: Node.JS-сервер, що що виконує основні запити, реалізовує чат реального часу та другий Node.JS-сервер, що є допоміжним та виконує роботу з нейронною мережею.

Також для проектування було обрано архітектурний патерн Model-View-Controller. Це спосіб організації коду, який передбачає виділення блоків, що відповідають за вирішення різних завдань. MVC - підхід до проектування додатки, який передбачає виділення коду в блоки модель, уявлення і контролер. Контролер обробляє вхідні запити. Модель дістає з бази даних інформацію, потрібну для виконання конкретних запитів. Подання визначає результат запиту, який отримує користувач.

Основні варіанти використання:

1. Створення посту – користувач має право створювати та публікувати блоки інформації, що пов’язані з конкретною географічною областю. Кількість постів, що може публікувати користувач розраховується в залежності від його рейтингу – особистої оцінки довіри в середині системи, що встановлюється іншими користувачами.

2. Перегляд постів – користувач має право переглядати публічні пости інших користувачів для отримання нових знань про місцевість, що цікавить. Кожний пост можна оцінювати на якість поданої інформації, відповідність до реальної місцевості. Також доступна система коментарів.

3. Соціальна взаємодія – користувачі мають можливість підписуватися одне на одного, створювати групові та приватні чати, обмінюватись повідомленнями та обмежувати доступ від небажаних користувачів.

3.2 Призначення розробки

Дана система буде використовуватись як соціальна мережа загального доступу. Кожна людина, що зацікавлена у вивченні місцевості: туристи, люди у відрядженнях, зможуть завчасно дізнатися про місце, куди прямують. «Eternal

Radiance» має стати цікавим та корисним довідником для зацікавлених людей, що допоможе знайти на мапі саме те місце, що подобається користувачу.

Туристам та робітникам у відрядженнях соціальна мережа допоможе знайти нові місця для відвідування, компанію за інтересом та отримати відносну довідку про безпечність конкретного району міста.

Система повинна експлуатуватися у вільному доступі на комп’ютері та, в подальшому, на смартфонах.

3.3 Вимоги до програмного продукту

Вимоги до функціональних характеристик

Функціональні вимоги регламентують функціонування або поведінку системи (behavioral requirements). функціональні вимоги відповідають на питання "Що повинна робити система" в тих чи інших ситуаціях.

Функціональні вимоги визначають основний "фронт робіт" розробника, і встановлюють цілі, завдання та сервіси, що надаються системою Замовнику.

Функціональні вимоги записуються за коштами розпорядчих правил:

- "система повинна дозволяти викладачеві тестові питання і відповіді";

- "сервіс повинен забезпечувати можливість перегляду зведеної інформації за результатами тестування ".

Іншим способом є варіанти використання (usecase). Usecase – варіант використання, прецедент, абстракція в області створення вимог до ПО. Прецеденти застосовуються для вилучення вимог до системи, тобто, того, що система передбачає робити. Необхідну поведінку системи задається одним або декількома варіантами використання, які визначаються відповідно до потреб акторів. Термін "варіант використання" відноситься до типу прецеденту. Примірник прецеденту відноситься до прояву поводження, що відповідає типу прецеденту. На рисунку 3.1

представлена Use-case діаграма, яка в загальному вигляді відображає функціональні вимоги до системи "Eternal Radiance".

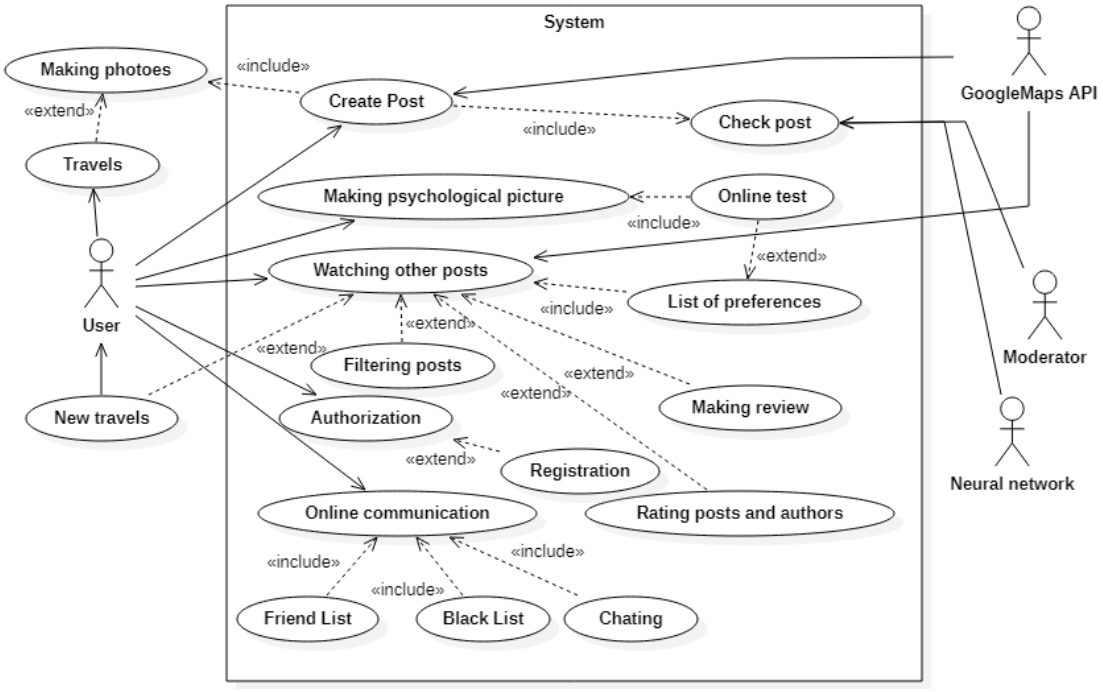

Рисунок 3.1 – Варіанти використання системи

Надалі приведені сценарії використання. Були описані саме ті, що обробляються системою:

1 Create post (Створення поста):

Область дії: система;

Учасники та інтереси: Користувач хоче створити новий пост;

Передумова: Користувач зареєстрований у системі;

Гарантія успіху: Користувач отримав повідомлення «Пост успішно додано до системи»;

Тригер: Користувач натискає кнопку «Створити пост»;

Основний сценарій:

1.1 Користувач відсилає запит на створення поста;

1.2 Система перевіряє рейтинг користувача;

1.3 Система запитує координати вказуємого місця. Користувач обирає місце на карті. Система підтверджує;

1.4 Система запитує опис місця, категорію та фото. Користувач вводить. Система підтверджує;

1.5 Система демонструє попередній вигляд поста.Користувач підтверджує. Система зберігає пост.

Розширення:

1.2а Рейтинг користувача менше 20%:

1.2а1 Система повідомляє, що користувач має низький рейтинг;

1.2а2 Кінець сценарію.

1.4а Користувач ввів не всі дані:

1.4а1 Система повідомляє, що даних не вистачає. Перехід до П.1.4.

2 Watching other posts:

Область дії: система;

Учасники та інтереси: Користувач хоче переглянути пости інших користувачів;

Передумова: Користувач зареєстрований у системі;

Гарантія успіху: Користувач побачив стрічку постів;

Тригер: Користувач переходить на головну сторінку;

Основний сценарій:

2.1 Система показує список-стрічку з переліком постів інших користувачів;

2.2 Користувач обирає пост із стрічки. Система виводить детальну інформацію про місце та вказує його на карті.

3 Chating

Область дії: система;

Учасники та інтереси: Користувач хоче виконати фільтрацію стрічки постів;

Передумова: Користувачі 1 та 2 зареєстровані в системі, рейтинг обох більше ніж 25;

Гарантія успіху: Стрічка постів успішно відсортована;

Тригер: Користувач вибрав дію «Чат» у списку взаємодій;

Основний сценарій:

3.1 Система завантажує сторінку чату, з'єднує користувачів і очікує подальших дій;

3.2 Користувач вводить повідомлення. Система підтверджує і відправляє повідомлення Користувачу2;

3.3 Користувач2 отримує сповіщення і переходить до чату. Система підтверджує, що повідомлення доставлено;

3.4 Користувач2 вводить відповідь. Система підтверджує і відправляє повідомлення Користувачу1;

3.5 Перехід до п.3.2 поки один з користувачів не закриє чат;

3.6 Система зберігає листування і завершує з'єднання.

4 Searching new posts by filters:

Область дії: система;

Учасники та інтереси: Користувач хоче виконати фільтрацію стрічки постів;

Передумова: Користувач зареєстрований у системі і знаходиться на головній сторінці;

Гарантія успіху: Стрічка постів успішно відсортована;

Тригер: Користувач натискає кнопку «Фільтр»;

Основний сценарій:

4.1 Користувач викликає меню фільтра. Система виводить;

4.2 Користувач вибирає критерії фільтрації та підтверджує. Система підтверджує;

4.3 Система виводить відфільтрований список постів.

Розширення:

4.3а Результатів пошуку за вказаними фільтрами не знайдено:

4.3.а1 Система сповіщає користувача, що результатів не знайдено;

4.3.а2 Система пропонує створити пост за вказаними критеріями;

4.3.а2а Користувач погоджується:

3.3.а2.а1 Перехід до сценарію 1.

4.3.а2б Користувач не погоджується:

4.3.а2.б1 Кінець сценарію.

5 Rating posts and authors:

Область дії: система;

Учасники та інтереси: Користувач хоче оцінити пост іншого користувача;

Передумова: Користувач зареєстрований у системі і переглядає пост;

Гарантія успіху: Користувач побачив повідомлення «Успішно оцінено»;

Тригер: Користувач натиснув на оцінку напроти заголовку поста;

Основний сценарій:

5.1 Користувач викликає форму оцінки. Система підтверджує;

5.2 Система виводить форму та запитує числову оцінку та опис. Користувач вводить. Система підтверджує;

5.3 Якщо числова оцінка < 40, то коментар розцінюється, як жалоба;

5.4 Користувач підтверджує відгук. Система зберігає;

Розширення:

5.3а Оцінка > 40, але < 70:

5.3.а1 Коментар розцінюється системою, як нейтральний.

5.3б Оцінка > 70:

5.3.б1 Коментар розцінюється, як позитивний.

6 Check post:

Область дії: система;

Учасники та інтереси: Модератор бажає перевірити пост;

Передумова: Модератор авторизований у системі;

Гарантія успіху: Модератор успішно перевірив пост;

Тригер: Модератор натиснув на кнопку «Перевірити» напроти заголовку поста;

Основний сценарій:

6.1 Модератор обирає пост на оцінку. Система підтверджує та виводить детальну інформацію;

6.2 Модератор оцінює пост за заданими критеріями;

6.3 Якщо пост схвалений, то він стає видимим для всіх користувачів. Інакше – пост помічається, як «не схвалено» та після 2-х днів автоматично видаляється;

6.4 Якщо в пост з поміткою «не схвалено» були внесені зміни від автора, то помітка знімається і пост повертається у чергу на перевірку. Перехід до П6.1.

7 Registration

Область дії: система;

Учасники та інтереси: новий Користувач бажає зареєструватись;

Передумова: Користувач не зареєстрований у системі;

Гарантія успіху: Система видала повідомлення «Успішно зареєстровано»;

Тригер: Користувач натиснув на кнопку «Реєстрація»;

Основний сценарій:

7.1 Користувач викликає форму реєстрації. Система підтверджує та виводить;

7.2 Система запитує дані реєстрації. Користувач вводить. Система підтверджує;

7.3 Система пропонує пройти ряд психологічних тестів. Користувач згоден. Перехід до сценарію №8.

Розширення:

7.2а Дані неповні/некоректні:

7.2.а1 Система видає повідомлення про помилку при вводі даних. Перехід до П7.2.

7.2б. Дані вже існують у системі:

7.2.б1. Система відає повідомлення «Користувач вже існує» і пропонує авторизуватися.

7.3а. Користувач не згоден:

7.3.а1. Система зберігає дані користувача без тестування. Кінець сценарію.

8 Creating psychological picture

Область дії: система;

Учасники та інтереси: новий Користувач бажає пройти тестування;

Передумова: Користувач успішно зареєстрований у системі;

Гарантія успіху: Система видала повідомлення «Успішно зареєстровано»;

Тригер: Користувач натиснув на кнопку «Пройти тестування»;

Основний сценарій:

8.1 Система відображає сторінку, що містить ряд психологічних тестів: на знаходження темпераменту, акцентуацію особистості та особистих переваг;

8.2 Користувач проходить усі тести. Якщо який-небудь із тестів не завершений, то Система сповістить про помилку та вкаже на незаповнені поля;

8.3 Система зберігає результат та враховує його при відображенні рекомендованих місць на карті.

9 Authorization

Область дії: система;

Учасники та інтереси: Користувач бажає скористуватися сервісом;

Передумова: Користувач успішно зареєстрований у системі;

Гарантія успіху: Система видала повідомлення «Авторизовано»;

Тригер: Користувач перейшов на сторінку авторизації/реєстрації;

Основний сценарій:

9.1 Система відображає форму авторизації та реєстрації;

9.2 Користувач вводить логін та пароль у форму авторизації. Система перевіряє;

9.3 Якщо користувач не зареєстрований, то перехід до сценарію №7. Інакше – перехід до П.9.4;

9.4 Система підтверджує користувача та відкриває сторінку його профіля.

Вимоги до надійності

Вимоги до надійності (або нефункціональні вимоги) – це вимоги до програмного забезпечення, які задають критерії для оцінки якості його роботи. На відміну від функціональних вимог, які визначають що система повинна робити, нефункціональні вимоги визначають якою система повинна бути. Даний підрозділ містить описання наступних видів нефункціональних вимог:

- Вимоги до інтерфейсу

1. Апаратні інтерфейси (Hardware Interfaces) – апаратні інтерфейси необхідні для підтримки системи, включаючи логічну структуру, фізичні адреси і очікувану поведінку;

2. Інтерфейси ПЗ (Software Interfaces) – інтерфейси програмного забезпечення з якими аплікація повинна взаємодіяти;

3. Комунікаційні інтерфейси (Communications Interfaces) – інтерфейси для комунікацій (взаємодії) з іншими системами та/або пристроями;

4. Інтерфейс користувача (User Interface) – засіб зручної взаємодії користувача з інформаційною системою;

- Апаратні та програмні вимоги (Hardware/Software Requirements) – опис апаратної та програмної платформ, необхідних для роботи (і підтримки) системи.
- Операційні вимоги (Operational Requirements):

1. Безпека та конфіденційність (Security and Privacy);
2. Надійність (Reliability);
3. Відновлювальність (Recoverability);

4. Продуктивність (Performance);
5. Потенціал (Capacity);
6. Збереження даних (DataRetention);
7. Керування помилками (ErrorHandling);
8. Правила перевірки (ValidationRules);
9. Узгоджені стандарти (ConventionStandards).

- Обмеження (Restrictions) – умови, що обмежують вибір можливих рішень по реалізації окремих вимог або їх наборів. Вони істотно обмежують вибір засобів, інструментів і стратегій при розробці зовнішнього вигляду і структури (в т.ч. архітектури) продукту або системи.

Апаратні та програмні вимоги (Hardware/Software Requirements):

1 Система представляє собою веб-сервіс, тому потребує стабільного підключення до інтернету, наявності веб-браузера Google Chrome, Opera, Firefox, Safari, чи іншого;

2 Мінімальна роздільна здатність екрану: для комп’ютеру – 800x700, для смартфону – 640x320.

Обмеження (Restrictions):

1 Розробка системи повинна вестися на кількох платформах. Front-end частина за допомогою ReactJS, Back-end частина – за допомогою VS Code;

2 Система повинна навчатися автоматичній перевірці постів за допомогою методів Машинного Навчання;

3 Користувачі, що порушили правила використання системи будуть заблоковані;

4 Результати тестування користувачів повинні залишатися анонімними. Особиста інформація – за бажанням самого користувача може стати публічною.

Операційні вимоги (Operational Requirements);

1 Надійність:

1.1 Якщо користувач створює пост, то система показує повідомлення о результаті менш ніж за 4 секунди;

1.2 У випадку критичної помилки система зберігає роботоспроможність у 80% випадків;

1.3 Якщо система запитує у БД дані по фільтрації, то ймовірність отримати коректний результат не менше, ніж 0.8;

1.4 Якщо система зберігає дані в БД, то ймовірність вдалого збереження не менше 0.9.

2 Продуктивність:

2.1 Якщо користувач запитує у системи дані про іншого користувача, то система відповідає на запит менше ніж за 0.5 секунд;

2.2 Якщо користувач запитує дані про гео-помітку, то система відповідає на запит менше ніж за 1 секунду;

2.3 Якщо користувач фільтрує дані, то система оброблює 1500 постів за секунду;

2.4 Якщо користувач запитує рейтинг другого користувача, то отримує відповідь менше ніж за 0.5 секунд.

3 Безпека:

3.1 Якщо система запитує інформацію з БД, то отримує її в цілісності не менш, ніж у 94% випадків;

3.2 Якщо користувач створює фейкові пости, чи відмічує місця, небезпечні для життя інших користувачів, то система відсіює їх більш ніж у 70% випадків;

3.3 Якщо зміст поста, чи безпечність відміченого місця стали під сумнівом, то в 90% випадків вони відправляються на ручну модерацію;

3.4 Якщо користувач реєструється, то його дані будуть захищені в 98% випадків.

4 Usability:

4.1 Якщо користувач створює пост, то йому це вдається не більше ніж за 40 секунд;

4.2 Якщо користувач бажає оцінити іншого користувача, то йому це вдається не більше, ніж за 15 секунд;

4.3 Якщо користувач бажає відфільтрувати список постів, то йому це вдається менше, ніж за 10 секунд;

4.4 Якщо модератор бажає перевірити пост, то в нього це виходить менш, ніж за 40 секунд.

5 Супроводження:

5.1 Якщо компанія бажає змінити функціонал системи, то зміни вступають у силу менш ніж за 2 місяці;

5.2 Якщо компанія бажає відновити робоче обладнання, то система почне працювати з ним не більш, чим за місяць;

5.3 Якщо компанія бажає портувати систему на інший гаджет, то поставлена задача буде виконана не більш, ніж за 1 рік;

5.4 Якщо компанія відмічає велику кількість відгуків про помилку системи, то виправлення вступлять у силу не більш, чим через 1 місяць.

Умови експлуатації

Дана система не потребує особливих умов експлуатації. Користувач має можливість використовувати соціальну мережу за будь-яких зручних для нього

умов. Для справної роботи ЕОМ, або смартфону температура навколишнього середовища не повинна перевищувати 50℃, та бути нижчою за 0℃.

Вимоги до складу і параметрів технічних засобів

Системні вимоги мають відповідати нормам, що вказані у табл. 3.1 та 3.2:

Таблиця 3.1 – Системні вимоги користувача

Вимога	Мінімальне значення	Рекомендоване значення
Платформа	Android 6.0 Windows 7 Linux	Android 8.0 Windows 10 Linux Ubuntu
Оперативна пам'ять (PC)	4 ГБ	8 ГБ та більше
Оперативна пам'ять (Mobile)	2 ГБ	6 ГБ та більше
Вільний простір на носії пам’яті	300 Мб	512 Мб і більше
Процесор	Snapdragon 535 Pentium 3	Snapdragon 685 та вище Core i5 і більше
Контролер (PC)	Клавіатура, миша	Клавіатура, миша
Розрядність	64	
Відеоадаптер (PC)	3D адаптер n-Vidia	Intel, AMD/ATI

Таблиця 3.2 – Системні вимоги серверу

Вимога	Мінімальне значення	Рекомендоване значення
Платформа	Linux Ubuntu	
Оперативна пам'ять (PC)	8 ГБ	16 ГБ та більше
Вільний простір на носії пам’яті	512 ГБ	2048 ГБ та більше
Процесор	Core I3	2x Xeon x3430
Контролер	Клавіатура, миша	Клавіатура, миша
Розрядність	64	
Відеоадаптер	-	3D адаптер n-Vidia

3.4 Вимоги до програмної документації

Попередній склад програмної документації:

1) Архітектура веб-сервісу;
2) Діаграма програмних класів;
3) Діаграма взаємодії;
4) Структура бази даних;
5) Опис інтерфейсу користувача;
6) Опис нейронної мережі;
7) Керівництво користувача;
8) Тестування основних функцій веб-сервісу та нейронної мережі;
9) Додатки – мають містити фрагменти коду, що має особливу важливість для роботи програми або представляють технічну реалізацію окремих модулів, що описані в описі інтерфейсу.

4 ПРОЕКТУВАННЯ ВЕБ-СЕРВІСУ

4.1 Опис програми

Проведений візуальний аналіз сервісів-аналогів та побудований прототип дизайну додатку. Визначена модель системи, зв’язок елементів та технології, що будуть реалізовувати функціонал.

Проектування системи виконане з деякими поправками відносно моделі системи для підтримки виключних випадків, що не були розглянуті на етапі аналізу.

Програмна реалізація системи виконана до етапу базового функціонування – система виконує базові запити та поставлені задачі та виконує аналіз вхідних від користувачів даних.

Стек технологій, що використаний для розробки, наступний:

1) Front-end – React, Redux, JavaScript, HTML, CSS
2) Back-end – Node.JS, Express, Socket.IO, JavaScript
3) DB – MySQL

Зв’язок клієнту з сервером реалізований завдяки закритому, шифрованому тунелю з використанням технології веб-сокетів. Це дозволяє ізолювати трафік даних користувача від інших користувачів системи та обмінюватися даними у реальному часі.

Загальна кількість необхідних серверів – 2:

1) Node.JS для виконання основного функціоналу та сценаріїв роботи нейронної мережі
2) MySQL server – незалежна база даних

Опис логічної структури

Виходячи з предметної області, система розроблюється з урахуванням того, що користувач реєструється у системі та потім проходить авторизацію, для

отримання доступу до повного функціоналу веб-сервісу. Авторизований користувач отримує можливість переглядати стрічку постів інших користувачів, взаємодіяти з користувачами системи завдяки системі коментарів, чатів реального часу, інших частин складової соціальної мережі.

Кожному користувачу присвоюється початковий рейтинг, який змінюється в залежності від активності користувача – цікава, актуальна інформація створених постів підвищує рейтинг, а нецікава, небезпечна чи нецензурна – сильно знижують. Від стану рейтингу залежить доступний функціонал системи. Високий рейтинг відкриває доступ до повного функціоналу сервісу. При зниженні рейтингу до певних значень спочатку блокується можливість створювати пости, далі – можливість коментації та оцінювання постів інших користувачів, під кінець – блокується можливість використання чатів. Рейтинг користувача встановлюють інші користувачі системи, виставляючи оцінку контенту, що публікує наш користувач.

Кожний користувач вказує інформацію щодо персональних даних (ПІБ, дата народження, стать, фото), особисті переваги щодо підбору системою рекомендованих для візиту місць. Завдяки рейтинговій системі інші користувачі мають можливість уникати небажаних користувачів, або людей, що порушують основні правила роботи з системою (правила користування, створення поміток-пустушок, або розміщення потенційно небезпечної інформації).

Кожний пост, створений користувачем, включає до себе: координати місця, що відображають мітку на карті та пов’язані з нею, фотографії місця, короткий відгук, а також області для коментарів та оцінки посту. Також пости можуть бути публічними та приватними. Приватні пости не проходять перевірку та доступні лише автору. Приватні пости необхідні у тому разі, якщо користувачу необхідно зберегти якесь місце для швидкого доступу до нього, на випадок регулярного відвідування.

Використовувані технічні засоби

а) Вимоги до програмного забезпечення back-end частини: підтримка Node.js 3.0 або вище; підтримка бази даних MySQL.

б) Вимоги до програмного забезпечення клієнтської частини: браузери: Google Chrome, Mozilla, Opera, Safari із ввімкненою опцією JavaScript та Геолокації;

в) Вимоги до швидкості інтернету: мінімальна швидкість 2 Мбіт/сек; рекомендована швидкість 15 Мбіт/сек.

г) Апаратні обмеження (мінімальні): розширення екрану 800*600; процесор з частотою >= 1,3 Ггц; оперативна пам’ять >= 2 Гбайт.

Виклик і завантаження

Для отримання доступу до функціоналу веб-застосування потрібно зареєструватися і підтвердити свою особистість через почтову скриньку. Користувач безпечно вводить свої дані, та зберігає їх, щоб при повторному вході не вводити заново. Для отримання доступу до спеціальної, більш швидкої геолокації, потрібно заключити договір щонайменше на 1 місяць.

Вхідні та вихідні дані

Вхідними даними програмного продукту інтерфейс, за допомогою якого користувач взаємодіє із системою, створює та редагує профіль, створює та читає публікації, відвідує профілі інших користувачів, обмінюється повідомлення, шукає пости за геолокацією. Також вхідними даними можна вважати ті дані, які користувач вводить під час створення публікацій. Отже, вихідними даними можуть бути створені публікації та коментарі, а також інтерфейс.

4.2 Структура та опис бази даних

Спираючись на аналіз ER-діаграми з послідуючим виділенням сутностей, була побудована БД за відповідною архітектурою. На рис.4.1 наведена візуалізація створених таблиць з урахуванням зв’язків між ними.

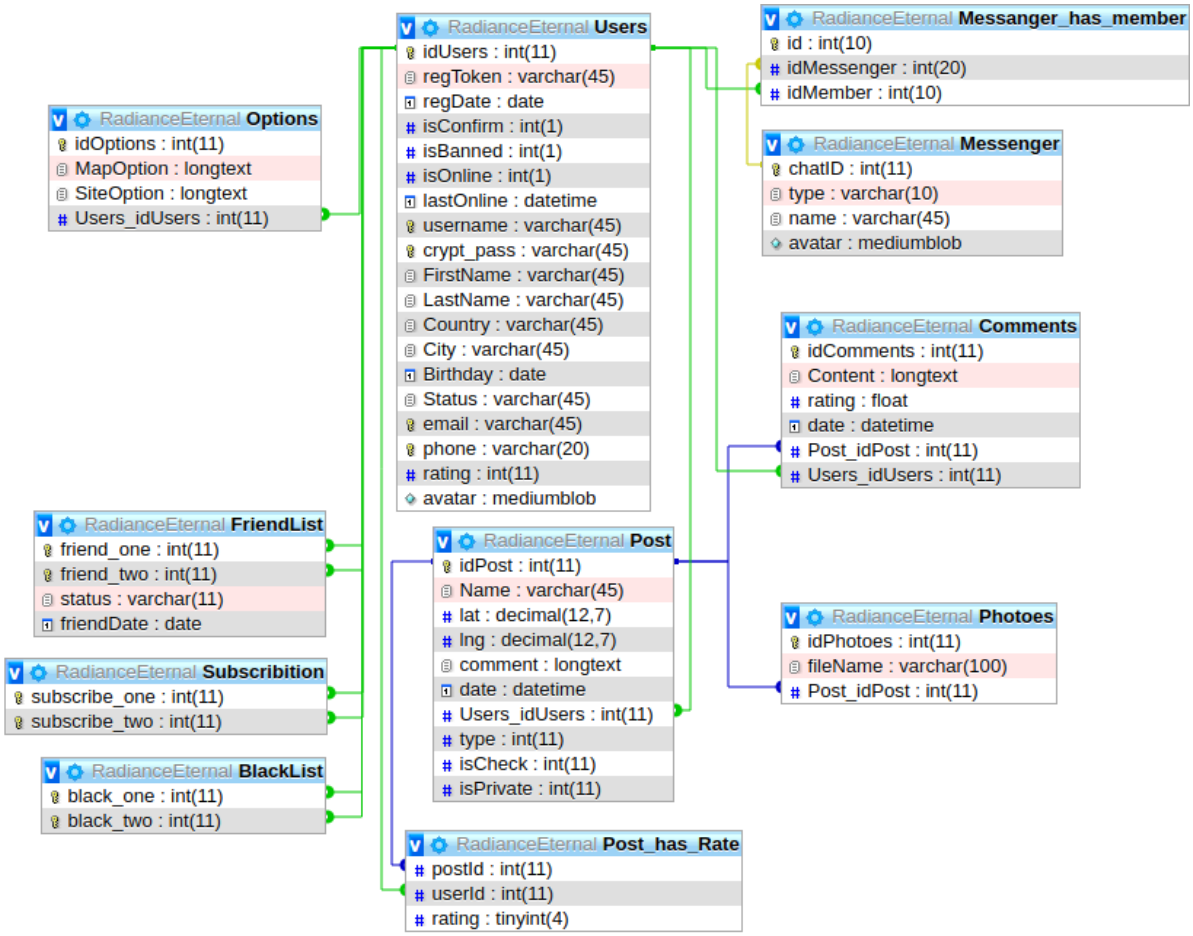

Рисунок 4.1 – Структура бази даних системи

Таблиця Users містить дані про користувачів, що були вказані підчас реєстрації. Варто відмітити поле regToken – це унікальний токен, що генерується під час створення користувача та під час процедури відновлення доступу до профілю. Необхідний для точної ідентифікації пошти користувача. Для зв’зку таблиці з іншими таблицями використовується ID користувача – унікальний ідентифікатор, що прихований від очей людини. Для тестування та відладки ID

користувача представлений у вигляді номеру по порядку, надалі – буде замінений на символьний шифр.

Таблиця Post містить дані про пости користувачів. Ідентифікація автора виконується через ID користувача.

Таблиця Comments містить перелік коментарів для кожного посту від кожного користувача.

Користувач має можливість створювати безліч постів, але один пост може бути створений лише одним користувачем. Також, користувач має можливість створювати безліч коментарів для безлічі постів.

Таблиця Photoes містить контактні дані про фотографії, що були завантажені користувачами – назву файлу, ID посту, до якого прикріплено.

Таблиця Options містить користувальницькі налаштування веб-сервісу – палітра кольорів елементів сайту та карти, що до вподоби окремому користувачу.

Таблиця Messenger містить контактні дані про чати користувачів – ID чату, тип, назву, аватар.

Таблиця Messenger_has_member містить перелік користувачів, що належать до чатів.

Таблиця FriendList містить зв’язки друзів – пости друзів будуть мати вищий пріоритет, та будуть відображатися спочатку стрічки постів, а також мають можливість створювати персональні чати.

Таблиця BlackList містить зв’зки заборонених користувачів – користувачі, що пов’язані цією таблицею втрачають можливість бачити пости, повідомлення та профілі одне одного.

Таблиця Subscribition містить зв’зки підписок -- пости підписок будуть мати середній пріоритет, та будуть відображатися у стрічці постів після постів друзів.

4.3 Проектування програмних класів

Діаграма класів – це статичне представлення структури моделі, що відображає декларативні елементи. Ця діаграма показує класи, інтерфейси та об’єкти, а також їх відносини [11].

Клас – це спеціальна конструкція, яка використовується для групування пов’язаних змінних та функцій.

На рис. 4.2 наведено діаграму програмних класів веб-сервісу.

Даний рисунок умовно поділений на дві частини: верхня половина – програмні класи Back-end складової, нижня – Front-end. Front-end частина має складну, багаторівневу ієрархію класів (більше 50) для реалізації BEM-методу розробки [1]. Кожний простий клас є однією «деталькою» більш складного класу котрий, в свою чергу, є «деталькою» ще більшого класу. У такий спосіб вдалося повторно використати більшість простих класів для побудови різних великих класів (див. Додаток А).

Далі приведений опис найбільш складних класів системи.

Клас App є зв’язуючим компонентом між Front та Back частинами проекту та коренем ієрархії класів клієнтської частини. Він зберігає в собі дані користувача, що використовуються іншими компонентами в процесі роботи. Під час роботи системи постійно відображається у вигляді контейнеру для інших класів, що зробило можливим реалізацію односторінкового проекту.

Клас Header описує параметри та методи заголовку сторінки. Містить в собі посилання на різні компоненти системи та меню користувача. Відображається завжди під час роботи із системою.

Клас MainPage представляє собою набір методів для відображення головної сторінки соціальної мережі. Є батьківським класом для компонентів FeedList, GoogleMapsAPI та виконує зв’язуючу роль між ними.

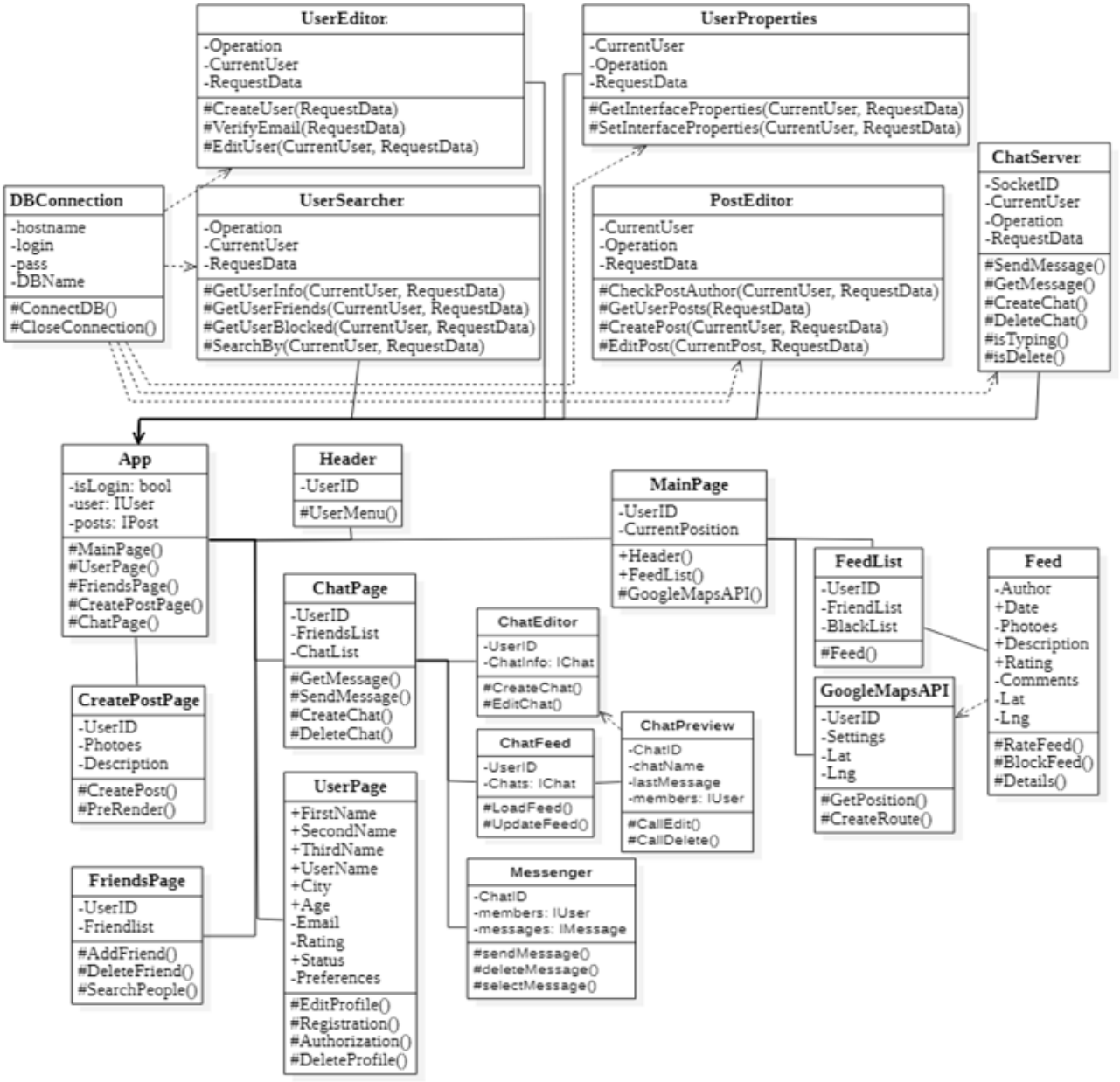

Рисунок 4.2 – Діаграма програмних класів

Клас FeedList представляє собою методи відображення та взаємодії постів користувачів. Відображається у вигляді стрічки постів у лівій частині екрану.

Клас GoogleMapsAPI виконує відображення карти та обробку методів взаємодії з картою.

Клас ChatPage – набір методів по обробці запитів для повідомлень, їх відправлення, видалення. А також – набір методів для обробки чатів, їх редагування, створення, сортування. Є батьківським компонентом для класів ChatEditor, ChatFeed, Messenger.

Клас ChatEditor – виконує обробку запитів на створення та редагування чатів, відображається на екрані у вигляді редактору.

Клас ChatFeed – виконує відображення стрічки активних чатів, у яких користувач є учасником. Пов’язаний із ChatEditor, викликаючи його екземпляри під час редагування та створення чатів.

Клас ChatPreview – відображає первинну інформацію про чат – назву, тип, останнє повідомлення. Є одиницею відображення інформації в батьківському класі ChatFeed.

Клас Messenger – виконує відображення інформації про чат, що був обраний у ChatFeed – назва, список учасників, повідомлення, що відсортовані по даті та згруповані по автору письма. Виконує запити на виділення, видалення та відправлення повідомлень.

Клас UserPage – виконує усі методи системи, що пов’язані із даними користувача. Створення профілю, редагування та перегляд. Має адаптивну поведінку – в залежності від вхідних параметрів може виконувати роль як редактора, так і переглядача. Відображається на екрані, відповідно до параметрів, як редактор, або переглядач.

Клас FriendsPage – компонент, що виконує пошук інших користувачів, що зареєстровані у системі. Відображається як стрічка із прев’ю користувачів та блоком фільтрів.

Клас CreatePostPage – компонент, що відповідає за створення та редагування постів. Містить методи по завантаженню фотографій, їх попередній обробці, введенню текстового опису та блок карти для виділення гео-помітки.

Клас UserEditor – виконує обробку запитів клієнту, що направлені на редагування інформації про користувача.

Клас UserSearcher – обробляє запити по пошуку користувачів

Клас PostEditor – обробляє запити для постів: створення, редагування, завантаження, фільтрація

Клас UserProperties – виконує запити користувача по завантаженню та збереженню налаштувань дизайну інтерфейсу.

Клас ChatServer – виконує запити користувача по обробці даних із чатів: створення та видалення чатів, відправлення та видалення повідомлень.

4.4 ER-модель розроблюваного продукту

На рис. 4.3 наведена орієнтовна ER-діаграма сервісу. На основі цієї діаграми буде виконане виділення сутностей та організація зв’язків між таблицями БД.

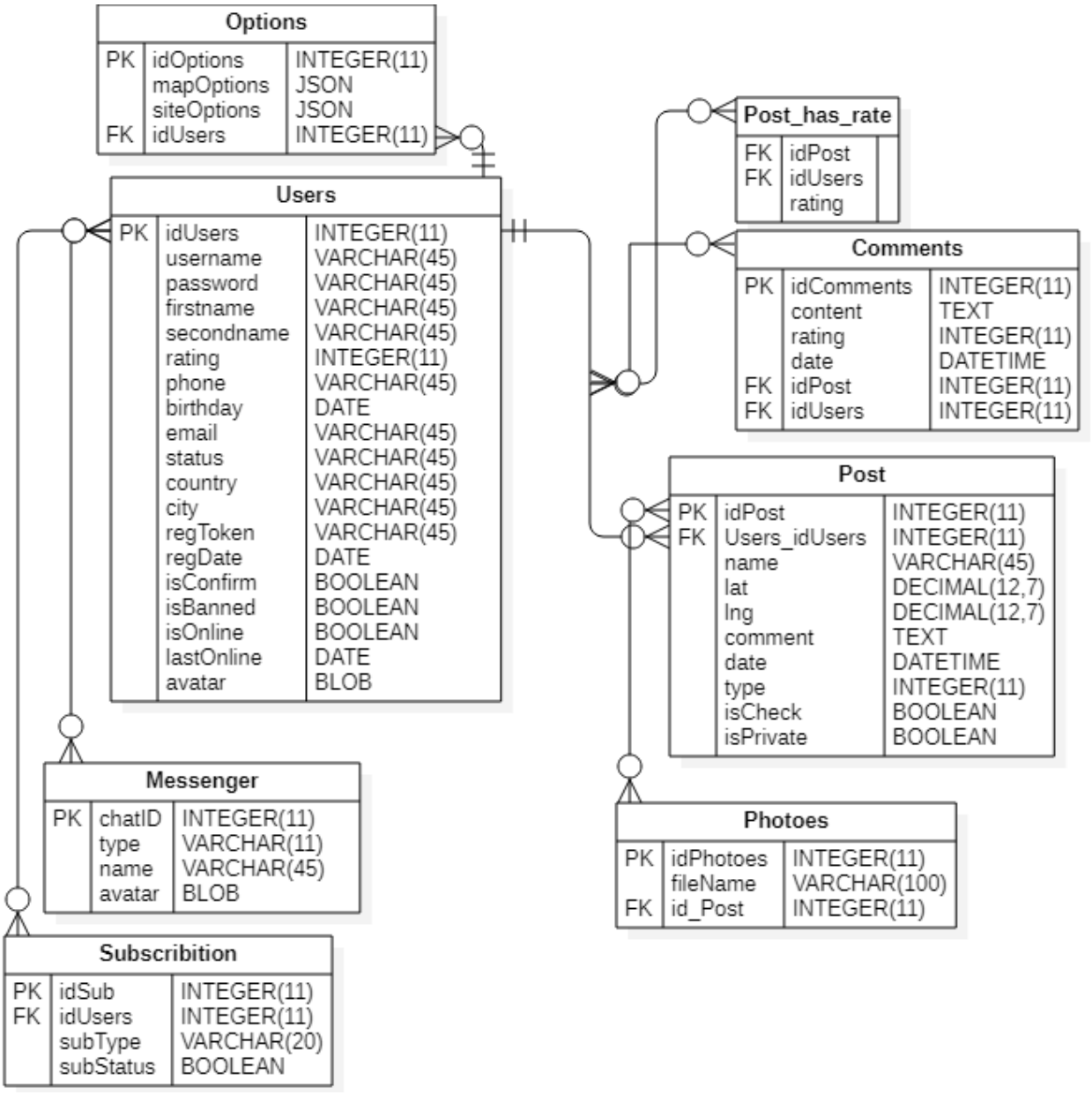

Рисунок 4.3 – ER-модель системи

На рисунку показано модулі та описано зв’язок між ними для кожної з моделей визначено атрибути, що забезпечують взаємодії між сутностями.

8 моделей даних, що фігурують у системі: Options, Users, Messenger, Subscribition, Post, Post_has_rate, Comments, Photoes.

Модель Users представляє собою набір даних користувача. У ній зберігається вся інформація, яка представлена у користувальницькому профілі для всіх інших відвідувачів сайту.

Модель Post це набір даних про публікацію, яка відображується у стрічці постів. Їх може бути велика кількість, але у всіх цих публікаціях буде одна спільна модель.

Модель Comments це збірник коментарів для постів від інших користувачів. Під час роботи системи їхня кількість буде дуже швидко рости, тому необхідно винести їх в окрему модель для оптимізації системи

Модель Messenger представляє собою збірник даних та налаштувань для чатів – назва, аватар, тип чату та перелік учасників. Варто зазначити, що оскільки повідомлень буде дуже багато та швидкість їх поповнення буде висока, вони будуть зберігатися напряму до файлової системи сервера, а не до БД.

Модель Options зберігає налаштування для графічної оболонки соціальної мережі. Кожний користувач має можливість кастомізувати дизайн, як йому буде зручно. До кастомізації відносяться кольори елементів, шрифти, розмір шрифтів, налаштування до карти.

Модель Subscribition представляє собою сховище персональних контактів користувача. Там будуть зберігатися дані про друзів, підписки, заблоковані користувачі.

Модель Photoes зберігає в собі зв’язок завантажених фотографій до постів. Для зниження навантаження з БД, файли фотографій також зберігаються до файлової системи серверу. Дана модель зберігає назву файлу, ID поста та фізичну адресу файлу у файловій системі.

Виконуючи побудову ER-моделі системи були виділені наступні сутності та зв’язки:

1) User (ID, username, password, email, phone, rating, avatar, firstName, lastName, birthday, status, country, city, regToken, regDate, isConfirm, isBanned, isOnline, lastOnline);
2) Post (ID, lat, lng, name, comment, date, type, isCheck, isPrivate, ID_user);
3) Post_has_Rate(ID_post, ID_user, rating);
4) Photoes (ID, fileName, ID_post);
5) FriendList (ID_friend1, ID_friend2, status);
6) BlackList (ID_black1, ID_black2);
7) Subscribition (ID_sub1, ID_sub2);
8) Options (ID_option, Map_option, Site_option, ID_user);
9) Messenger (ID_messanger, name, date, avatar).

Модель Subscribition, що вказана на ER-діаграмі була розділена на три таблиці FriendList, BlackList, Subscribition. Зроблено це з ціллю розподілити навантаження на БД та розділити вхідну інформацію за призначенням.

Додаткові таблиці зв’язків «багато до багатьох»:

1) Users_has_Messenger (ID_User, ID_Messenger);
2) Post_has_Rate (ID_Post, ID_User, rating);
3) Comments (ID, Content, rating, date, ID_Post, ID_User).

4.5 Діаграма активності

Для візуального представлення діяльності використовується діаграма активності. Дія є фундаментальною одиницею визначення поведінки. Вона перетворює множину вхідних сигналів у множину вихідних сигналів.

Для розуміння послідовності виконання алгоритмів всередині системи, на рисунках 4.4—4.6 продемонстровано послідовність декількох алгоритмів базових елементів системи.

Діаграма активності (рис. 4.4) в даному випадку демонструє організацію завантаження даних на сторінці. Завдяки асинхронності методів завантаження, вдалося досягти швидкого завантаження сторінки для користувача. Одночасно виконується завантаження постів, маркерів та даних про користувача, котрі разом формують подану користувачу інформацію на головній сторінці.

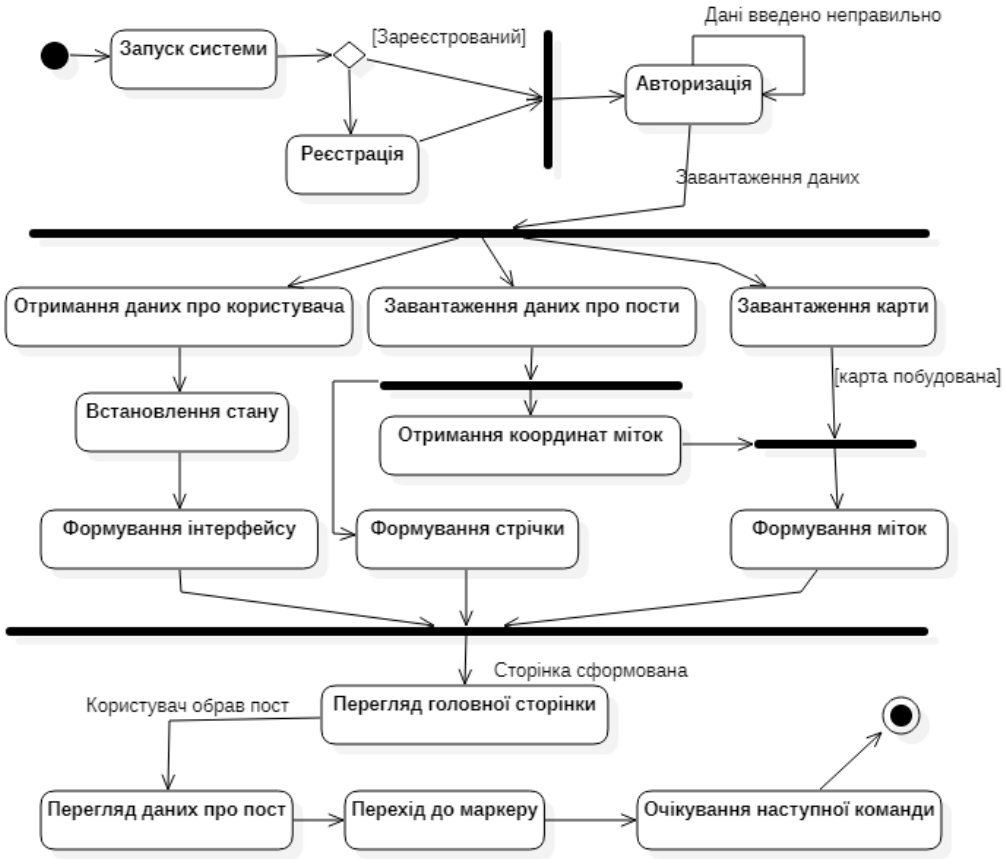

Рисунок 4.4 – Діаграма активності для варіанту використання «Watching other posts»

На рис. 4.5 вказана діаграма активності для сторінки створення посту. Під час завантаження сторінки виконується перевірка рейтингу користувача та, якщо він нижче 50, то система сповіщає користувача про неможливість створити пост.

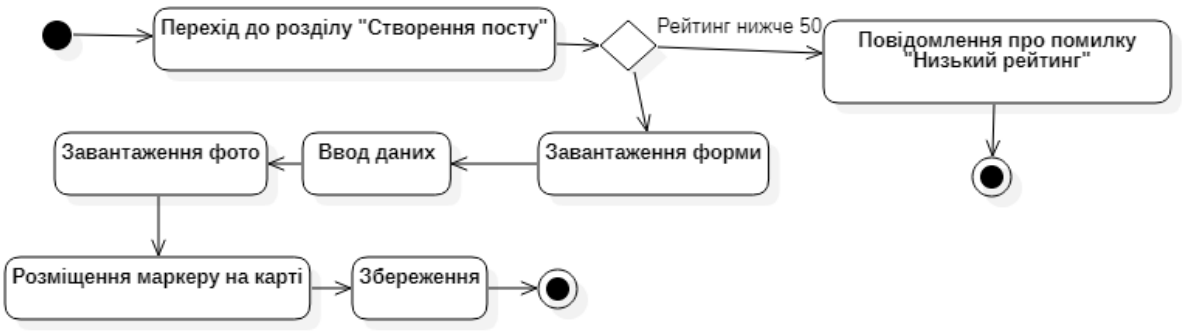

Рисунок 4.5 – Діаграма активності для варіанту використання «Create post»

На рис. 4.6 вказана діаграма активності для сторінки чатів. Оскільки в чаті приймають участь від двух та більше користувачів, необхідна подвійна перевірка під час завантаження – перевірка рейтингу нашого користувача, та перевірка рейтингу співбесідника. Якщо рейтинг когось із них нижче 20, то система сповіщає про неможливість завантаження чату. У випадку, якщо у чаті 3 та більше учасників, то чат відкривається лише для тих користувачів, чий рейтинг вище 20.

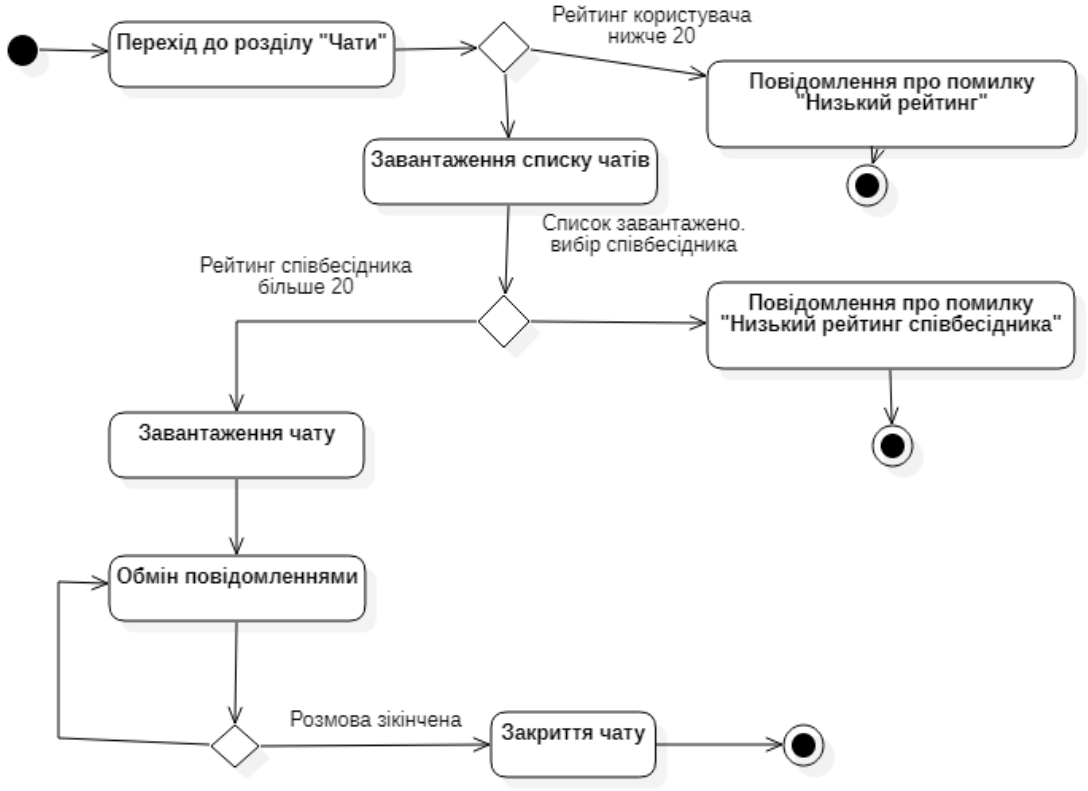

Рисунок 4.6 – Діаграма активності для варіанту використання «Chatting»

Оскільки переписка має циклічний характер (відправлення, читання, відповідь, читання), чат продовжує свою активність до тих пір, поки користувачі обмінюються повідомленнями. Коли набір повідомлень закінчився, чат переходить у режим очікування. Якщо користувачі покидають чат, то підключення до чату завершується.

4.6 Діаграми послідовності

Діаграма послідовності – це різновид діаграми в UML, яка відображає взаємодії об’єктів, впорядкованих за часом. Діаграма допомагає визначити об’єкти та їх взаємодію у графічному представлені [12].

На рисунку 4.7 продемонстрована деталізація логіки варіанту використання «Check post» (перевірка посту) та потік повідомлень серед об’єктів системи.

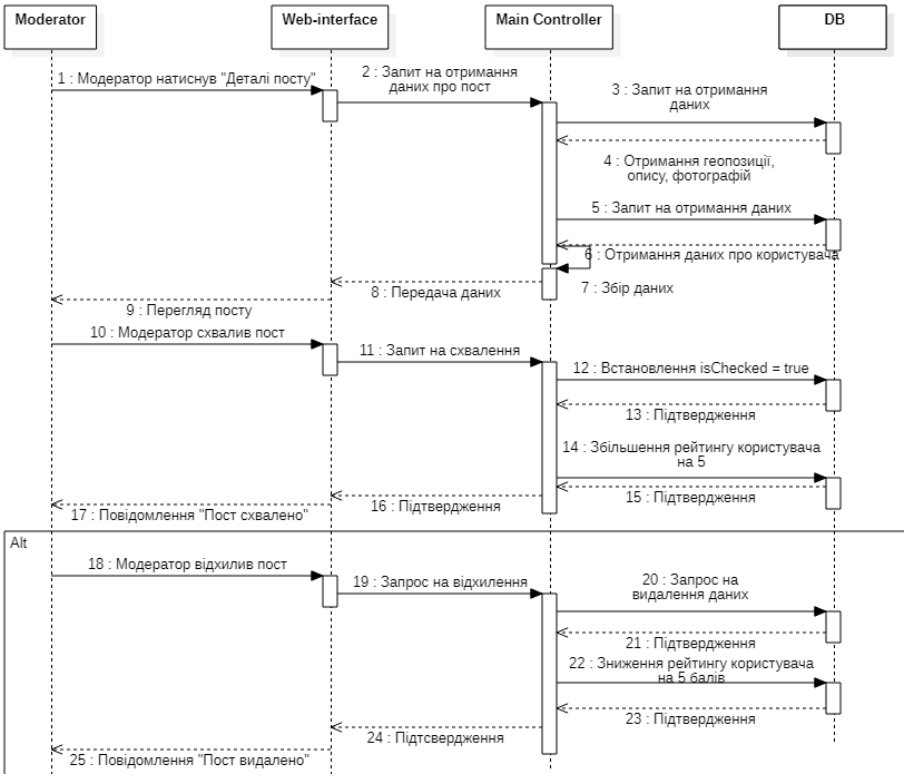

Рисунок 4.7 – Діаграма взаємодії для варіанту використання «Перевірити пост»

Як видно з діаграми, процес збору даних про пост відбувається у два етапи: отримання геопозиції, опису, фотографій та отримання даних про автора. Виконано це з ціллю спростити запити, що надсилаються до БД, що позитивно вплинуло на якість результату та гнучкість запитів. Аналогічно, підтвердження чи відхилення посту виконується у два запити.

На рис. 4.8 продемонстрована деталізація логіки варіанту використання «Chatting» (надіслання повідомлення)

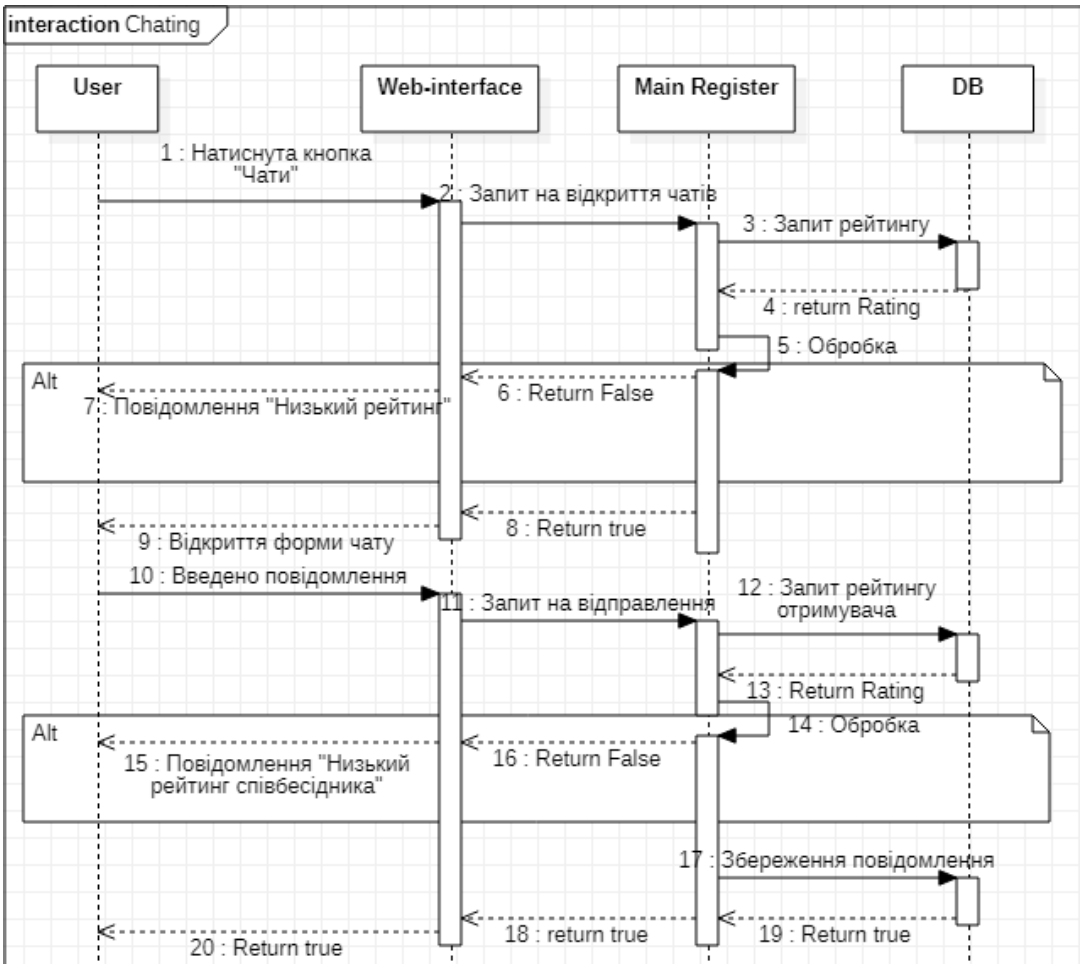

Рисунок 4.8 – Діаграма взаємодії для варіанту використання «Відправлення повідомлення»

4.7 Інтерфейс користувача

Інтерфейс користувача повинен бути легким та недратуючим, бути інтуїтивно зрозумілим та мінімалістичним. Це те, що повинно бути максимально зрозумілим більшості людей. У ході розробки були допущені усі норми створення візуальної частини продукту. Усі вони відображені на головній сторінці сайту. На рисунках 4.9—4.11 вказані початкові прототипи сторінок сервісу. Прототипи демонструють орієнтовне розміщення елементів, їх призначення та кількість.

На рис. 4.9 вказаний прототип головної сторінки.

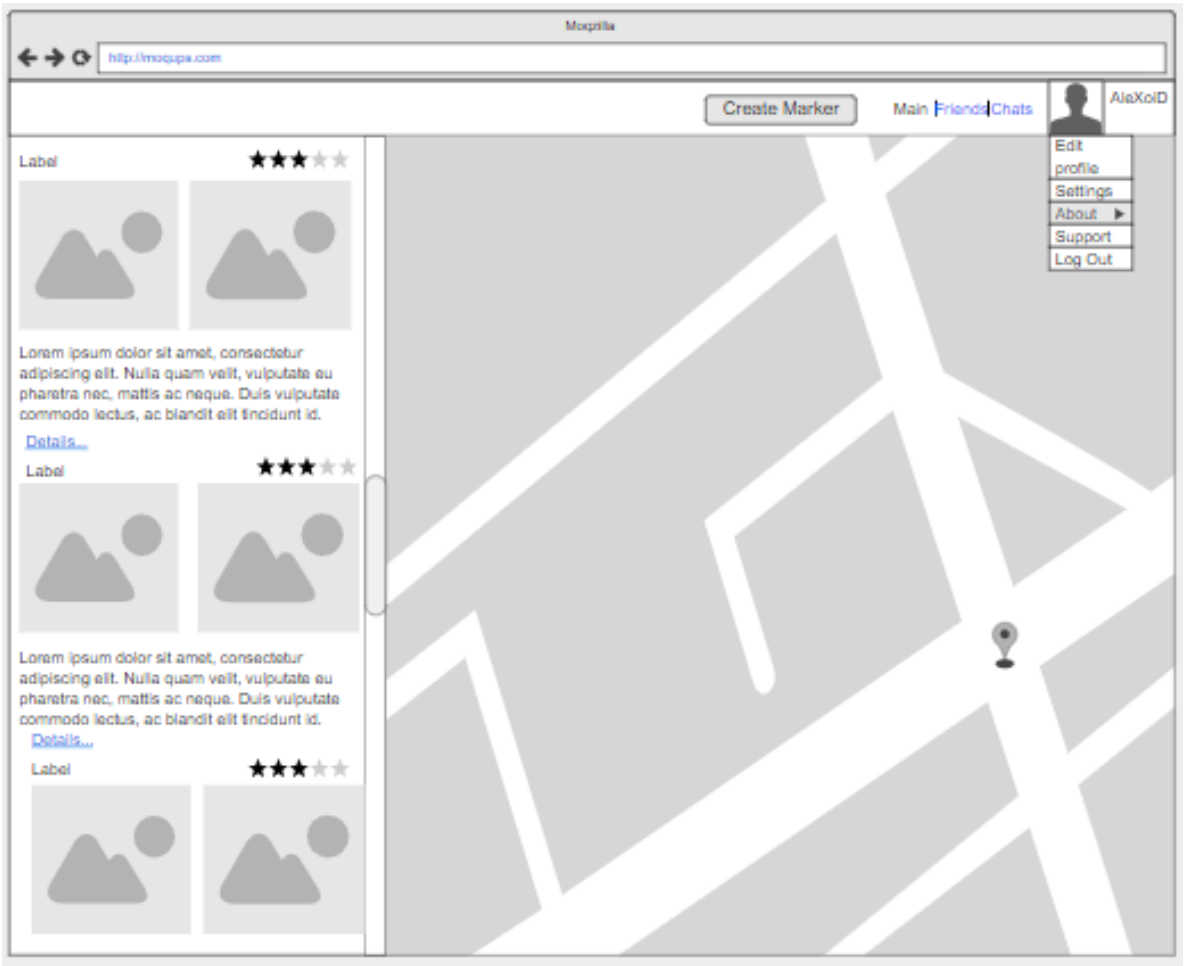

Рисунок 4.9 – Прототип головної сторінки соціальної мережі

Головна сторінка сервісу розробляється таким чином, щоб бути максимально зрозумілою для кожного користувача, щоб людина одразу зрозуміла, що де знаходиться та за що відповідає.

На рис. 4.10 наведено прототип сторінки «Друзі». Функціонал цієї сторінки повинен дозволяти користувачу шукати людей, що зареєстровані у системі, використовуючи систему фільтрації.

Сторінка буде поділена на три блоки. У першому буде розташована карта, на якій відстежується геолокація друзів. У другому – стрічка користувачів, що відповідають умовам пошуку. У третьому розташований блок фільтрів для пошуку.

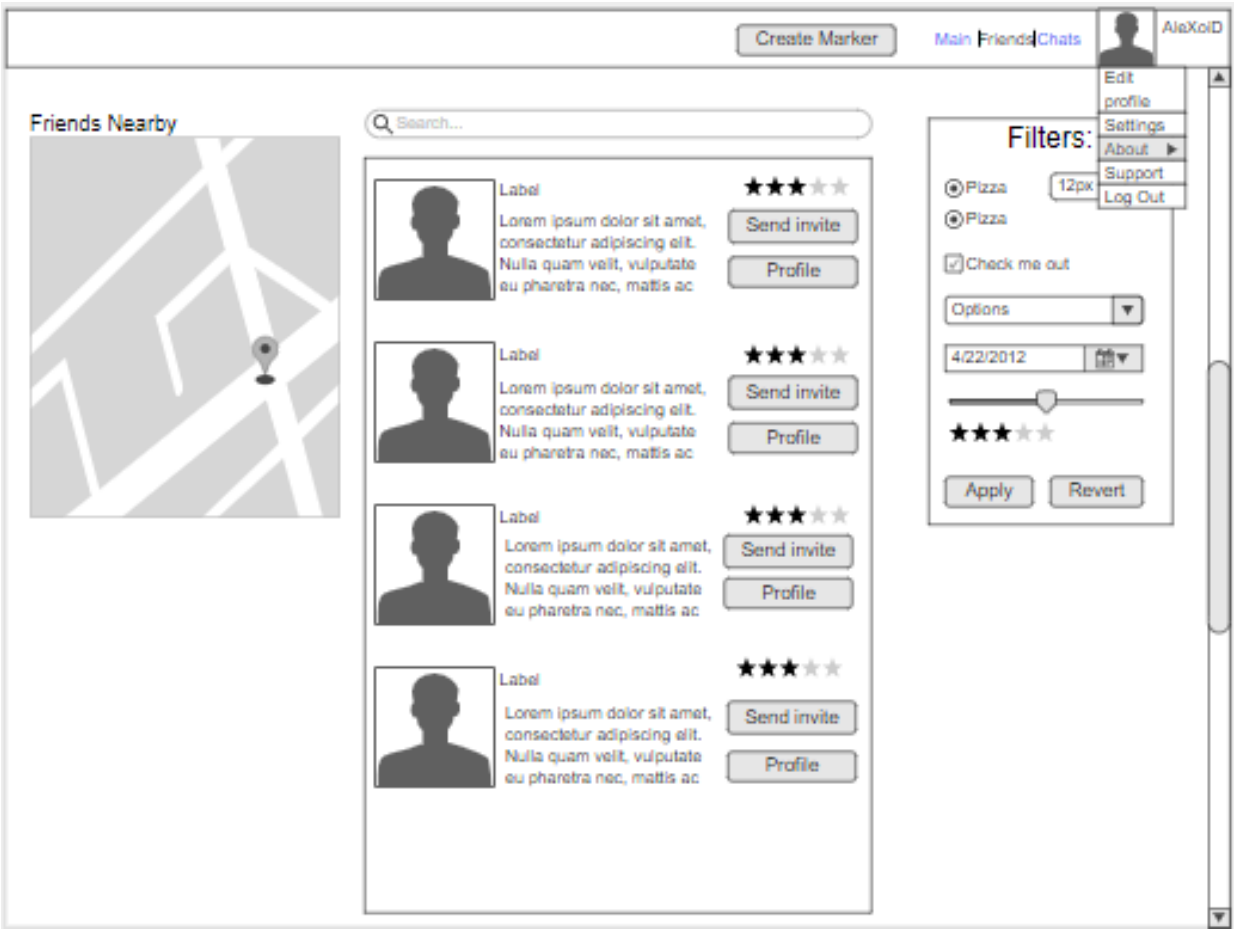

Рисунок 4.10 – Прототип сторінки «Друзі»

На рис. 4.11 вказаний прототип сторінки редагування профілю. Ця сторінка має використовуватись як для створення нового, так і для редагування існуючого користувача. Користувачу необхідно ввести свої персональні дані, що будуть

використовуватися системою під час роботи. Сама інформація, що вводить користувач – найпростіша, для ідентифікації користувача, його віку та контактні дані у разі виникнення проблем (втрата паролю, спілкування з адміністратором тощо).

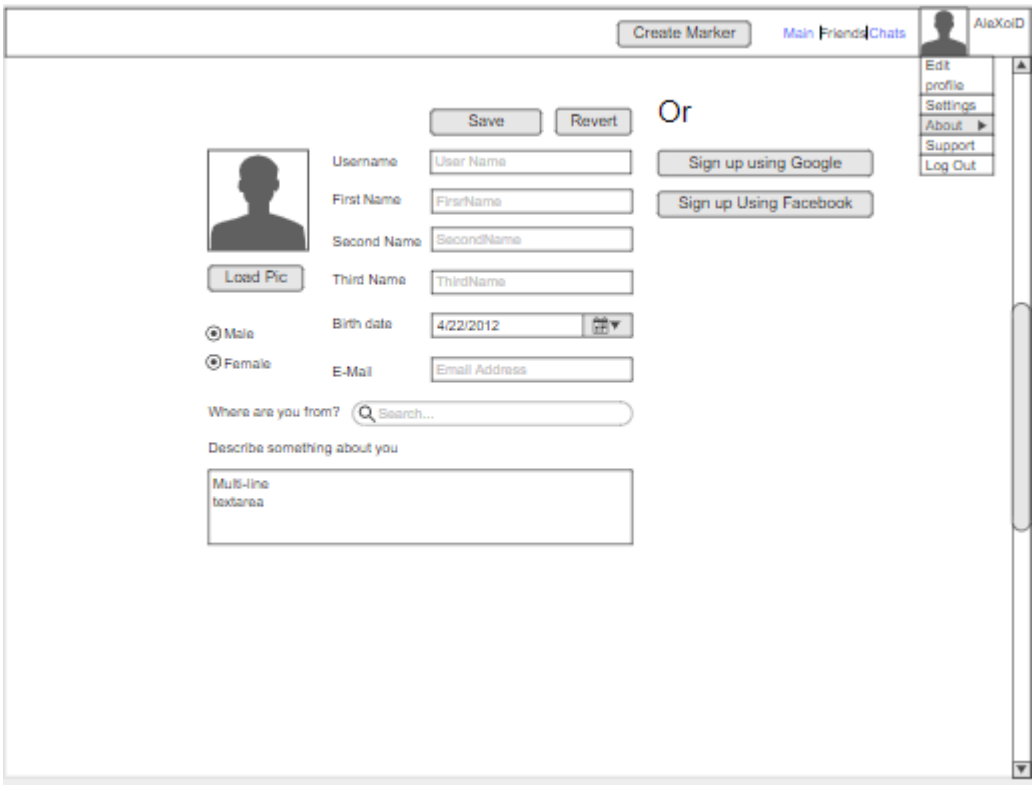

Рисунок 4.11 – Прототип сторінки «Створити профіль»

5 РЕАЛІЗАЦІЯ ТА ВИПРОБУВАННЯ ВЕБ-СЕРВІСУ

5.1 Програмна реалізація інтерфейсу

Програмна реалізація напряму впливає на інтерфейс користувача. В залежності від алгоритмів внутрішньої логіки виконуються зміни візуального інтерфейсу.

Інтерфейс реалізується у середі програмування Visual Studio Code за допомогою бібліотек мов програмування JavaScript React, JSX, NPM та Node.JS а також пакетному менеджеру yarn, обробнику стилів “emotions” та ін. При реалізації інтерфейс перетерпів деяких змін.

Інтерфейс головної сторінки наведено на рис. 5.1.

Відповідно до прототипу, головна сторінка мережі має мінімалістичний інтерфейс та максимально зрозумілий функціонал.

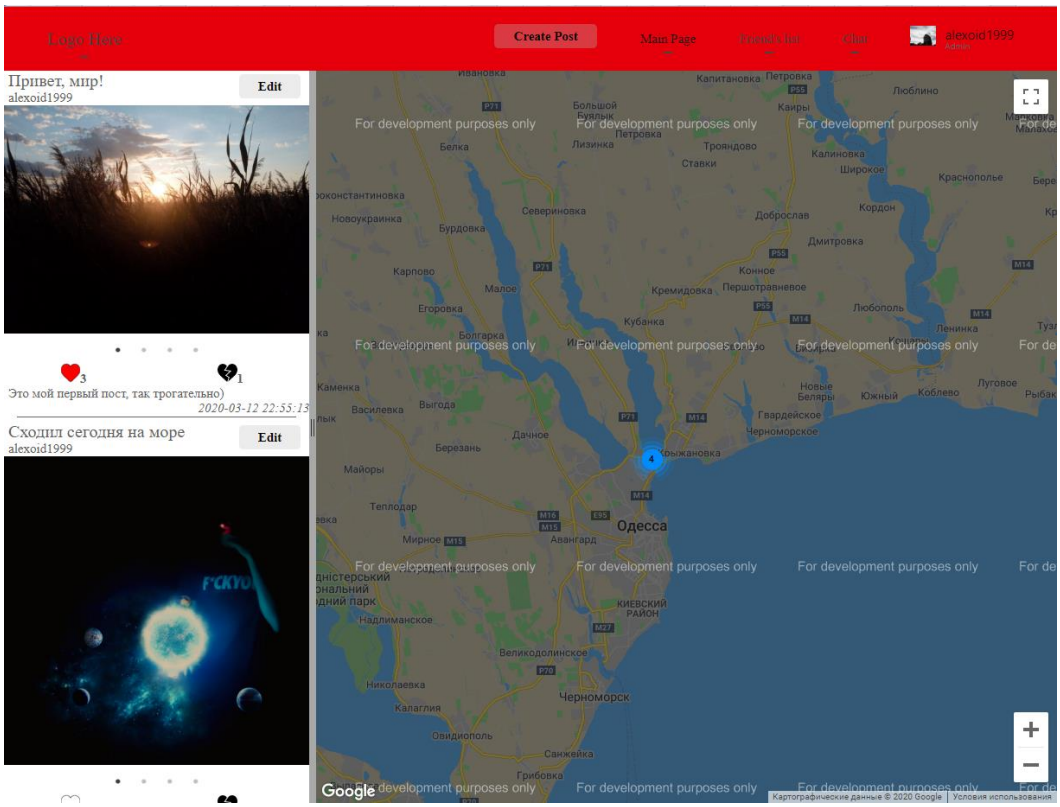

Рисунок 5.1 – Головна сторінка соціальної мережі (режим стрічки)

Як видно зі скріншоту, інтерфейс користувача отримав властивості автоматичного налаштування під сторінку. Отже, він буде гармонічно підлаштовуватися під екрани різних розмірів.

Сторінка змінює розміри своїх компонентів (Стрічка постів та карта) для того, щоб сфокусувати увагу користувача на обраному варіанті. Якщо користувачу легше шукати місця саме на карті, то він може налаштувати відношення стрічки до карти вручну за допомогою горизонтального повзунку (на скріншоті – сіра лінія між стрічкою та картою). Присутні обмеження зміни пропорції у рамках 25%-75% ширини екрану. Реалізовано це завдяки компоненту PageConstructor (див. Додаток А).

На рис. 5.2 продемонстровано головну сторінку сервісу під час перегляду окремо обраного посту.

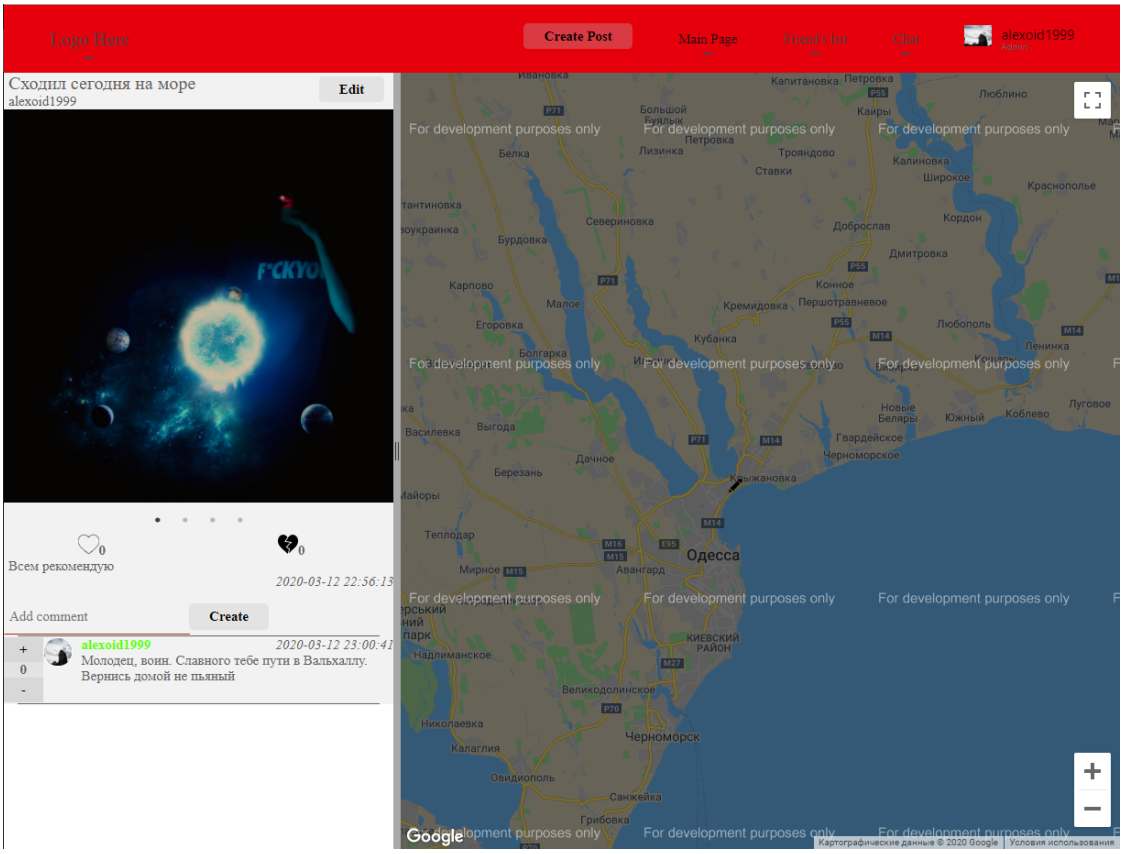

Рисунок 5.2 – Головна сторінка соціальної мережі (режим перегляду посту)

Завантаження списку постів розбито на «сторінки» по 5 постів за запит. Це зроблено для оптимізації трафіку. Коли користувач переходить до кінця списку, останній автоматично поповнюється ще п’ятьма постами. Так відбувається до тих пір, поки користувач не знайде шуканий результат, або не закінчаться пости, що відповідають умовам пошуку. Уривок коду, що виконує описану задачу наведено у Додатку А.

Технологія React, починаючи з 15-ої версії дозволяє використовувати так звані Хуки («Hooks»), що розроблені з ціллю повністю автоматизувати керування життєвим циклом («Life Cycle») компонентів та відмовитися від класів, повністю перелаштуватися на функціональне програмування компонентів[14]. В уривку коду, що знаходиться в Додатку А, представлена робота хука useEffect.

Спочатку, коли сторінка тільки завантажується, в кінці списку постів встановлюється блок із кнопкою «Load more» із прикріпленим атрибутом «ref». За даним атрибутом хук відстежує положення цього елементу на сторінці і, коли кнопка з’являється в області відображення (на екрані), відбувається виклик функції onCallNextPage(), із вказанням поточної довжини списку. Дана функція є функцією батьківського компоненту, що виконує запит на сервер (див. Додаток А).

Своєрідним ядром клієнтської частини, що відповідає за зв’язок із сервером, виступає функція httpPost (див. Додаток А). Вона є єдиним методом Front-end частини, через який відбувається відправка та отримання даних. Використовується у кожному місці, де необхідно відправити, чи отримати дані на сервер Node.JS. За основу цієї функції була взята бібліотека Socket.IO – оболонка для Web-sockets (long-pool request функція мови JavaScript для двустороннього обміну даними у реальному часі). Функція httpPost приймає на вхід адресу, куди відсилати, тип даних, що відсилаються та самі дані у вигляді JSON-змінної. Тип даних, вказаний на вході, впливає на заголовок, за яким дані будуть відправлені.

Після отримання відповіді, список постів поповнюється, сторінка перемальовується (re-render) та відображається з оновленою стрічкою, що містить нові пости.

Сторінка «Друзі» (рис. 5.3) на даний момент знаходиться на фінальних етапах розробки. Рейтинг користувачів відображається, як колір нікнейму (високий – зелений, низький – червоний). Завантаження списку користувачів відбувається аналогічно зі стрічкою постів.

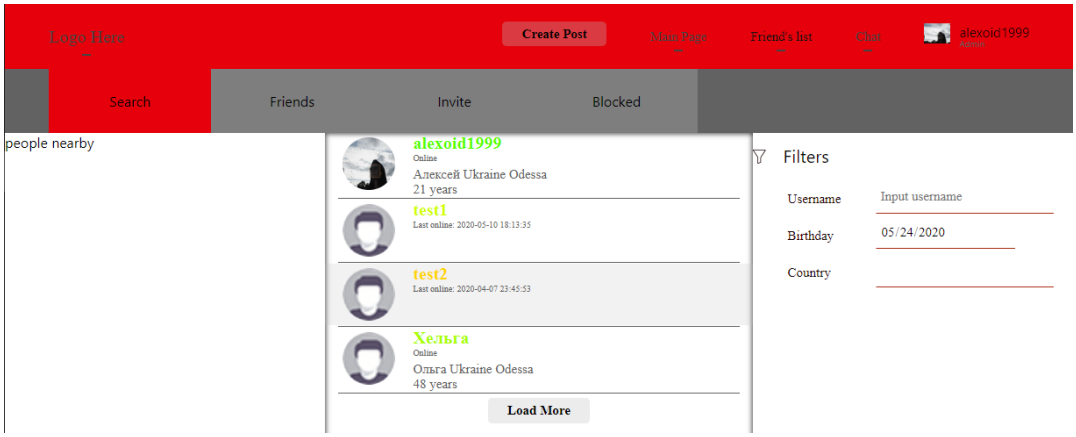

Рисунок 5.3 – Сторінка «Друзі», що зараз в розробці

Сторінка редагування профілю наведена на рис. 5.4:

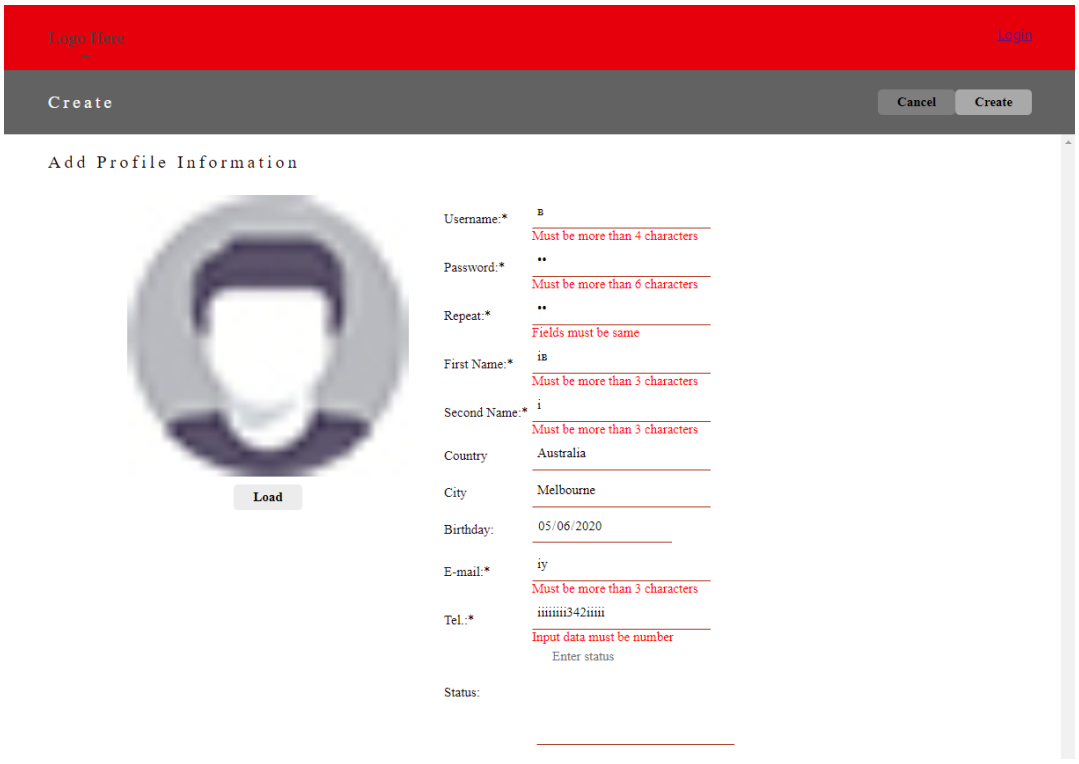

Рисунок 5.4 – Сторінка «Створити профіль»

За потреби, кожне поле вводу даних можна захистити від помилкових символів, або зробити обов’язковим до заповнення за необхідним шаблоном. Реалізація наведена у Додатку А.

Головною особливістю мови TypeScript є жорстка типізація даних при передачі між компонентами – параметру може бути присвоєне тільки те значення, що задовольняє типу (параметр, що об’явлений із типом number може присвоїти тільки число і т.д.). У даному випадку, батьківський компонент TextInputItem приймає на вхід дані, що вказані в інтерфейсі IInputProps. Також TypeScript підтримує створення власних типів даних, що продемонстровано створеним типом IRestict. Інтерфейс, котрий треба використовувати в даному типі, визначається спільним для всіх інтерфейсів параметром type. Коли параметр type обраний, TypeScript запитує інші параметри, що вказані поряд із відповідним type. Після цього, можна викликати екземпляр TextInputItem з потрібними обмеженнями на внесення даних (див. Додаток А).

У випадку, якщо умова заповнення не виконана, користувачу відображається повідомлення про помилку, а підтвердження дії блокується – на рисунку 4.4 кнопка «Create» виділена сірим і не натискається. Якщо все гаразд та умови виконані, то вона стає прозорою, як сусідня кнопка «Cancel», а при наведенні курсором – червоніє.

Дана сторінка використовується як для створення нового, так і для редагування існуючого профілю користувача (рис. 5.5).

Також був реалізований інтерфейс сторінки чатів (рис. 5.6). Компонент цієї сторінки, на даний момент, єдиний, що відправляє та отримує дані саме до Node JS серверу. Тобто, сторінка підключена до веб-сокету та оновлюється у реальному часі. Якщо інший користувач виконає якісь дії на цій сторінці (створить чат, де є поточний користувач, напише/видалить повідомлення), то це одразу побачить наш користувач без потреби у додаткових діях (оновлення сторінки, натискання кнопки, тощо).

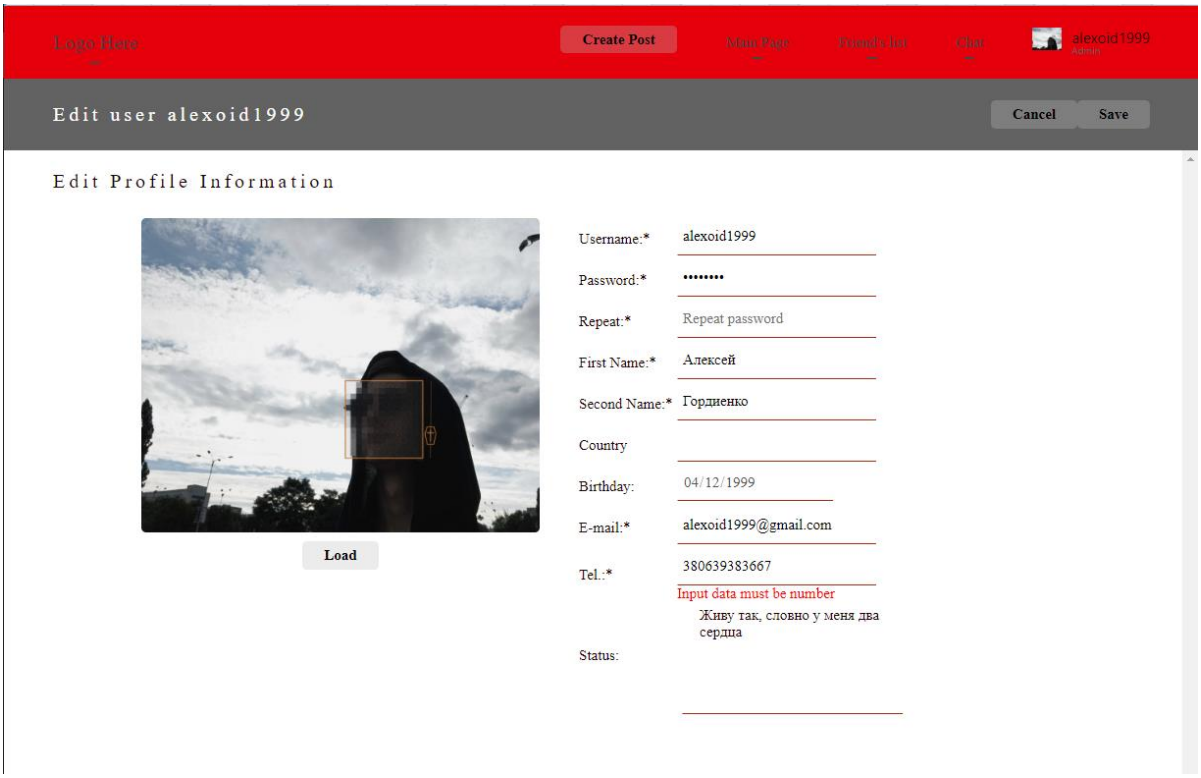

Рисунок 5.5 – Редагування існуючого профілю

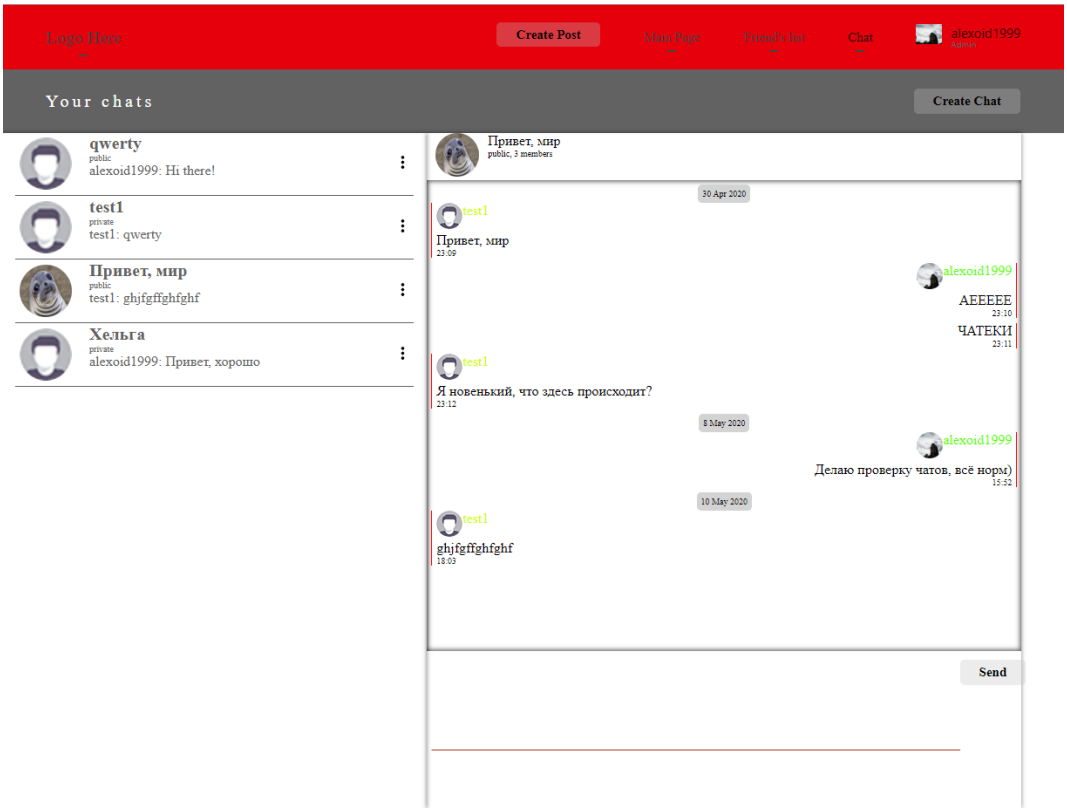

Рисунок 5.6 – Сторінка «Чати»

5.2 Реалізація нейронної мережі

Нейронна мережа, як вказано у Розділі 2, розроблена за допомогою фреймворку TensorFlowJS. Були створені окремі компоненти по пошуку та читанню фотографій, створення, збереження, завантаження та навчання моделі, а також керуючий скрипт.

Для створення навчальної вибірки, було завантажено 1600 фотографій, котрі були відсортовані по категоріям. Скрипт налаштований таким чином, що під час запуску треба вказати папку, що зберігає папки з категоріями (рис. 5.7), а також папку зі збереженою раніше моделлю.

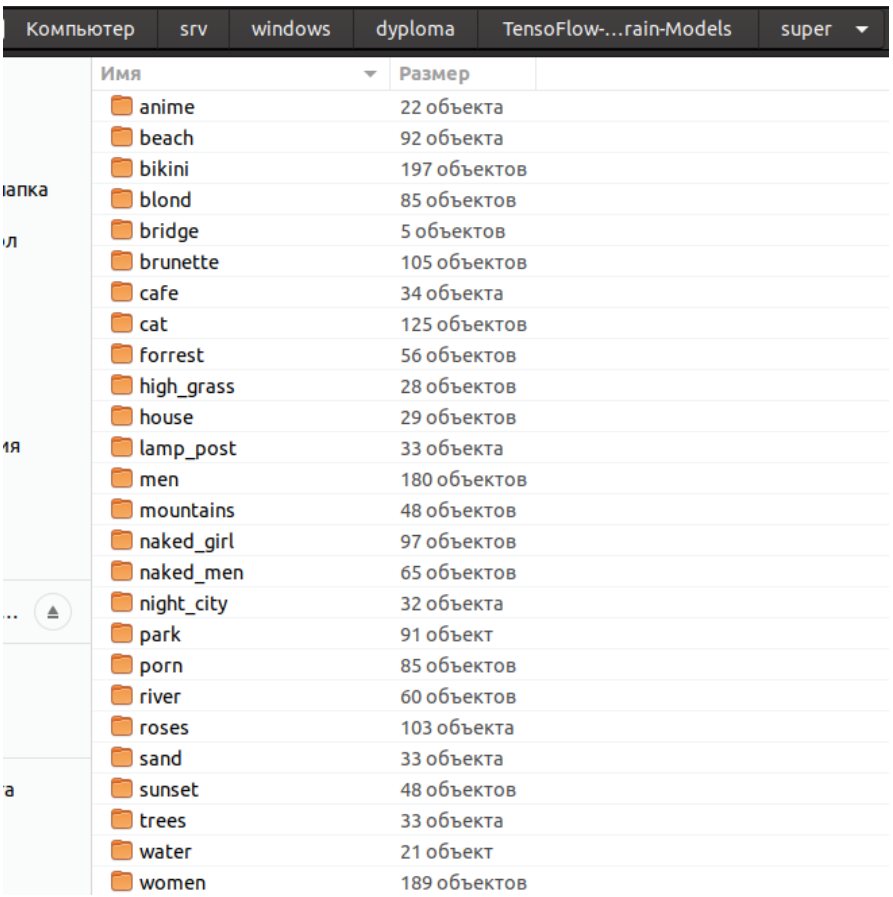

Рисунок 5.7 – Структура відсортованої навчальної вибірки

Якщо останнє відсутнє (перший запуск), то скрипт виконує завантаження пустого шаблону mobileNet моделі, що є основою під час створення нової моделі.

Емпіричними дослідженнями було визначено, що оптимальними параметрами мережі є створення двох внутрішніх шарів по 200 нейронів кожний. Перший шар має функцію активації «softplus», другий шар – «hardSigmoid». Присутня функція оптимізації за алгоритмом «Adam». Лістинг коду, що відповідає за побудову моделі приведено у Додатку А:

Під час запуску скрипту, виконується перегляд вказаної папки на наявність .jpg файлів та кожному присвоюється категорія з назвою, що відповідає назві папки, де була знайдена фотографія (рис 5.8).

Рисунок 5.8 – Демонстрація процесу зчитування фотографій під час навчання

На даний момент нейронна мережа здатна розпізнавати 26 різних образів з точністю 70-90%. Однак, присутні так звані «артефакти» – окремі фотографії, котрі нейронна мережа не може правильно розпізнати через особливості самої фотографії – багато шумів, розмитість, дим тощо. У таких випадках модель відмічає фотографії як ті, що потребують людського дослідження.

Варто зазначити, що система розпізнає образ усієї сцени на зображенні, а не окремі її об’єкти, тому у разі знаходження кількох образів на одній картинці, ймовірність буде поділена між ними.

5.3 Керівництво користувача

Щоб розпочати роботу з системою, необхідно відкрити інтернет браузер на ваш смак та ввести в адресне поле наступну адресу: localhost:3000. Далі користувача зустрічає сторінка авторизації. Якщо ви – новий користувач сервісу, то на сторінці є кнопка «Create profile». При натисканні на цю кнопку вас буде переадресовано на сторінку створення профілю. Вказуйте актуальні дані, особливо нікнейм, пароль та вашу поштову скриньку. Після заповнення всіх обов’язкових полей та натискання кнопки «Create», на вказану поштову скриньку буде відправлено лист із підтвердженням реєстрації. Відкрийте письмо із заголовком «ERA registration» та перейдіть за посиланням «main page». Вас буде переадресовано на сторінку авторизації. Введіть нікнейм та пароль, що були вказані раніше. У разі успіху перед вами відкриється головна сторінка сервісу з картою та стрічкою постів. Для того, щоб переходити між різними сторінками системи, потрібно натиснути відповідний пункт у меню користувача, що знаходиться у верхній правій частині сторінки. Після натискання, сторінка одразу зміниться на обрану.

5.4 Інструкція із налаштування програмної системи

Для того, щоб продукт працював коректно, необхідно:

- стабільно-працюючий інтернет;
- операційна система Windows, Linux, Mac, iOS, Android;
- інтернет-браузер (будь-який);

Коли всі вимоги були дотримані, можна почати користуватись продуктом. Щоб це зробити, необхідно відкрити інтернет-браузер та перейти по адресі: localhost:3000.

5.5 Керівництво системного програміста

Загальні відомості про програму

Програма може використовуватися будь-якою людиною будь-якого віку, яка має змогу користуватися смартфоном або комп’ютером. Вона допоможе всім охочим пришвидшити пошук новин у різних регіонах, а також знаходити нових знайомих у різних точках планети.

Основні варіанти використання:

1. Створення профілю та публікацій.
2. Створення точки геолокації для пошуку новин.
3. Створення повідомлень та коментарів.

Платформа повинна використовуватися Linux або Windows. Оперативна пам’ять складає не менше 128Мб. Вільний простір на жорсткому диску не меше за 300МБ. Процесор Pentium 3 або Core i5 і більше. Розрадяність 32 або 64 система. Відеоадаптер 3D адаптер n-Vidia або Intel, AMD/ATI.

Налаштування програми

Для налаштування даного програмного продукту запускається NodeJS сервер, який контролюється локальною машиною.

Виставляються показники, що необхідні для постійної роботи програмного продукту без збоїв. Для цього виціляться ЦПУ 1, та пам’ять 500Мб. Після усіх цих налаштувань користувачеві видається посилання на програмний продукт, для подальшої роботи із ним.

Перевірка програми

Для перевірки програми, потрібно звернутися до сервера, та зробити тестові запити, при роботі із програмою. Створити користувача, відредагувати його, створити публікацію неправильно, відредагувати її, створити коментар та видалити його, зберегти інформацію та перевірити cookies та наостанок перевірити логи програми.

Повідомлення системному програмісту

Сервер працює без помилок – сервер запущений та відповідає на запити.

Запити відправлені та відповіді отримані – сервер реагує на запити правильно.

Виділено ЦПУ 1 та 500мб пам’яті – визначені усі показники роботи.

Перевірено логи сервера – жодної помилки під час роботи.

Перевірено функціонал програми – пройдені усі тестові запити без помилок.

Характеристики програми

Система розгортається з використанням пакетного менеджеру Yarn, що входить до складу Node.JS. Для запуску програми необхідно завантажити архів з кодом програми. Структура проекту налічує чотири папки: Front, NodeServer, Messages, Tensorflow Retrain Model. Для запуску серверу необхідно перейти до папки Front та виконати наступні команди:

```
npm install yarn
yarn install
yarn start
```

Після виконання цих команд потрібно зачекати 15 секунд, після чого клієнтська частина буде увімкнена та доступна за локальною адресою localhost:3000.

Для запуску серверної частини веб-сервісу необхідно перейти до папки NodeServer та виконати такі самі дії. Коли серверна частина запрацює, необхідно ознайомитись з логами серверу, де будуть вказані базові налаштування для підключення до серверу. У разі потреби, на клієнтській частині у файлі Front/src/backend/httpPost.ts необхідно встановити ip-адресу та порт, що були вказані у логах серверу – необхідно для запуску у режимі розробника.

Вхідні та вихідні дані

Вхідними даними програмного продукту інтерфейс, за допомогою якого користувач взаємодіє із системою, створює та редагує профіль, створює та читає публікації, відвідує профілі інших користувачів, обмінюється повідомлення, шукає новини за геолокацією. Також вхідними даними можна вважати ті дані, які користувач вводить під час створення публікацій. Отже, вихідними даними можуть бути створені публікації та коментарі, а також інтерфейс.

Повідомлення

Сервер працює з помилками – необхідно знайти та виправити помилки, потім запустити сервер знову.

Запити відправлені, проте відповіді отримані із помилками– необхідно визначити причину помилок та виправити їх.

Виділено ЦПУ 2 та пам’ять 1000 мб – необхідно знайти причину збільшення пам’яті та виправити її.

Перевірено логи сервера із помилками – виправити помилки та перезапустити сервер.

Перевірено функціонал програми, частина не працює – необхідно виправити помилки та перезапустити сервер.

Виконання програми

Для виконання роботи програми потрібно перевірити роботу сервера. Далі, подивитися чи були перезапуски чи критичні помилки. Тепер, потрібно перейти на потрібний адрес програми та створити користувача, відредагувати його, створити публікацію неправильно, відредагувати її, створити коментар та видалити його, зберегти інформацію та перевірити cookies та наостанок перевірити логи програми.

Повідомлення оператору

Для перевірки програми, потрібно звернутися до сервера, та зробити тестові запити, при роботі із програмою. Створити користувача, відредагувати його, створити публікацію неправильно, відредагувати її, створити коментар та видалити його, зберегти інформацію та перевірити cookies та наостанок перевірити логи програми.

5.6 Керівництво з технічного обслуговування

Керівництво призначено для перевірки та підтримки нормального функціонування програмного продукту, нагляду та швидкого вирішування виникаючих помилок. В якості експлуатаційних документів наведено керівництво програміста, керівництво системного програміста, керівництво оператора.

Загальні вказівки

Для перевірки роботи програми потрібно перевірити роботу сервера. Далі, подивитися чи були перезапуски чи критичні помилки. Тепер, потрібно перейти на потрібний адрес програми та створити користувача, відредагувати його, створити публікацію неправильно, відредагувати її, створити коментар та видалити його, зберегти інформацію та перевірити cookies та наостанок перевірити логи програми.

Вимоги до технічних засобів

Платформа повинна використовуватися Linux або Windows. Оперативна пам’ять складає не менше 128Мб. Вільний простір на жорсткому диску не меше за 300МБ. Процесор Pentium 3 або Core i5 і більше. Розрадяність 32 або 64 система. Відеоадаптер 3D адаптер n-Vidia або Intel, AMD/ATI.

Опис функцій

Виставляються показники, що необхідні для постійної роботи програмного продукту без збоїв. Для цього виціляться ЦПУ 1, та пам’ять 500Мб.

Вхідними даними програмного продукту інтерфейс, за допомогою якого користувач взаємодіє із системою, створює та редагує профіль, створює та читає публікації, відвідує профілі інших користувачів, обмінюється повідомлення, шукає новини за геолокацєю. Також вхідними даними можна вважати ті дані, які користувач вводить під час створення публікацій. Отже, вихідними даними можуть бути створені публікації та коментарі, а також інтерфейс.

6 ТЕСТУВАННЯ ВЕБ-СЕРВІСУ

6.1 Призначення і область використання

Даний програмний продукт може використовуватися будь-яким користувачем в інтернеті, який бажає знайти нові улюблені місця відпочинку. Продукт стане важливим помічником для людей під час подорожей, швидко шукаючи всю потрібну інформацію для них.

Користувачі матимуть змогу обмінюватися повідомленнями та висловлювати свої думки, в якості публікацій, тому потрібно проводити контроль публікаціями користувачів, щоб вони відповідали усім правилам.

Програма може використовуватися будь-якою людиною будь-якого віку, яка має змогу користуватися смартфоном або комп’ютером. Вона допоможе всім охочим пришвидшити пошук новин у різних регіонах, а також знаходити нових знайомих у різних точках планети.

6.2 Технічні характеристики

Весь процес пошуку та виводу даних займає декілька секунд і більше не потребується, поки користувач не перезапустить сторінку. Цей принцип дозволяє користувачу швидко переміщатися між сторінками без перезапуску.

Під час виконання процесу реєстрації програма перевіряє чи зареєстрований користувач с таким ідентифікатором. Якщо користувач зареєстрований, він буде переправлений на сторінку авторизації. Далі йде перевірка по електронній пошті або логіну. Якщо користувач з такою поштою вже існує у системі, то буде виведено повідомлення про це.

Під час виконання варіанту використання «Створити публікацію» система звертається до БД один раз, попередньо виконуючи валідацію даних, чи відповідає інформація первинним вимогам – наявність фотографій, опису, геопомітки а також рейтинг користувача дозволяв створювати пости. Дані, які вводить користувач

перевіряються на одразу на пристрої клієнта. При помилковому вводі, система повідомляє користувача про це, і користувач повинен ввести дані правильно для успішного збереження. Після успішного збереження даних, система переадресує користувача на головну сторінку, та сповістить користувача про те, що публікацію було відправлено на модерацію. В разі схвалення системою модерації публікація з'явиться у стрічці постів. Інакше – у профілі користувача з'явиться повідомлення з посиланням на публікацію та проханням відредагувати пост.

Під час геолокації виконується пристрій клієнта посилає на сервер дані про геолокацію (при умові, що користувач дозволив використовувати такі дані). При успішному визначенню місця, система відображає користувачу карту та його позицію на ній. Також на карті таким самим чином будуть відмічені інші користувачі системи, що є друзями користувача.

6.3 Очікувані техніко-економічні показники

Веб-сервіс, що був створений у рамках обраної теми, був протестований потенційними користувачами на відповідність поставленим вимогам та виконанню поставленої мети – підвищити конверсію серед маловідомих місць міста. Тестування було проведено методом соціального опиту користувачів за наступними критеріями: зручність використання, якість інформації, доступний функціонал, потенційна конверсія.

- Критерій «Зручність використання» показує якість виконаних методів та зручність реалізованого інтерфейсу;
- Критерій «Якість інформації» показує наповненість сервісу повною, актуальною та зрозумілою користувачу інформацією;
- Критерій «Доступний функціонал» показує наповненість сервісу функціями, що користувач очікує від веб-сервісу подібного призначення;

- Критерій «Потенційна конверсія» показує якість виконання мети сервісу, а саме – зацікавити людей у відвіданні місця та підвищити вірогідність того, що людина дійсно туди піде.

Соціальний опит був проведений серед 30 людей, результати були зібрані, підсумовані та зображені на рис. 6.1-6.2. Кожна людина обирала тільки один із чотирьох сервісів-аналогів та нового веб-сервісу у кожному з критеріїв.

За результатами опиту було виявлено приріст потенційної конверсії серед користувачів на 30% у порівнянні з найбільш ефективними аналогами, що показує високу ефективність веб-сервісу навіть у не закінченому вигляді (веб-сервіс на даний момент працює у тестовому режимі). На жаль, підрахувати реальну конверсію у рамках карантину є неможливою задачею. Але з головною задачею – зацікавити людину у вивченні міста більш детально та ознайомити з новими місцями, система впоралася на відмінно.

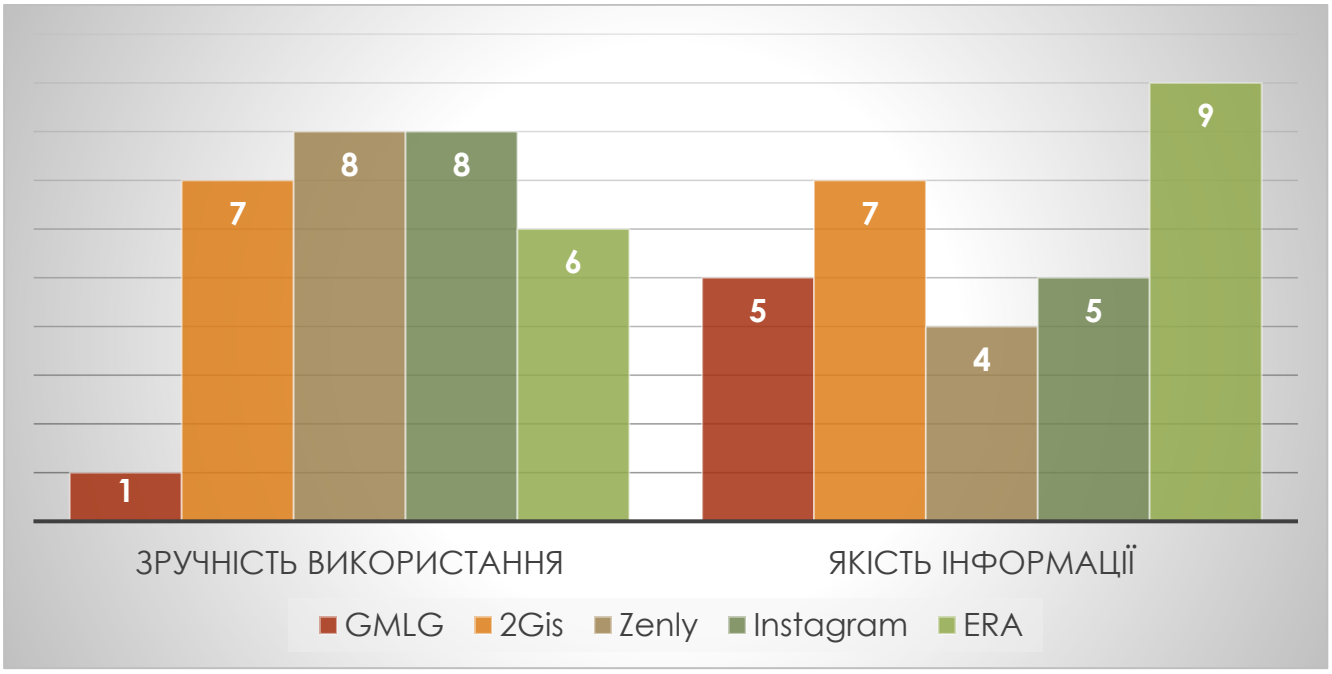

Рисунок 6.1 – Результати експлуатаційних випробувань для категорій «Зручність використання» та «Якість інформації»

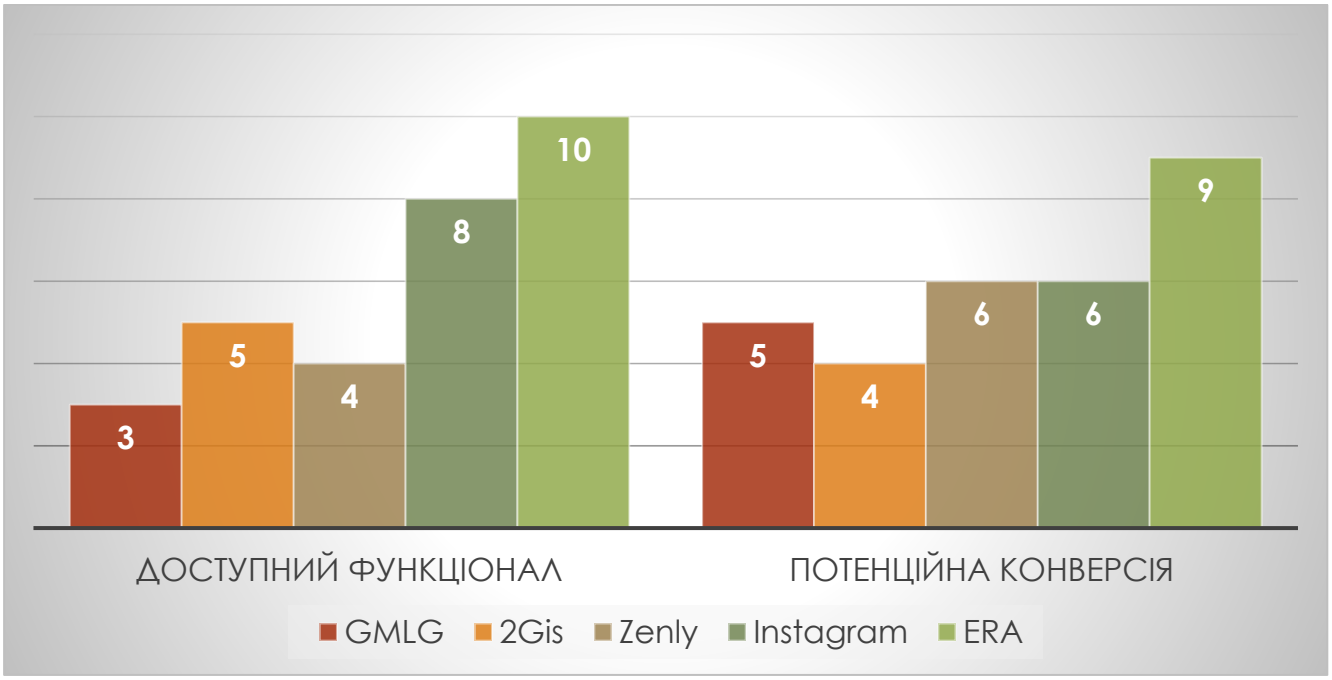

Рисунок 6.2 – Результати експлуатаційних випробувань для категорій «Доступний функціонал» та «Потенційна конверсія»

6.4 Оцінювання якості

Програмний продукт – це веб-застосування для вивчення міст, подорожей та комунікації, тому його можна віднести до класу – 5014.

1 *Надійність:*

1.1 Якщо користувач створює пост, то система показує повідомлення о результаті менш ніж за 4 секунди;

1.2 У випадку критичної помилки система зберігає роботоспроможність у 80% випадків;

1.3 Якщо система запитує у БД дані по фільтрації, то ймовірність отримати коректний результат не менше, ніж 0.8;

1.4 Якщо система зберігає дані в БД, то ймовірність вдалого збереження не менше 0.9.

2 *Продуктивність:*

2.1 Якщо користувач запитує у системи дані про іншого користувача, то система відповідає на запит менше ніж за 0.5 секунд;

2.2 Якщо користувач запитує дані про гео-помітку, то система відповідає на запит менше ніж за 1 секунду;

2.3 Якщо користувач фільтрує дані, то система оброблює 1500 постів за секунду;

2.4 Якщо користувач запитує рейтинг другого користувача, то отримує відповідь менше ніж за 0.5 секунд.

3 *Безпека:*

3.1 Якщо система запитує інформацію з БД, то отримує її в цілісності не менш, ніж у 94% випадків;

3.2 Якщо користувач створює фейкові пости, чи відмічує місця, небезпечні для життя інших користувачів, то система відсіює їх більш ніж у 70% випадків;

3.3 Якщо зміст поста, чи безпечність відміченого місця стали під сумнівом, то в 90% випадків вони відправляються на ручну модерацію;

3.4 Якщо користувач реєструється, то його дані будуть захищені в 98% випадків.

4 *Usability:*

4.1 Якщо користувач створює пост, то йому це вдається не більше ніж за 40 секунд;

4.2 Якщо користувач бажає оцінити іншого користувача, то йому це вдається не більше, ніж за 15 секунд;

4.3 Якщо користувач бажає відфільтрувати список постів, то йому це вдається менше, ніж за 10 секунд;

4.4 Якщо модератор бажає перевірити пост, то в нього це виходить менш, ніж за 40 секунд.

5 Супроводження:

5.1 Якщо компанія бажає змінити функціонал системи, то зміни вступають у силу менш ніж за 1 місяць;

5.2 Якщо компанія бажає відновити робоче обладнання, то система почне працювати з ним не більш, чим за місяць;

5.3 Якщо компанія бажає портувати систему на інший гаджет, то поставлена задача буде виконана не більш, ніж за 7 місяців;

5.4 Якщо розробник вирішує провести повну діагностику, то система почне працювати не більше ніж через день.

Нижче приведені показники якості веб-застосунку за таблицею оцінювання якості програмного продукту (ГОСТ 19.301-2000) [23].

Таблиця 6.1 – Показники групи «Надійність»

Найменування групи і комплексних показників якості	Діапазон	Значення у ТЗ
1.1. Стійкість функціонування	1±0,5	0,90
1.2. Працездатність	1±0,5	0,80

Таблиця 6.2 – Показники групи «Здатність до супроводу»

Найменування групи і комплексних показників якості	Діапазон	Значення у ТЗ
2.1. Структурність	0,97±0,2	1
2.2. Простота конструкції	0,97±0,25	1
2.3. Наочність	1±0,2	1
2.4. Повторюваність	0,9±0,3	0,7
2.5 Повнота документації	1±0,35	1

Таблиця 6.3 – Показники групи «Зручність використання»

Найменування групи і комплексних показників якості	Діапазон	Значення у ТЗ
3.1. Легкість освоєння	0,9±0,3	0,9
3.2. Доступність програмних документів	0,95±0,35	0,8
3.3. Зручність експлуатації та обслуговування	0,85±0,35	0,85

Таблиця 6.4 – Показники групи «Універсальність»

Найменування групи і комплексних показників якості	Діапазон	Значення у ТЗ
5.1. Гнучкість	0,9±0,3	0,8
5.2. Мобільність	0,7±0,35	0,5
5.3. Здатність до модифікації	0,9±0,35	1,2

Таблиця 6.5 – Показники групи «Функціональність»

Найменування групи і комплексних показників якості	Діапазон	Значення у ТЗ
6.1. Повнота реалізації	1±0,25	0,85
6.2. Узгодженість	1±0,2	-
6.3. Логічна коректність	1±0,2	-
6.4. Перевіреність	0,9±0,2	0,8
6.5. Захищеність	0,8±0,2	0,9

6.5 Об’єкт випробувань

Після реалізації модулю було проведено тестування основних класів.

Клас App є зв’язуючим компонентом між Front та Back частинами проекту та коренем ієрархії класів клієнтської частини. Він зберігає в собі дані користувача, що використовуються іншими компонентами в процесі роботи. Під час роботи системи постійно відображається у вигляді контейнеру для інших класів, що зробило можливим реалізацію односторінкового проекту.

Клас Header описує параметри та методи заголовку сторінки. Містить в собі посилання на різні компоненти системи та меню користувача. Відображається завжди під час роботи із системою.

Клас MainPage представляє собою набір методів для відображення головної сторінки соціальної мережі. Є батьківським класом для компонентів FeedList, GoogleMapsAPI та виконує зв’язуючу роль між ними.

Клас FeedList представляє собою методи відображення та взаємодії постів користувачів. Відображається у вигляді стрічки постів у лівій частині екрану.

Клас GoogleMapsAPI виконує відображення карти та обробку методів взаємодії з картою.

Клас ChatPage – набір методів по обробці запитів для повідомлень, їх відправлення, видалення. А також – набір методів для обробки чатів, їх редагування, створення, сортування. Є батьківським компонентом для класів ChatEditor, ChatFeed, Messenger.

6.6 Мета випробувань

Метою випробувань є визначення ефективності створеного програмного продукту.

6.7 Кошти і порядок випробувань

Випробування проводяться за допомогою автоматичних юніт-тестів, симуляцією дій користувача та перевіркою роботи нейронної мережі.

6.8 Методи випробувань

Розглянемо функціональну діаграму причинно-наслідкових зв’язків для прецеденту «Watching other posts».

«Watching other posts» реалізований завдяки комплексу методів, що реалізовані у системі. Загальна послідовність отримання даних виглядає наступним чином:

1) Функція-контролер App викликає функцію getMarkers();
2) Виклик функції httpGet() за адресою “localhost:5001/”;
3) Виклик функції SQL для сбору даних;

4) httpGet() отримує відповідь у вигляді JSON-файлу, передає його далі та завершує з’єднання;

5) getMarkers() отримує JSON, перетворює його на масив, та передає цей масив у змінну Markers;

6) Функція-контролер App забирає цю змінну та передає функції BodyBlock();

Вищевказані функції обробляють кожен елемент масиву Markers, збирають з них дані та використовують для відображення інформації.

Для тестування обрані етапи 2-5, оскільки лише на цих етапах виконується обробка даних з серверу. У кінці цієї послідовності ми маємо масив даних, що зберігаються в БД.

Створимо таблицю рішень на основі цієї послідовності. В якості умов буде вибрано наступні обмеження:

- Підключення до серверу встановлене
- Дані присутні на сервері
- Дані передані до клієнту
- Дані відповідають вимогам системи

Події будуть наступні:

- Виконання запиту
- Сбір результату у JSON
- Передача до клієнту
- Обробка даних

Додатково у події додані обробники помилок

Таблиця 6.6 – Функціональна діаграма рішень

	Назва	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Умова	Підключення встановлене	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	Дані присутні	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	Дані передані	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	Дані відповідають вимогам системи	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Подія	Виконання запиту									1	1	1	1				
	Сбір результату у JSON													1	1		
	Передача до клієнту															1	
	Обробка даних																1
Помилки	«No connection»	1	1	1	1	1	1	1	1								
	«No results»									1	1	1	1				
	«Response Error»													1	1		
	«Invalid Data»															1	

Обробка помилок буде виконуватись, якщо результат не буде вдалим.

6.9 Тестування методом симуляції дій користувача

Будуємо тестові набори. Кожна строка – це тест.

Таблиця 6.7 – Таблиця тестів

Тест 1	Якщо система виконує запит до серверу про користувача, то результат буде відповідати необхідному шаблону

Продовження таблиці 6.7

Тест2	Якщо відвідувач забажав зайти до системи, але ввів неправильні дані, то система виведе повідомлення «Такого користувача не існує»
Тест3	Якщо відвідувач забажав зайти до системи, та ввів правильні дані, то система отримає повну інформацію про користувача

Програмна реалізація тестів була оформлена за допомогою інструментів тестового компоненту React Jest-TS. Для проведення тестування був створений файл BodyBlock.test.ts, в якому прописані методи тестування описаних раніше прецедентів. Для перевірки даних були створені тестові блоки інформації (див. Додаток А).

expectedBuf – описує формат даних, котрий необхідний для роботи системи та відображення постів.

nullUser – описує формат даних «нульового користувача» – користувача, що не має прав у системі, а лише може авторизуватися/зареєструватися.

goodUser – описує формат даних реально існуючого користувача.

Файл зберігає у собі необхідні для тестування контрольні параметри та методи для виклику компонентів, що необхідно протестувати. Під час старту тесту Jest викликає методи getMarkers() та loginFunction().

Для першої функції тестовий модуль перевіряє, що отримані дані мають необхідний для системи формат, який описано параметром expectBuf.

Для другої функції тестовий модуль перевіряє, що саме повертає система у разі передачі правильних та помилкових даних про користувача. Контрольні дані описані параметрами nullUser та goodUser. Очікується, що у разі передачі правильного логіну та паролю система поверне дані, що знаходяться в goodUser. Скріншоти процесу тестування наведені у Додатку Б. Нижче (рис. 7.1) приведений результат тестування:

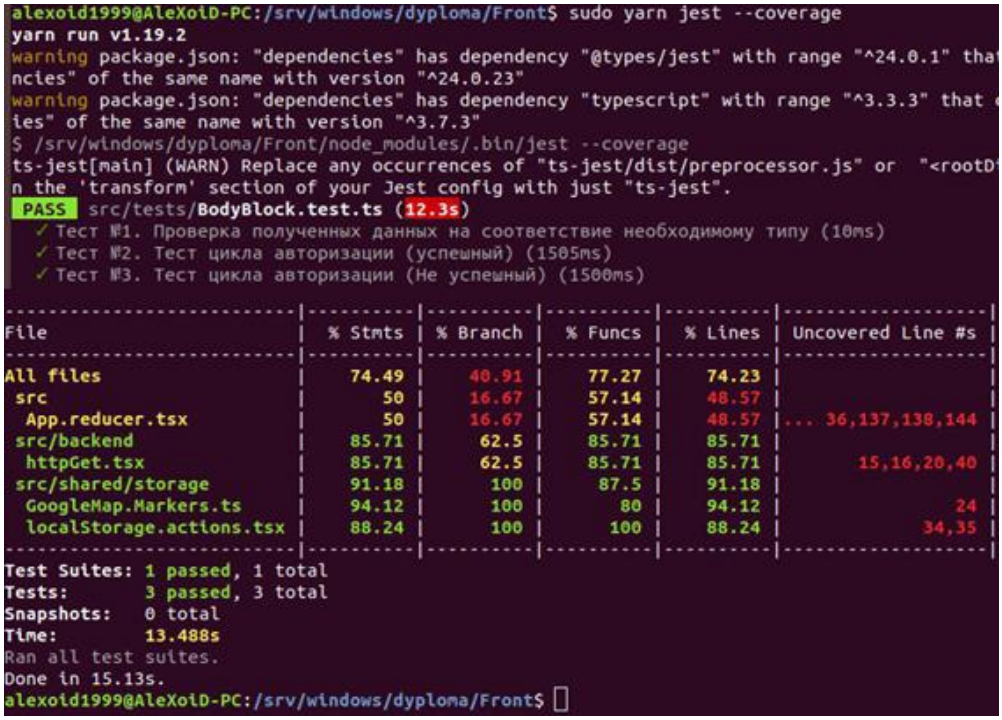

Рисунок 6.3 – Результат тестування програмного модулю

Як видно з результатів тестування, відклик серверу достатньо швидкий (10мс). Разом із повним циклом авторизації (запит, перевірка БД, отримання результату, відправка до клієнту) займає 1.5 секунди. При цьому було задіяно 74.49% всього коду файлів клієнта, залишок – функції іншого призначення. Система була успішно протестована засобами автоматичного тестування.

6.10 Тестування моделі нейронної мережі

Під тестуванням моделі розуміється перевірка якості розпізнання образів на фотографіях. Для тесту були обрані фотографії, що максимально схожі за очікуємою якістю та змістом на ті, що будуть викладатися користувачами.

Демонстрація роботи скрипту наведена на рис. 6.4. Результат розпізнавання був накладений на самі фотографії та приведений у Додатку Б.

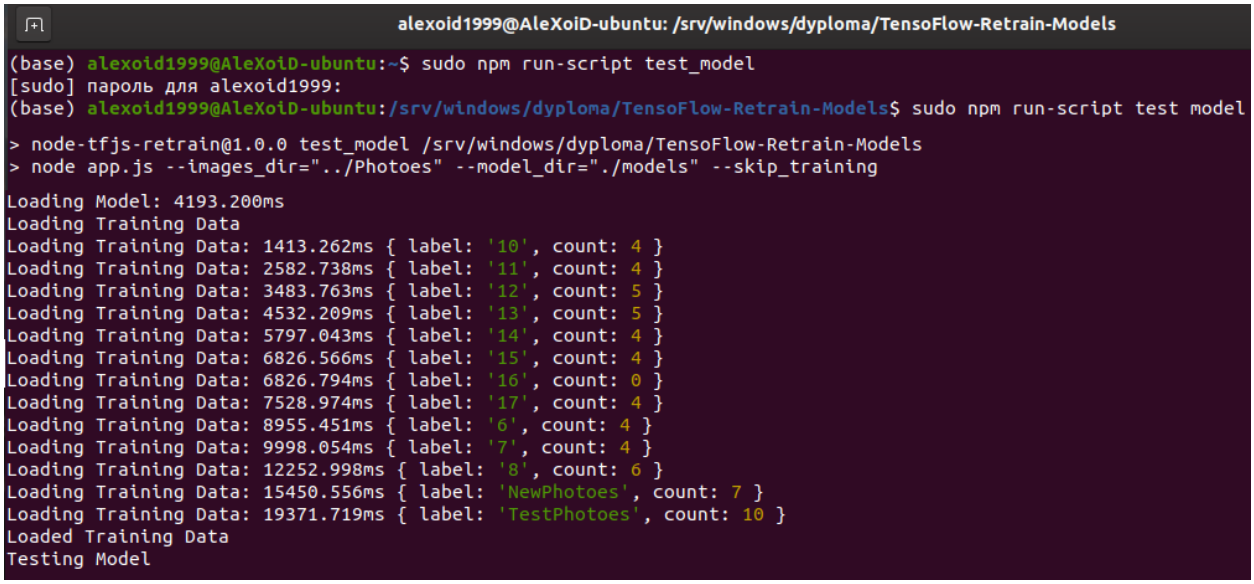

Рисунок 6.4 – Лістинг роботи скрипту з тестування моделі

Скрипт налаштований на автоматичне зчитування фотографій з усіх папок, після чого починається аналіз, конвертація у згорткову матрицю та розпізнавання образів.

ВИСНОВКИ

Дана кваліфікаційна робота була розроблена з метою підвищення конверсії серед місць, що обділені увагою серед туристів та місцевих жителів шляхом створення веб-сервісу, соціальної мережі, інформація в якій буде поповнюватися самими користувачами.

При проектуванні веб-сервісу була визначена мета та актуальність роботи, проведений опис системи та аналіз аналогів. Також були визначені вимоги нефункціональні (у вигляді сценаріїв якості) та функціональні (у вигляді сценаріїв використання).

Після проектування була описана архітектура проекту з описом кожного елементу та особливостями використаних методик. Описана схема логічного представлення даних у системі у вигляді ER-діаграми, а також 11 таблиць бази даних. Були створені прототипи графічного інтерфейсу користувача, а потім ці прототипи були реалізовані відповідними front-end та back-end технологіями. Основний функціонал програмних модулів був реалізований на мові програмування TypeScript та MySQL разом з додатковими бібліотеками та фреймворками. Також були описані особливості та методи розробки нейронної мережі.

Останнім етапом кваліфікаційної роботи було проведення тестування, а саме тестування компоненту головної сторінки методами побудови діаграми причинно-наслідкових зв’язків, методами автоматичного тестування, тестування роботи нейронної мережі а також перевірка ефективності методом соціального опитування, котре показало, що система вже на початку своєї роботи підвищила потенційну конверсію серед користувачів на 30% у порівнянні з аналогами. Результати тестування приведені в додатках.

Розроблена соціальна мережа є першим кроком для досягнення моєї цілі підвищити конверсію, покращити географічну адаптацію та знання міста серед жителів міста та туристів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. БЕМ.Інфо – Методологія БЕМ [Electronic resource] – Режим доступу: https://ru.bem.info/methodology/

2. Вікіпедія – Путівник [Electronic resource] – Режим доступу: https://uk.wikipedia.org/wiki/Путівник, 03.09.2019;

3. Rookee – Что такое Google Maps [Electronic resource] – Режим доступу: https://wiki.rookee.ru/google-maps/, 02.05.2020;

4. Вікіпедія – 2ГИС [Electronic resource] – Режим доступу: https://ru.wikipedia.org/wiki/2%D0%93%D0%98%D0%A1, 02.05.2020

5. vc.ru – Стартап дня: мобильное приложение для определения местоположения Zenly – [Electronic resource] – Режим доступу: https://vc.ru/flood/32172-startap-dnya-mobilnoe-prilozhenie-dlya-opredeleniya-mestopolozheniya-zenly, 03.05.2020;

6. Вікіпедія – Instagram [Electronic resource] – Режим доступу: https://ru.wikipedia.org/wiki/Instagram, 04.05.2020;

7. Hostinger уроки – Что такое блог? [Electronic resource] – Режим доступу: https://www.hostinger.com.ua/rukovodstva/chto-takoje-blog/, 04.05.2020;

8. Любченко В.В. Методичні вказівки до розрахунково-графічної роботи з дисципліни «Менеджмент проектів програмного забезпечення» 6.050103 – «Програмна інженерія» Одеса; ОНПУ, 2014. - 4 с.

9. Buklib.net – Складові системи планування проекту [Electronic resource] – Режим доступу: https://buklib.net/books/23851/, 06.12.2019;

10. Wikipedia [Electronic resource] – Режим доступу: https://en.wikipedia.org/wiki/JavaScript, 19.10.2019;

11. W3C [Electronic resource] – Режим доступу: https://www.w3.org/standards/webdesign/htmlcss.html, 20.10.2019;

12. W3Schools – JSON [Electronic resource] – Режим доступу: https://www.w3schools.com/js/js_json_intro.asp, 10.01.2020;

13. W3Schools – What is React [Electronic resource] – Режим доступу: https://www.w3schools.com/whatis/whatis_react.asp, 10.01.2020;

14. W3Schools – Node.JS tutorial [Electronic resource] – Режим доступу: https://www.w3schools.com/nodejs/default.asp, 18.01.2020;

15. Yarn – Getting started [Electronic resource] – Режим доступу: https://yarnpkg.com/ru/docs/getting-started, 19.01.2020;

16. TensorFlow [Electronic resource] – Режим доступу: https://www.tensorflow.org/, 26.09.2021

17. Wikipedia – MySQL [Electronic resource] – Режим доступу: https://en.wikipedia.org/wiki/MySQL, 15.03.2020;

18. Зіноватна С. Л. Методичні вказівки до курсового проектування з дисципліни «Бази даних» для студентів напряму 6.050103 – «Програмна інженерія» Одеса; ОНПУ, 2014. - 18 с.

19. Конноллі Т. Бази даних. Проектування, реалізація і супровід. Теорія і практика. 3-є видання. : Вільямс, 2003. – 1440 с.

20. Крьонке Д. Теорія і практика побудови баз даних, 8-е видання. СПб.: Питер, 2003. – 800 с.

21. Крісілов, В.А., Писаренко, К.О. і Хуї, В.Н. 2019. Метод динамічного формування контенту в умовах обмежених ресурсів. *Прикладні аспекти інформаційних технологій*. 2, 2 (Квіт 2019), 89-104. DOI [Electronic resource] – Режим доступу: http://doi.org/10.15276/aait.02.2019.1

22. Комлева, Н.О., Любченко, В.В. і Зіноватна, С.Л. 2020. Методологія інформаційного моніторингу та діагностики об’єктів, представлених кількісними оцінками, з використанням кластерного аналізу. *Прикладні аспекти інформаційних технологій*. 3, 1 (Квіт 2020), 376–392. DOI [Electronic resource] – Режим доступу: http://doi.org/10.15276/aait.01.2020.1

23. Кунгурцев О.Б. Стандарти та керування якістю програмного забезпечення, Конспект лекцій для студентів напряму 6.050103 – «Програмна інженерія» Одеса; ОНПУ, 2021. – 115с.