Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмного забезпечення для соціальної мережі з функцією відображення геолокації користувачів»

Виконав: студент __4__ курсу
групи ___1ПІ-20б___ спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Кондратюк О.О.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Романюк О.В.
(прізвище та ініціали)
Рецензент:. к.т.н., доц. каф ЗІ Войтович О.П.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ
д.т.н., професор Романюк О.Н.
«_10_» __червня__ 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
«12» березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Кондратюку Олександру Олександровичу

1. Тема роботи – розробка програмного забезпечення для соціальної мережі з функцією відображення геолокації користувачів.

Керівник роботи: Романюк Оксана Володимирівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. №80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи: тип програми – мобільний застосунок; модель розробки – ітеративна; засоби організації даних програми – фреймворк Spring та бібліотека Retrofit; вхідні дані: запит користувача, дані про місцезнаходження; середовище розробки – Intellij IDEA; мова програмування – Java;

4. Зміст текстової частини: вступ; обґрунтування вибору методу розробки та постановка задач дослідження; розробка моделі, методу та алгоритмів роботи програмного продукту; розробка програмного забезпечення; тестування програми; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу: актуальність теми; можливості, що відкриваються; мета, об'єкт та предмет дослідження; задачі дослідження; новизна і практична цінність отриманих результатів; порівняльний аналіз аналогів; діаграма сутностей; алгоритм відслідковування геолокації користувачів; метод і алгоритм формування стрічки рекомендацій користувача; технологічний стек; екран реєстрації застосунку; екран автентифікації застосунку; апробація та публікація результатів роботи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Романюк О. В., к.т.н., доцент кафедри ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

Ч.ч.	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану мобільних систем з використанням засобів для відстеження геолокації	14.03.24 – 15.03.24	виконано
2	Розробка моделі системи	21.03.24 – 31.03.24	виконано
3	Розробка методу формування стрічки рекомендацій користувача	31.03.24 – 05.04.24	виконано
4	Розробка алгоритму відслідковування геолокації користувачів	05.04.24 – 10.04.24	виконано
5	Розробка дизайну інтерфейсу програмного застосунку	15.04.24 – 30.04.24	виконано
6	Розробка програмного забезпечення	30.04.24 – 12.05.24	виконано
7	Тестування програмного забезпечення	12.05.24 – 19.05.24	виконано
8	Оформлення матеріалів до захисту БДР	19.05.24 – 30.05.24	виконано

Студент _____ **О. О. Кондратюк**
(підпис) (ініціали та прізвище)

Керівник бакалаврської дипломної роботи _____ **О. В. Романюк**
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 58 сторінок формату А4, на яких є 25 рисунків, 4 таблиці, список використаних джерел містить 23 найменування.

У бакалаврській дипломній роботі було визначено доцільність, практичну цінність та мету розробки, яка полягає в покращенні користувацького досвіду взаємодії з соціальною мережею за рахунок розробки програмного забезпечення для соціальної мережі з функцією для відображення місцезнаходження людей поруч з користувачем.

Запропоновано метод формування стрічки рекомендацій за допомогою штучного інтелекту. Розроблено алгоритм відслідковування геолокації користувачів.

Реалізація програмного засобу виконана з використанням мови програмування Java, фреймворк Spring та бібліотека Retrofit. Для забезпечення функціональності програми та взаємодії з користувачами використані засоби взаємодії з GPS, що дозволяють обробляти дані про місцезнаходження користувачів та відображати їх для інших користувачів. У результаті виконання бакалаврської дипломної роботи розроблено програмний засіб, працездатність і правильність роботи якого перевірено, підготовлена інструкція користувача та визначено системні вимоги.

Отримані в бакалаврській роботі результати можна використати для відстеження місцезнаходження людей поруч з користувачем та комунікації між користувачами.

Ключові слова: геолокація, GPS, соціальна мережа, користувачі, місцезнаходження, відстеження.

ABSTRACT

The bachelor's thesis consists of 58 A4 pages, which have 25 figures, 4 tables, the list of sources used contains 23 items.

The bachelor's thesis identified the feasibility, practical value, and purpose of development, which consists of improving the user experience of interacting with a social network by developing software for a social network with a feature for displaying the location of people nearby.

A method for generating a recommendation feed using artificial intelligence is proposed. An algorithm for tracking user geolocation is developed.

The software implementation was done using the Java programming language, Spring framework, and Retrofit library. To ensure the functionality of the program and interaction with users, tools for interacting with GPS were used, allowing processing user location data and displaying them for other users. As a result of the bachelor's thesis, a software tool has been developed, the performance and correctness of which have been verified, a user manual has been prepared, and system requirements have been defined.

The results obtained in the bachelor's thesis can be used for tracking the location of people nearby and communication between users.

Keywords: geolocation, GPS, social network, users, location, tracking.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному цифровому світі соціальні мережі відіграють важливу роль у спілкуванні та взаємодії між людьми. З кожним днем їх вплив на наше життя лише зростає, надаючи нові можливості для спілкування, обміну інформацією та координації дій. У цьому контексті актуальною стає розробка програмних засобів, які розширюють можливості соціальних мереж та забезпечують користувачів новими функціями.

Однією з таких потенційно цікавих функцій є відображення місцезнаходження людей поруч з користувачем. Це може стати корисним інструментом для планування зустрічей, організації подій або просто для того, щоб знати, де перебувають ваші друзі чи колеги.

Процес розробки такого програмного засобу передбачає вивчення потреб користувачів, вибір оптимальних технологій та розробку ефективного інтерфейсу. Крім того, важливо врахувати питання конфіденційності та безпеки даних, щоб забезпечити комфортне використання сервісу.

Програмне забезпечення для соціальної мережі з функцією відображення місцезнаходження людей поруч з користувачем надає такі можливості як:

1. Покращення взаємодії користувачів: Забезпечення можливості відображення місцезнаходження інших користувачів поруч з поточним користувачем дозволить покращити комунікацію та взаємодію між ними. Це особливо важливо для організації зустрічей, подій чи спільних заходів.

2. Підвищення безпеки та контролю: Можливість відслідковувати місцезнаходження друзів чи родичів може бути корисною для забезпечення їх безпеки та контролю за їхнім розташуванням, особливо у випадку екстрених ситуацій.

3. Створення нових сервісів та функцій: Розробка програмного засобу для цієї функції відкриває широкі можливості для створення нових сервісів та функцій в соціальних мережах, які можуть бути корисними для різних груп користувачів, включаючи бізнес, туризм, освіту тощо.

Тому актуальною є розробка програмного засобу для реалізації в соціальній мережі функції відображення місцезнаходження людей поруч з користувачем.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно з планом виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є покращення користувацького досвіду взаємодії з соціальною мережею за рахунок розробки програмного забезпечення для соціальної мережі з функцією для відображення геолокації людей поруч з користувачем.

Відповідно до поставленої мети потрібно вирішити такі **завдання**:

- здійснити аналіз стану питання та методів розв'язання задачі;
- спроектувати модель мобільного застосунку;
- розробити метод формування стрічки рекомендацій користувача;
- розробити алгоритм відслідковування геолокації користувачів та загальний алгоритм роботи програми;
- розробити базу даних, яка буде забезпечувати збереження інформації про користувачів та їх місцезнаходження.
- розробити графічний інтерфейс мобільного застосунку;
- розробити модуль для відображення місцезнаходження, який буде отримувати дані про місцезнаходження користувачів і відображати їх на графічному інтерфейсі;
- розробити модуль соціальної мережі;
- провести тестування програмного застосунку;
- розробити інструкцію користувача та описати системні вимоги програми для відстеження читання.

Об'єктом дослідження є процес розробки програмних модулів для програмного забезпечення для соціальної мережі з функцією відображення місцезнаходження людей поруч з користувачем.

Предметом дослідження є методи і засоби реалізації програмного забезпечення для реалізації в соціальній мережі функції відображення геолокації

людей поруч з користувачем.

Методи дослідження. У процесі дослідження були використані такі методи як: порівняльний аналіз аналогів, для виявлення недоліків та постановки задач для їх вирішення; теорія алгоритмів для розробки алгоритмів роботи програмного засобу; комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Новизна отриманих результатів.

1. Удосконалено систему відслідковування геолокації користувачів, яка, на відміну від відомих, реалізована у вигляді окремого модуля на окремому сервері, що дало можливість підвищити гнучкість її інтеграції, швидкість роботи та користувацький досвід.

2. Уперше запропоновано метод формування стрічки рекомендацій користувача, особливістю якого є використання штучного інтелекту, що дало можливість підвищити релевантність рекомендацій відповідно до інтересів і потреб користувачів.

Практична цінність отриманих результатів полягає в тому, щоб на основі отриманих в ході виконання бакалаврської дипломної роботи теоретичних положень розроблено алгоритми і програмне забезпечення для соціальної мережі з функцією відображення місцезнаходження людей поруч з користувачем.

Особистий внесок здобувача. Усі наукові результати, що викладені у бакалаврській дипломній роботі, отримано автором самостійно. У наукових працях, опублікованих у співавторстві, автору належать такі результати: дослідження різних API для відстеження місцезнаходження пристрою [1]; розробка системи для інтеграції в соціальній мережі функціоналу відстеження місцезнаходження [2].

Апробація матеріалів бакалаврської дипломної роботи. Результати роботи доповідалися на:

- LIII Всеукраїнській науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії 20-22 березня 2024 р. (м. Вінниця, 2024 р.) [1];

– Молодь в науці: дослідження, проблеми, перспективи (м. Вінниця 2024 р.) [2].

Публікації. За тематикою дослідження опубліковано 2 наукові праці у збірниках матеріалів конференцій.

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану мобільних систем з використанням засобів для відстеження місцезнаходження

З розвитком технологій та зростанням популярності мобільних застосунків, розробка програмних засобів для реалізації в соціальній мережі функції відображення місцезнаходження людей поруч з користувачем стає все більш актуальною. Однією з ключових переваг розробки мобільного застосунку для цієї функціональності є його доступність та зручність використання для користувачів, які можуть швидко та з легкістю відстежувати місцезнаходження своїх друзів або контактів у реальному часі.

Мобільний застосунок дозволить користувачам мережі відображати місцезнаходження інших користувачів, з якими вони мають спільний доступ до цієї інформації. Це може бути корисно для планування зустрічей, визначення оптимального місця зустрічі або просто для відстеження руху та місцезнаходження близьких людей. Водночас, важливо забезпечити високий рівень захисту особистої інформації та конфіденційності даних користувачів.

Розробка програмного забезпечення для мобільних платформ з такою функціональністю має включати в себе можливості з відображення місцезнаходження користувачів на мапі, налаштування приватності та доступу до цієї інформації, а також інструменти для взаємодії з цими даними, наприклад, можливість надсилати повідомлення або запрошення на зустріч.

Ефективність та стабільність роботи застосунку в умовах великого навантаження вимагають ретельного проєктування та оптимізації програмного забезпечення. Використання потужних серверних ресурсів та технологій масштабування може допомогти забезпечити плавну роботу застосунку при будь-якій кількості активних користувачів.

Однією з ключових вимог до розробки такого застосунку є забезпечення його стабільної роботи та ефективності в умовах високого навантаження,

особливо у великих мережах з великою кількістю активних користувачів. Також важливою є можливість інтеграції з іншими соціальними мережами або сервісами, щоб розширити функціональність та зручність використання застосунку.

Отже, Розробка програмного забезпечення для соціальної мережі з функцією відображення місцезнаходження людей поруч з користувачем є важливим завданням, що вимагає комплексного підходу та уваги до деталей з метою забезпечення якості обслуговування та задоволення потреб користувачів.

1.2 Порівняльний аналіз аналогів

У сучасному ритмі життя, де кожен день приносить нові виклики та можливості, збереження зв'язку з рідними та друзями стало дійсно надзвичайно важливим. Застосунки для відстеження місцезнаходження, що пропонуються на ринку, стають надійним рішенням для цієї потреби. Вони працюють у режимі реального часу, надаючи можливість користувачу отримувати актуальну інформацію про місцезнаходження близьких, що забезпечує спокій та безпеку.

Окрім забезпечення безпеки, ці застосунки також допомагають у плануванні зустрічей та організації подій. Вони забезпечують зручний спосіб відстеження місцезнаходження друзів, колег або членів сім'ї, що полегшує координацію та спілкування. Наприклад, користувач може швидко визначити оптимальне місце зустрічі або відстежувати переміщення групи людей під час подорожі або події. Ці застосунки надають користувачу оновлення у реальному часі щодо місцезнаходження близьких, забезпечуючи спокій та полегшуючи координацію [3]. Також вони є корисним інструмент для планування зустрічей та організації подій, а також для відстеження місцезнаходження друзів чи колег. До найбільш відомих рішень відносяться:

- Life360;
- Glympse;
- Foursquare Swarm.

Life360, в основному, дозволяє вам слідкувати за близькими людьми цілодобово. Застосунок дозволяє створити мережу родини та друзів будь-якого

віку. Сервіс дозволяє батькам бачити місцезнаходження своїх дітей в будь-який час. Також це може бути корисним для людей що піклуються про людей похилого віку. Крім того, ви можете використовувати застосунок, щоб визначити місцезнаходження людини, а також отримувати сповіщення, якщо автомобіль в якому вони перебувають перевищує швидкісний ліміт. Також він може надсилати вам сповіщення, якщо особа, яку ви відстежуєте, виходить за визначену область (відома як геозонування), і допомагає вам знайти їх, якщо вони заблукають [4]. Приклад роботи Life360 зображено на рисунку 1.1.

Рисунок 1.1 – Приклад роботи Life360

Glympse – це застосунок для iPhone, Android і Blackberry, який дозволяє легко та безпечно ділитися своєю локацією з іншими в реальному часі. Він надає можливість надіслати свою локацію через електронну пошту, SMS, Facebook або Twitter. Glympse використовує можливості GPS у мобільному телефоні, щоб надати змогу ділитися геолокацією через веб-карту з будь-ким [5]. Приклад роботи Glympse зображено на рисунку 1.2.

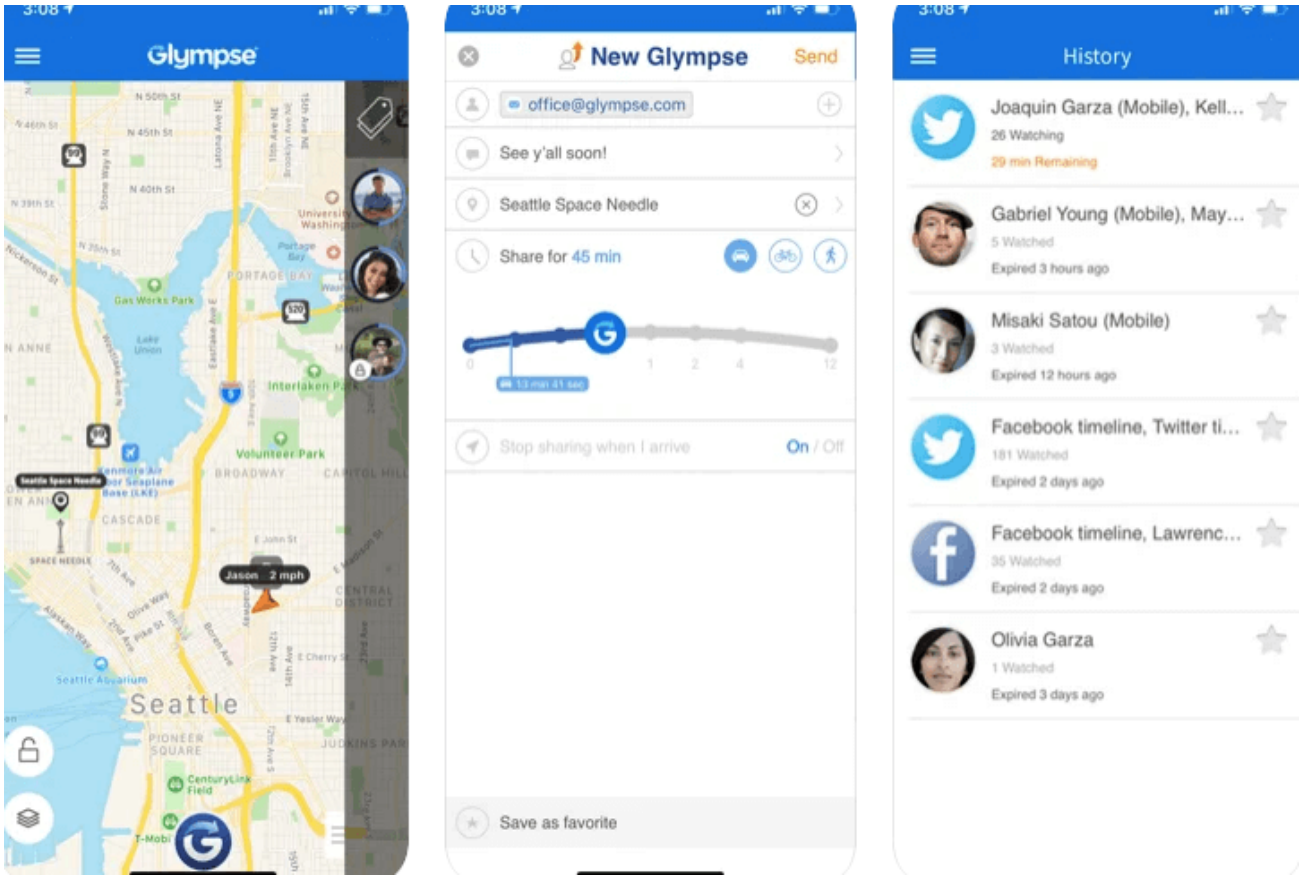

Рисунок 1.2 – Приклад роботи Glympse

Люди використовують Foursquare і Swarm з різних причин, але основні переваги включають:

– рекомендації на основі місця розташування: Foursquare надає персоналізовані рекомендації на основі місця розташування користувача і його минулих уподобань;
– планування подорожей: Foursquare і Swarm використовуються мандрівниками для знаходження популярних та маловідомих визначних місць, ресторанів та розваг у невідомих містах;
– знаходження місцевих підприємств: Користувачі покладаються на Foursquare, щоб відкривати для себе нові місцеві підприємства, такі як: кав'ярні, музеї, виставки, ресторани, тощо. Також Foursquare дозволяє читати відгуки від інших користувачів;

– гейміфікація: Swarm гейміфікує досвід відвідування різних місць, пропонуючи винагороди, наклейки та віртуальні монети за реєстрацію, додаючи елемент розваг та змагання до відкриття нових місць.

В цілому, Foursquare і Swarm популярні своїми місцевими послугами, соціальною взаємодією та персоналізованими рекомендаціями, що робить їх цінними інструментами як для місцевого дослідження, так і для планування подорожей [6]. Приклад роботи Foursquare Swarm зображено на рисунку 1.3.

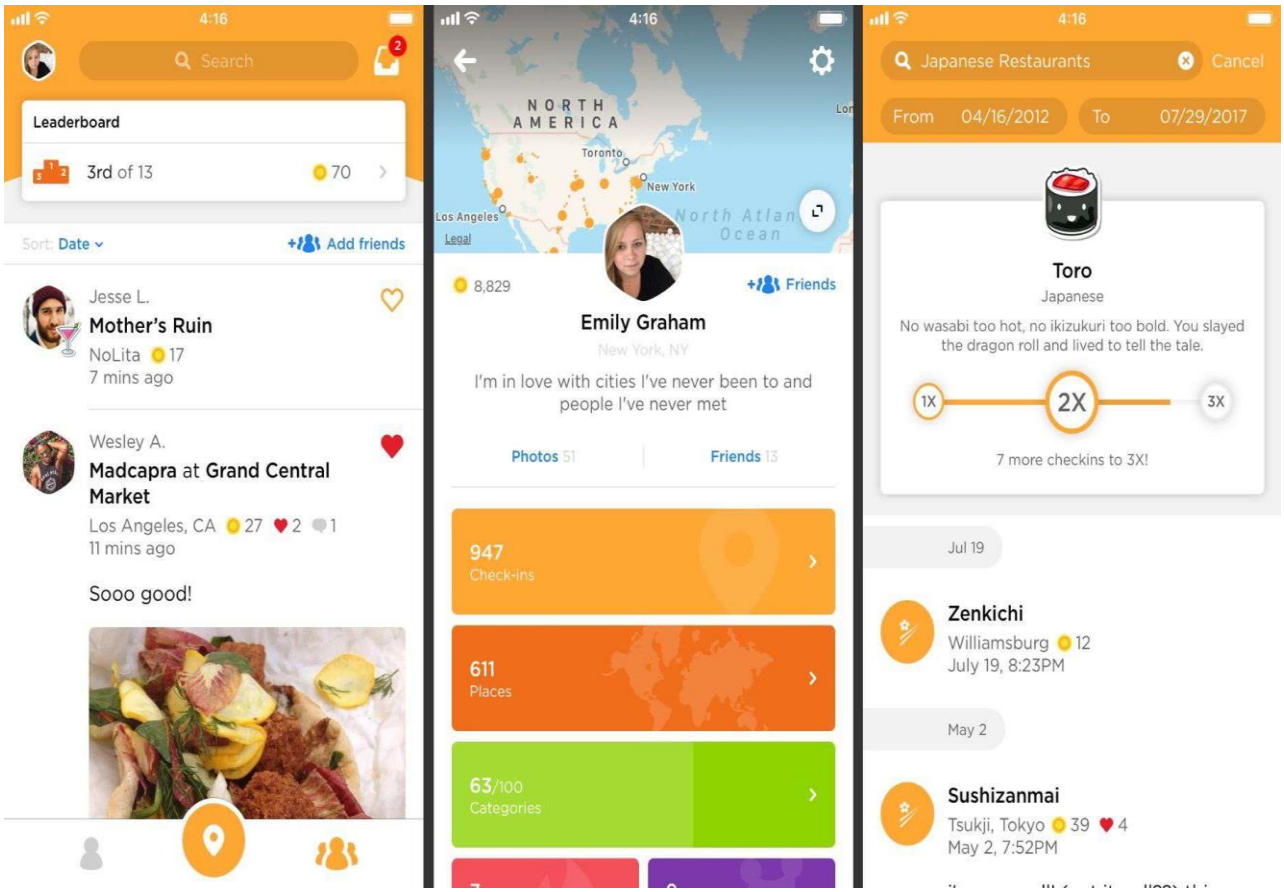

Рисунок 1.3 – Приклад роботи Foursquare Swarm

Розробка програмного забезпечення для соціальної мережі з функцією відображення геолокації користувачів є складним завданням, яке потребує глибокого розуміння потреб користувачів та вивчення їхніх пріоритетів. Щоб успішно реалізувати цю функцію, необхідно врахувати декілька важливих аспектів:

- інтеграція з GPS: Для взаємодії з користувачами та збору даних про їх місцезнаходження необхідно інтегрувати засоби GPS. Це дозволить точно визначати місцезнаходження користувачів та відображати їх на мапі з високою точністю.;
- розробка алгоритму формування рекомендацій: Для створення алгоритму формування рекомендацій користувача можна застосувати штучний інтелект. Алгоритм має враховувати індивідуальні переваги користувача, щоб надавати максимально релевантні рекомендації щодо подій, місць або інших користувачів поруч;
- захист персональних даних: Важливо розробити механізми захисту інформації для забезпечення безпеки та конфіденційності персональних даних користувачів. Це може включати шифрування даних під час передачі та зберігання. Необхідно забезпечити надійні методи автентифікації та авторизації користувачів для захисту їхніх акаунтів від несанкціонованого доступу. Також потрібно надати користувачам можливість контролювати, хто може бачити їхнє місцезнаходження,;
- ефективне використання ресурсів пристроїв: Для забезпечення оптимальної продуктивності та ефективного використання ресурсів пристроїв користувачів, необхідно оптимізувати процес відображення місцезнаходження. Це може включати кешування даних, використання мінімальної кількості ресурсів та мінімізацію впливу на енергоспоживання пристроїв;
- покращення користувацького досвіду: Важливо розробити інтуїтивно зрозумілий і зручний інтерфейс для користувачів, який дозволить їм легко взаємодіяти з функцією відображення місцезнаходження.

Загалом, виконання всіх цих кроків допоможе забезпечити ефективну та безпечну реалізацію функції відображення місцезнаходження людей поруч з користувачем у соціальній мережі.

Результати порівняння аналогів наведено в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика застосунків

Критерії	Life360	Glympse	Foursquare Swarm	Власний програмний засіб
Простота використання	+	+	-	+
Відстеження точного місцезнаходження	+	-	-	+
Функціонал характерний соціальній мережі	-	-	+	+
Доступність всього функціоналу безкоштовно	-	-	+	+
Швидкість роботи	+	-	+	+
Споживання ресурсів	-	+	+	+
Загальна оцінка	3	2	4	6

Опис критеріїв за якими проводилась порівняльна характеристика:

1.Простота використання: простота використання включає в себе зручний інтуїтивно зрозумілий інтерфейс, доступність для широкого кола людей, мінімальна кількість кроків, яку має виконати користувач для виконання поставленої задачі.

2.Відстеження точного місцезнаходження: Застосунок повинен відстежувати місцезнаходження користувачів з високою точністю, для забезпечення найкращого користувацького досвіду.

3.Функціаонал характерний соціальній мережі: застосунок повинен надавати функціонал характерний сучасним соціальним мережам, наприклад можливість створювати різноманітні дописи, додавати інших користувачів у список друзів тощо.

4.Доступність всього функціоналу безкоштовно: весь функціонал застосунку повинен бути доступний користувачеві без додаткових витрат.

5.Швидкість роботи: система повинна швидко обробляти запити користувачів для покращення користувацького досвіду.

6.Споживання ресурсів: застосунок не повинен займати багато місця на

диску, повинен продуктивно та ефективно використовувати ресурси пристрою, для мінімального споживання ресурсів.

Враховуючи дані наведені в таблиці, Розробка програмного забезпечення для соціальної мережі з функцією відображення місцезнаходження людей поруч з користувачем може бути доцільною. Новий продукт має потенціал вирішити недоліки наявних рішень, покращити досвід користувачів соціальної мережі і відкрити нові можливості для використання технології відстеження місцезнаходження.

Створення власного модулю дає змогу врахувати усі особливості та потреби користувачів, а також забезпечити більшу гнучкість у впровадженні нового функціоналу. Це дозволить підвищити задоволення користувачів від використання соціальної мережі, оскільки вони отримають доступ до нового досвіду та розширених можливостей.

Отже, розробка власного програмного забезпечення для соціальної мережі з функцією відображення місцезнаходження людей поруч з користувачем може стати перспективним кроком для покращення якості та функціональності соціальної мережі.

Також розробка програмного забезпечення для соціальної мережі з функцією відображення місцезнаходження людей поруч з користувачем може відкрити багато можливостей перед бізнесом як і в зацікавлені нових користувачів так і для максимально довгого їх утримання на платформі за рахунок функцій які надає програмне рішення та його перевагою над наявними аналогами.

1.3 Аналіз методів розв’язання задачі

Перший підхід до розробки програмних засобів для втілення функціоналу відображення місцезнаходження людей поруч з користувачем у соціальній мережі полягає в створенні окремої бібліотеки. Цей підхід передбачає розробку набору інструментів, які можна легко включити у вже існуючий код мережі. Це може спростити інтеграцію функціоналу та забезпечити однорідний досвід користувачів.

Варто зазначити, що використання окремої бібліотеки може призвести до залежностей і конфліктів у коді мережі, якщо інтеграція не здійснюється належним чином. Тому важливо враховувати всі аспекти цього підходу та ретельно планувати процес розробки та інтеграції функціоналу в мережу.

Другий підхід включає розробку окремого сервісу для інтеграції функціоналу в існуючі системи соціальних мереж. Це означає створення готового модулю, який користувачі можуть легко інтегрувати у свої мережі.

Основна перевага цього підходу полягає в тому, що він дозволяє зосередитися на розробці самого модуля, не витрачаючи зайвих зусиль на створення повномасштабної соціальної мережі. Такий модуль може включати у себе не лише функціонал відображення місцезнаходження, але й інші корисні можливості, наприклад, обмін повідомленнями або планування подій.

Плюсом є те, що цей підхід зменшує зусилля, необхідні для інтеграції, порівняно з першим підходом. Користувачам не потрібно будувати все з нуля або впроваджувати складний код, а просто підключають готовий сервіс до своєї мережі.

Обидва ці підходи мають свої переваги та недоліки, і вибір між ними повинен залежати від конкретних потреб та задач проєкту. При розгляді можливих методів розробки програмних засобів для втілення функції відображення місцезнаходження людей поруч з користувачем у соціальній мережі, варто також врахувати можливість розширення функціоналу цього модуля. Наприклад, додавання можливості вибору діапазону відстані для відображення користувачів поруч та інші покращення.

1.4 Постановка задач проєкту

Проведений аналіз порівняльних характеристик наявних рішень, було сформовано наступні задачі для успішної розробки програмних засобів для реалізації в соціальній мережі функції відображення місцезнаходження людей поруч з користувачем:

– здійснити аналіз стану питання та методів розв’язання задачі;

- спроектувати модель мобільного застосунку;
- розробити метод формування стрічки рекомендацій користувача;
- розробити алгоритм відслідковування геолокації користувачів та загальний алгоритм роботи програми;
- розробити базу даних, яка буде забезпечувати збереження інформації про користувачів та їх місцезнаходження.
- розробити графічний інтерфейс мобільного застосунку;
- розробити модуль для відображення місцезнаходження, який буде отримувати дані про місцезнаходження користувачів і відображати їх на графічному інтерфейсі;
- розробити модуль соціальної мережі;
- провести тестування програмного застосунку;
- розробити інструкцію користувача та описати системні вимоги програми для відстеження читання.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У першому розділі було розглянуто стан мобільних систем з використанням засобів для відстеження місцезнаходження. Було проведення порівняння відомих платформ таких як: Life360; Glympse; Foursquare Swarm.

У результаті проведеного дослідження було виявлено, що досліджувані платформи здобули свою популярність за рахунок зручності використання та доступності. Проаналізувавши досліджувані платформи було виявлено, що основним їх недоліком є або відсутність відстеження точного місцезнаходження, або недоступність деяких функцій в безкоштовній версії. Таким чином було доведено доцільність розробки власного програмного рішення. На основі отриманої інформації було сформовано задачі, які необхідно виконати для успішної реалізації модуля для відстеження людей поруч з користувачем.

2 РОЗРОБКА МОДЕЛІ, МЕТОДУ ТА АЛГОРИТМІВ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних даних системи

Для реалізації програмних засобів для реалізації в соціальній мережі функції відображення місцезнаходження людей поруч з користувачем будуть використані мова програмування Java, Spring Framework та бібліотека Retrofit.

Java є однією з найпопулярніших мов програмування у світі, використовуваною для широкого спектру програмних застосувань, включаючи веб-застосунки, мобільні застосування, корпоративне програмне забезпечення та багато іншого. Вона здатна працювати на будь-якій платформі, оскільки є кросплатформенною мовою програмування. Велика кількість доступних бібліотек та фреймворків робить Java вибором номер один для розробників усіх рівнів досвіду.

Мова Java славиться своєю швидкістю, безпечністю та надійністю. Об'єктно-орієнтований підхід сприяє створенню структурованих та легко розширюваних програм. Вона також надає багатий набір інструментів для роботи з великими обсягами даних та розробки серверних застосувань [7]. У контексті розробки системи для реалізації в соціальній мережі функції відображення місцезнаходження людей поруч з користувачем, Java виявляється ідеальним вибором, оскільки вона забезпечує потужність та гнучкість для створення високопродуктивних та масштабованих систем.

Spring Framework (Spring) – це відкрита платформа для розробки програмного забезпечення, яка надає широкі можливості для створення різноманітних застосунків на мові програмування Java. Цей фреймворк забезпечує високорівневу інфраструктурну підтримку, включаючи управління біном об'єктів, управління транзакціями, взаємодію з базами даних, роботу з веб-застосунками та інші необхідні функції. Одним з ключових переваг Spring є його модульність, що дозволяє розробникам використовувати тільки необхідні компоненти для своїх проектів і легко розширювати функціонал за потребою. Використання Spring

дозволяє розробникам швидко створювати ефективні, масштабовані та підтримувані Java-застосунки на будь-якій платформі розгортання.

Бібліотека Retrofit – це безпечний клієнт REST для Android, Java та Kotlin, розроблений компанією Square. За допомогою цієї бібліотеки завантаження даних у форматі JSON або XML з веб-API стає простим. У бібліотеці Retrofit, як тільки дані завантажені, вони перетворюються в об'єкт Plain Old Java Object (POJO), який повинен бути визначений для кожного "ресурсу" у відповіді. Retrofit - це проста і швидка бібліотека для отримання та відвантаження даних через веб-сервіс на основі REST [8].

Розробка програмного забезпечення для соціальної мережі з функцією відображення місцезнаходження людей поруч з користувачем передбачає кілька етапів.

Спочатку потрібно розробити основний модуль соціальної мережі який буде вміщати в собі функціонал характерний сучасним соціальним мережам.

Наступним кроком потрібно створити модуль який буде отримувати інформацію про місцезнаходження користувача використовуючи GPS. Система глобального позиціонування (GPS) – це система, яка надає користувачам послуги позиціонування, навігації та синхронізації часу [9].

Далі розробка модулю для захисту цієї інформації від потенційного перехоплення зловмисниками.

Після цього розробка модулю для передачі та збереження інформації про місцезнаходження користувача в хмарі.

Розробка цих модулів вимагає розуміння роботи системи GPS, навичок веб-програмування та розуміння принципів REST для легкого та ефективного спілкування між API [10]. Також для забезпечення безпеки даних про місцезнаходження користувачів можна використати бібліотеки для шифрування даних.

Отже, розробка програмних модулів для реалізації системи для реалізації в соціальній мережі функції відображення місцезнаходження людей поряд з користувачем є комплексним завданням, яке включає кілька етапів. Модулі

повинні бути ретельно спроектовані, реалізовані та протестовані, щоб забезпечити якісний досвід роботи з системою.

2.2 Розробка моделі системи

Для кращого розуміння того, як працюють програмні модулі у системі для відслідковування людей поряд з користувачем, було прийнято рішення створити модель системи за допомогою діаграми діяльності в мові моделювання UML (рис. 2.1). UML-діаграма – це ефективний спосіб візуалізації систем і програмного забезпечення. Програмні інженери використовують діаграми UML для кращого розуміння дизайну, архітектури коду та потенційної реалізації складних програмних систем. Діаграма діяльності UML дозволяє візуалізувати послідовність дій та процеси, які відбуваються в системі. Це допомагає краще зрозуміти логіку роботи програмних модулів і їх взаємодію.

На діаграмі можуть бути показані такі елементи системи, як користувачі, модулі відслідковування, обробники даних та інші компоненти. Вона відображає послідовність дій, що відбуваються під час взаємодії користувача з системою, а також внутрішні процеси системи.

Створення діаграми діяльності в мові UML дозволяє здійснити аналіз роботи системи на високому рівні абстракції, що сприяє кращому розумінню її функціональності та взаємодії компонентів. Такий підхід допомагає покращити процес розробки та забезпечити оптимальну архітектуру системи для відслідковування людей поряд з користувачем. Крім того, діаграми UML застосовуються для моделювання робочих та бізнес-процесів [11].

Згідно з цією діаграмою, користувач спочатку має автентифікуватися, щоб отримати доступ до функціоналу системи. Якщо користувач ввів коректні дані автентифікація вважається успішною. Після успішної автентифікації застосунок запитує дозвіл на відстеження точної геолокації. Якщо користувач надав цей доступ, то відображається екран, на якому представлені місцезнаходження людей, які знаходяться поряд з ним, якщо ж ні, то система не виконує ніяких дій. Така

послідовність дій забезпечує зручний та ефективний спосіб використання системи для користувача.

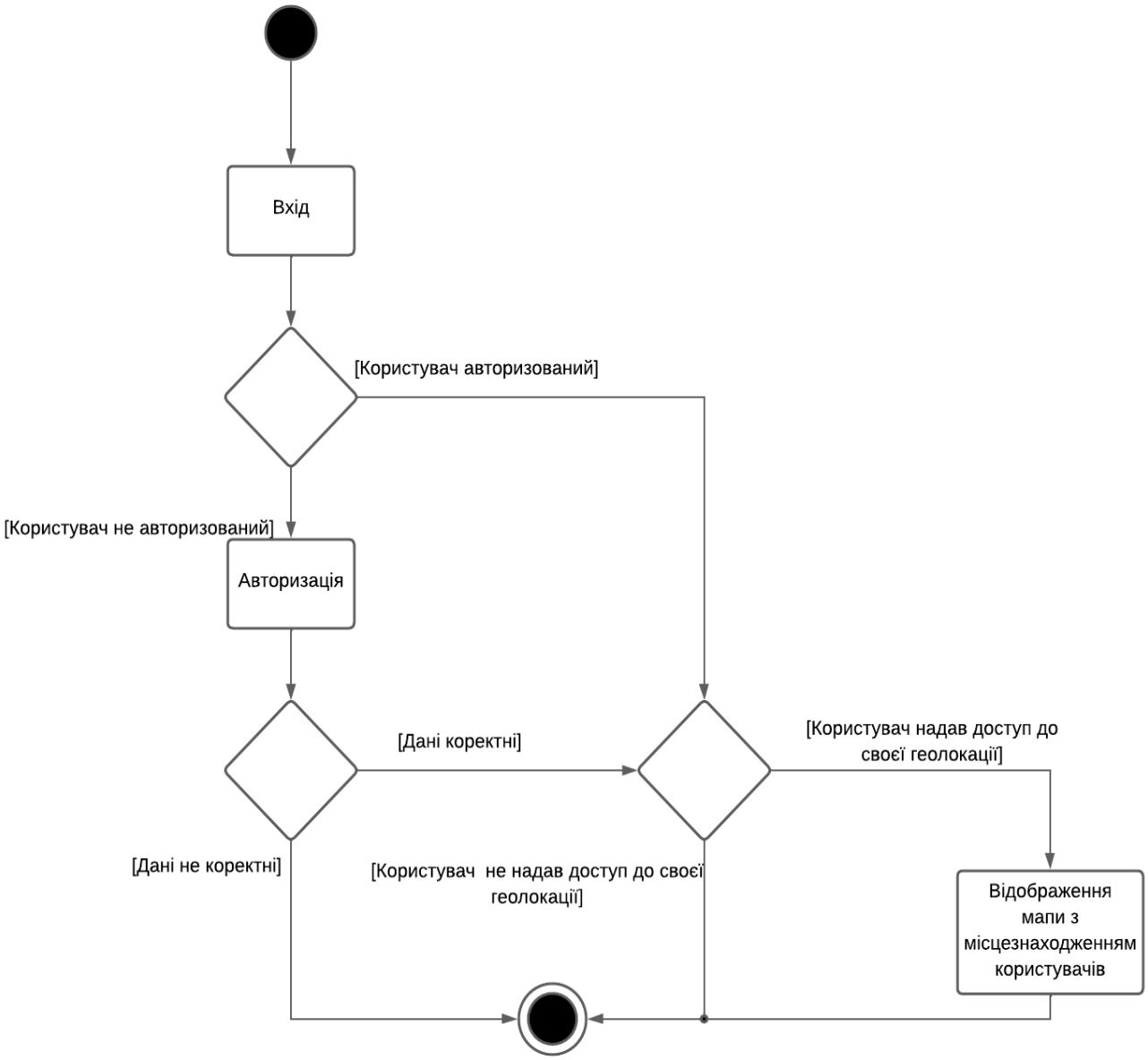

Рисунок 2.1 – Діаграма роботи системи відстеження місцезнаходження

Розроблений на основі цієї діаграми програмний модуль забезпечить ефективну роботу системи для відслідковування геолокації користувачів.

2.3 Розробка методу формування стрічки рекомендацій користувача

Розробка методу формування рекомендацій користувача за допомогою штучного інтелекту має на меті можливість підвищити релевантність

рекомендацій відповідно до інтересів і потреб користувачів, за рахунок того, щоб надавати штучному інтелекту інформацію про вподобання користувача зібрані на основі його інтересів, та про всі дописи, що з’явилися за останній час.

Було розроблено алгоритм формування стрічки рекомендацій для користувача за допомогою на основі його інтересів. Блок-схема алгоритму продемонстрована на рисунку 2.2.

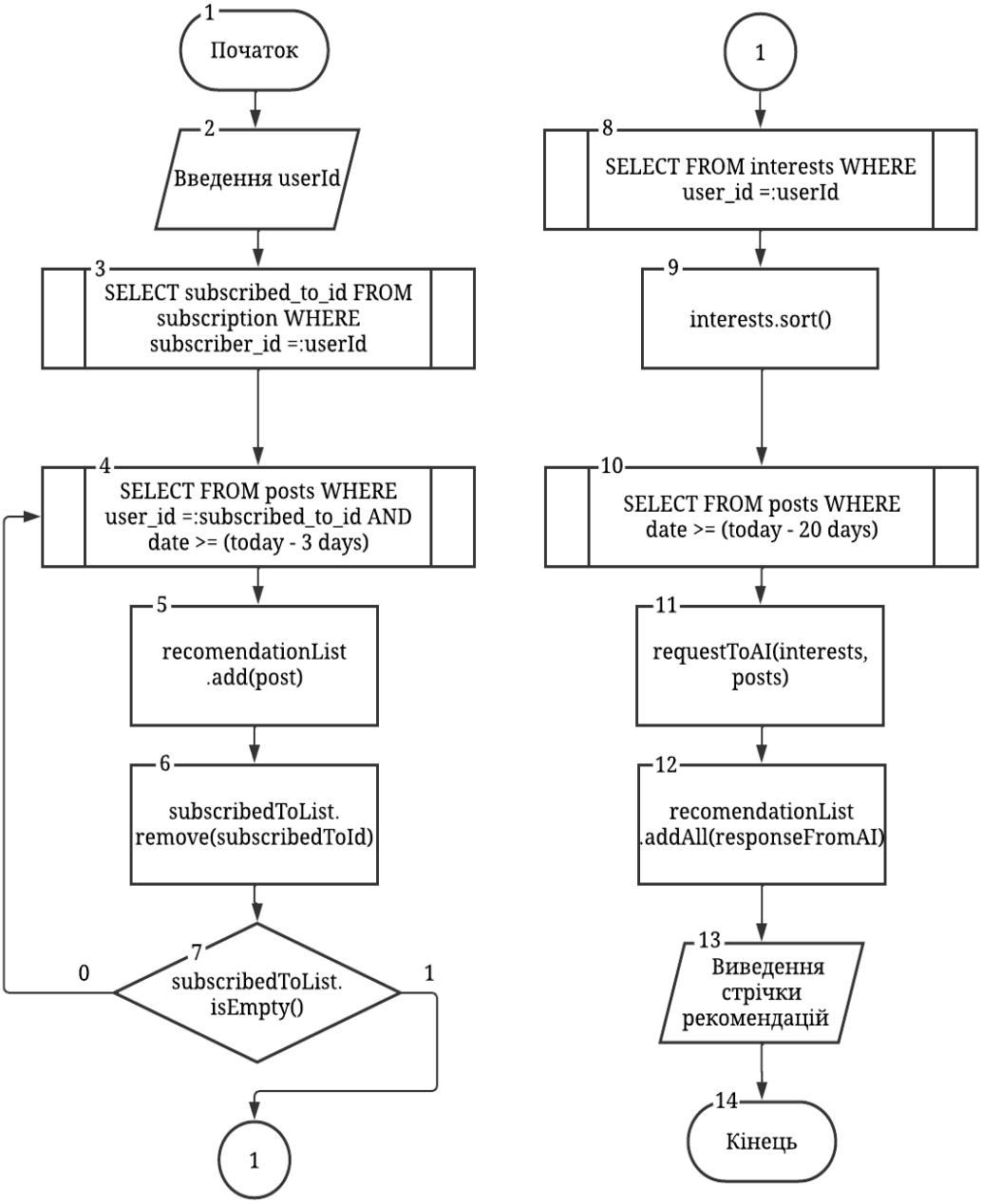

Рисунок 2.2 – Блок-схема алгориту формування стрічки рекомендацій користувача

Згідно з алгоритмом спочатку система отримує унікальний ідентифікатор користувача в базі даних.

Після цього формується запит в базу даних на отримання списку ідентифікаторів користувачів на яких підписаний користувач.

Після цього з бази даних отримується список дописів користувачів за останніх 3 дні, цей процес повторюється для кожного користувача на якого підписаний користувач, що зробив запит.

Наступним кроком є формування списку дописів, що можуть зацікавити користувача. Для цього в базі даних є таблиця, що зберігає інформацію про інтереси користувача, отримавши інформацію з неї потрібно відсортувати цю інформацію за тим, що найбільш цікаве користувачу на даний момент.

Після цього з бази даних отримується інформація про всі дописи створені за останні 20 днів. Це потрібно для того, щоб зібрану інформацію передати на обробку штучному інтелекту, що сформує список рекомендацій, що найкраще підходять користувачеві. Після того, як було отримано відповідь від штучного інтелекту фінальним кроком є відображення списку рекомендацій користувачеві.

Розроблений метод формування стрічки рекомендацій користувача, особливістю якого є використання штучного інтелекту, надав можливість більш точно відповідати інтересам користувачів.

2.4 Розробка алгоритму відслідковування геолокації користувачів

Для розробки системи відслідковування людей поряд з користувачем необхідно створити загальний алгоритм, який буде керувати роботою модулю застосунку [12] (рис. 2.3). Згідно з діаграмою, перший крок – це автентифікація користувача, щоб вони могли увійти до системи і скористатися її функціоналом.

Після успішної автентифікації система перевіряє, чи надав користувач доступ до своєї геолокації. Це важливий крок, оскільки для ефективної роботи застосунку необхідно мати доступ до географічних даних користувача.

Якщо доступ до геолокації надано, користувач може перейти на екран з мапою, на якій показані місцезнаходження інших користувачів. Це дозволяє

користувачеві бачити, де знаходяться його друзі або родичі в реальному часі. Одночасно з передачею місцезнаходження користувача у хмару для зберігання, інші користувачі можуть запитати цю інформацію з хмари. Усі дані, які передаються між мобільним застосунком та хмаровим сервісом, підлягають шифруванню для забезпечення безпеки та конфіденційності.

Це важливий аспект роботи системи, оскільки забезпечується захист особистої інформації користувачів від нелегітимного доступу та зловживання. Шифрування даних забезпечує надійний захист від можливих загроз і дозволяє користувачам використовувати систему з впевненістю в безпеці своєї особистої інформації.

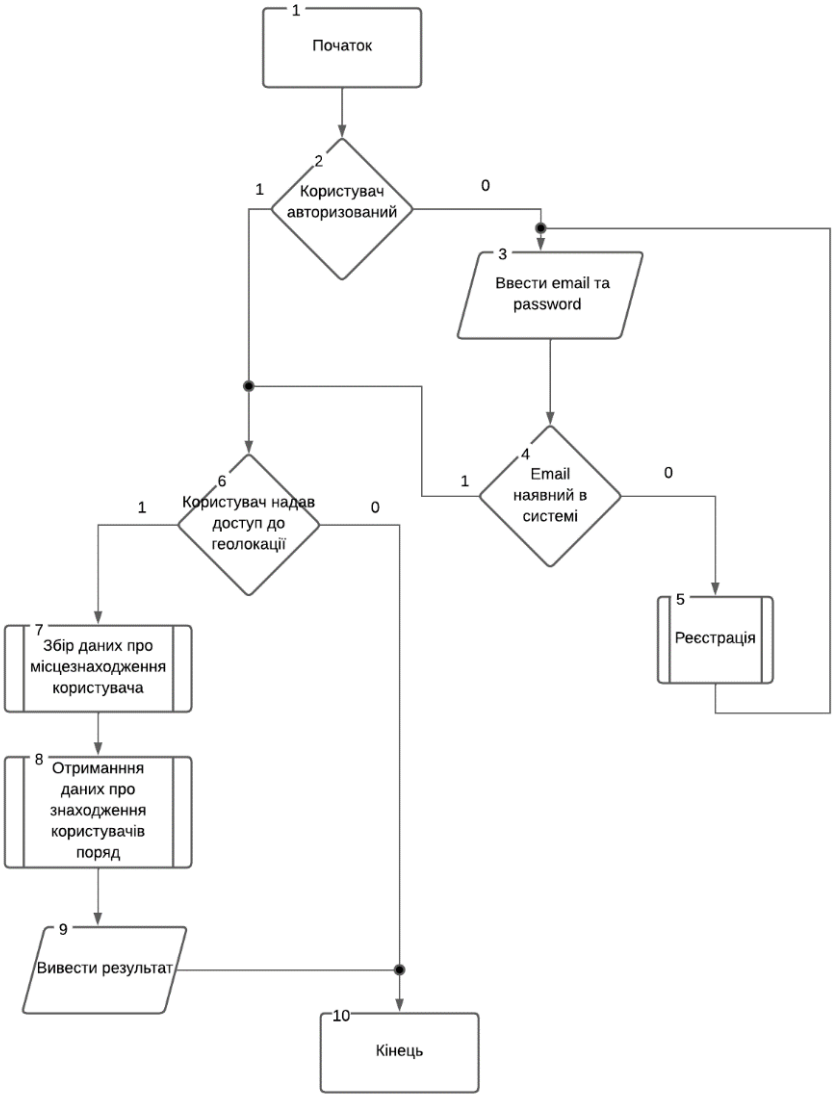

Рисунок 2.3 – Блок-схема алгоритму роботи системи відстеження місцезнаходження

Такий алгоритм забезпечує зручний та ефективний спосіб використання системи для користувачів, а також забезпечує захист їх приватності та безпеки даних.

Розроблений алгоритм надав можливість підвищити гнучкість її інтеграції, швидкість роботи та користувацький досвід.

2.5 Розробка діаграми сутностей бази даних

Бази даних грають є важливою частиною веб-розробки, де вони використовуються для зберігання та організації даних, які використовуються в веб-застосунках. Є два види баз даних: реляційні бази даних – це найпоширеніший тип баз даних у веб-розробці, де дані організовані у вигляді таблиць, що мають відносини між собою та NoSQL бази даних – у деяких випадках, коли передбачається робота з великим обсягом даних або потребою у високій швидкодії, використання NoSQL баз даних може бути вигідним.

Для розробки програмного забезпечення для соціальної мережі з функцією відображення геолокації користувачів було обрано реляційну базу даних. Також було розроблено схему, яка демонструє основні сутності, їх поля та зв’язки між ними. Рисунок діаграми сутностей зображено на рисунку 2.4.

Було розроблені такі сутності:

1. Користувач – користувач має такі поля: ім’я, фотографію профілю, електронну пошту, пароль та роль.
2. Підписка – містить інформацію про те який користувач на кого підписаний.
3. Вподобання допису – містить інформацію про те, який користувач вподобав який допис.
4. Допис – допис містить такі поля: ідентифікатор автора, посилання на зображення, текст, дату публікацію та дату останньої зміни.
5. Вподобання коментаря – містить інформацію про те який користувач вподобав який коментар.

6. Коментар до допису – містить такі поля: ідентифікатор автора, ідентифікатор допису та текст коментаря.

7. Коментар до коментаря – містить такі поля: ідентифікатор автора, ідентифікатор коментаря та текст.

8. Сповіщення – містить такі поля: повідомлення, ідентифікатор отримувача, дата та час.

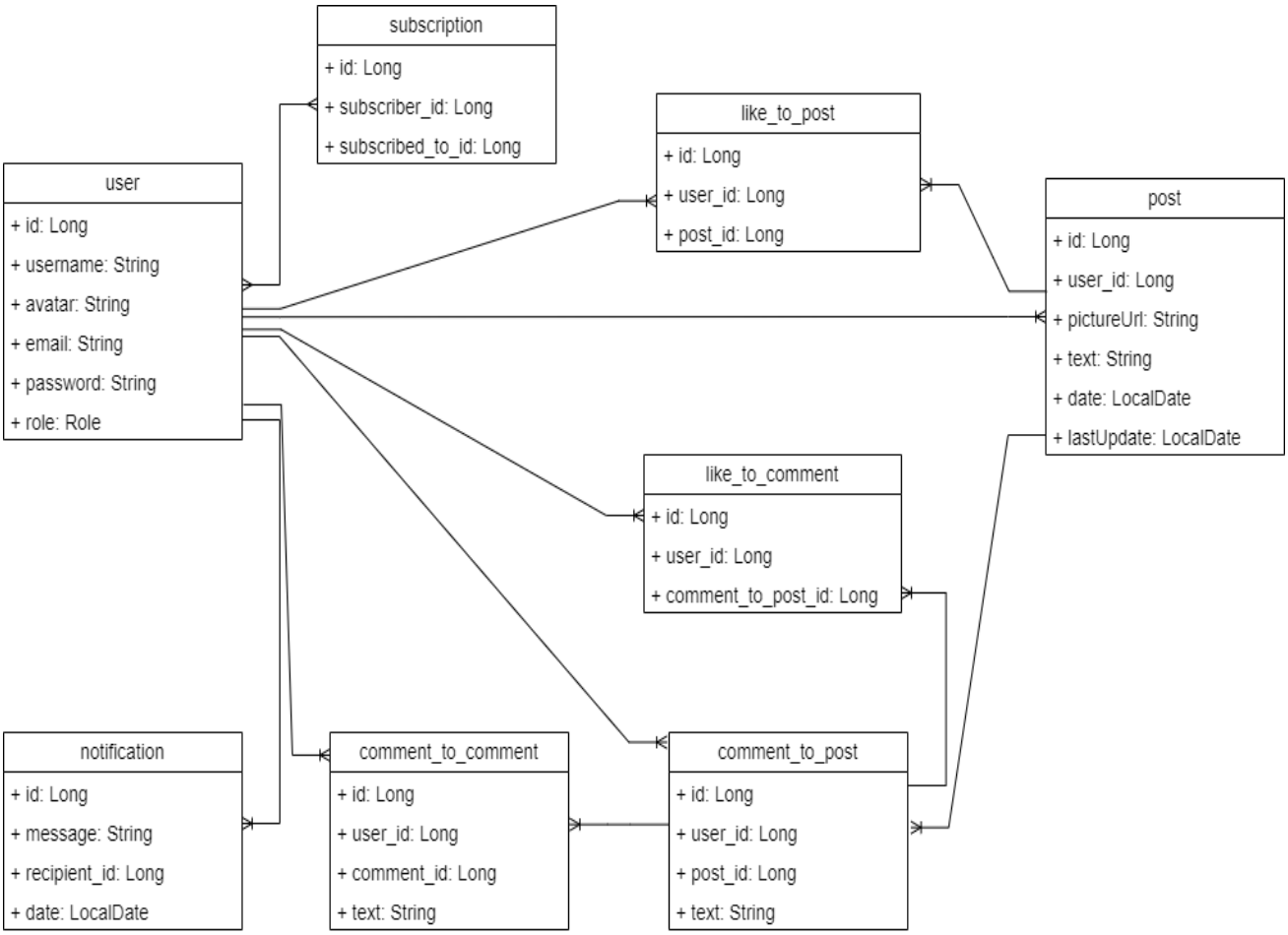

Рисунок 2.4 – Діаграма сутностей

Розроблена діаграма сутностей, в якій детально описані сутності та зв'язки між ними, є важливим етапом у процесі створення програмного забезпечення. Вона значно полегшує розробку бази даних, оскільки надає структурований підхід до моделювання даних. Ретельно розроблена діаграма сутностей є невід'ємною частиною процесу розробки бази даних для застосунку, забезпечуючи структурований, логічний та ефективний підхід до управління даними.

2.6 Розробка дизайну інтерфейсу програмного застосунку

Інтерфейс користувача – це зовнішній вигляд та відчуття програмного забезпечення. Це те, за допомогою чого користувачі контролюють та взаємодіють зі своїми пристроями. Він відіграє ключову роль у створенні продукту. Якісний інтерфейс користувача має бути зручним у використанні, забезпечуючи плавну, зручну та приємну взаємодію. Тому розробляючи інтерфейс для застосунку було вирішено зробити його максимально простим та зручним для користувача.

Основними кольорами застосунку є синій та білий, тому що вони є нейтральними та добре контрастують один з одним.

Після запуску застосунку користувача зустрічає екран завантаження з логоитипом програми рисунок (рис. 2.5).

Рисунок 2.5 – Екран завантаження

Оскільки екран реєстрації є ключовою частиною будь-якого застосунку, тому що він дозволяє користувачам створювати облікові записи і отримувати доступ до функціональності системи, у проєкті було спроектовано екран реєстрації (рис. 2.6). Для забезпечення коректної роботи застосунку та захисту персональних даних користувачів, екран реєстрації має включати деякі

обмеження та перевірки, які забезпечують введення правильних та достовірних даних.

Ось декілька обмежень та перевірок, які можуть бути введені на екрані реєстрації:

– перевірка правильності формату електронної пошти: Система повинна перевіряти, чи введена користувачем електронна пошта має правильний формат;
– перевірка унікальності електронної пошти: Система повинна перевірити, чи введена користувачем електронна пошта є унікальною в системі, щоб уникнути створення дублікатів облікових записів;
– перевірка наявності обов'язкових полів: Система повинна перевіряти, чи всі обов'язкові поля були заповнені користувачем перед тим, як надіслати дані для реєстрації.

Ці обмеження та перевірки допомагають забезпечити безпеку та коректну роботу застосунку, запобігаючи введенню некоректних або недійсних даних, а також уникнути створення неправильних або дубльованих облікових записів.

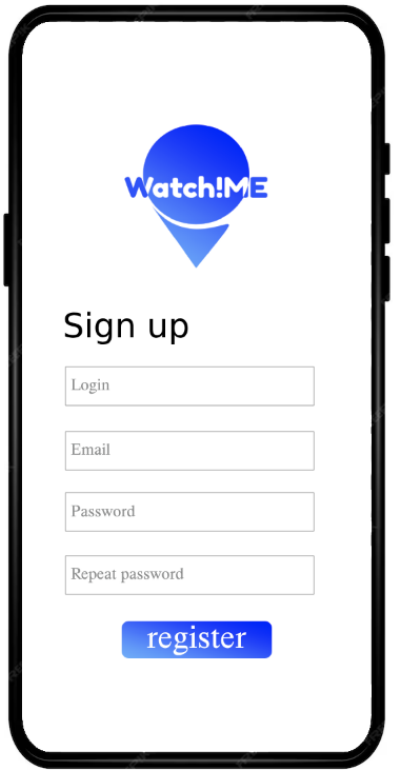

Рисунок 2.6 – Екран реєстрації

Після реєстрації користувача потрібно його авторизувати в системі, для цього було спроектовано екран автентифікації (рис. 2.7), де користувач повинен вказати електронну пошту яка закріплена за його акаунтом, та пароль.

Рисунок 2.7 – Екран автентифікації

Після автентифікації користувача на екрані з мапою він може переглядати місцезнаходження інших користувачів, які знаходяться поблизу. З моменту імплементації системи з соціальною мережею, мапу можна зробити інтерактивною, що відкриває багато нових можливостей для взаємодії користувачів з застосунком.

Після інтеграції системи з соціальною мережею є можливість додати активності до інтерактивної мапи, наприклад:

- показ профілю користувача: Користувач може обрати на мапі конкретного користувача та переглянути його профіль, який містить інформацію про нього, його фотографії, тощо;
- обмін повідомленнями: Користувачі можуть надсилати повідомлення один одному прямо з мапи, що дозволить зручно спілкуватися, коли вони знаходяться поблизу;
- функція позначок: Користувач може позначити певні місця на мапі та ділитися ними з іншими користувачами, що сприяє плануванню зустрічей або подорожей.

Для забезпечення приватності користувачів, система може обмежити доступ до їх місцезнаходження лише для тих, хто надає взаємний доступ. Це важливий аспект, який дозволяє користувачам контролювати, хто має доступ до їхньої особистої інформації про місцезнаходження.

Реалізація цього обмеження може бути здійснена різними способами. Наприклад, система може відображати на мапі лише тих користувачів, які є взаємними підписниками в соціальній мережі або які надали спеціальний доступ до свого місцезнаходження. Це означає, що користувачі мають контроль над тим, кому їхня інформація про місцезнаходження доступна, та можуть вибирати, з ким ділитися цією інформацією.

Такий підхід сприяє збереженню приватності користувачів і відповідає сучасним стандартам захисту особистих даних. Він дозволяє створювати довіру до системи серед користувачів та забезпечує їм контроль над тим, яка інформація про них доступна для інших користувачів.

2.7 Висновки

У другому розділі було проведено аналіз вхідних та вихідних даних системи, щоб краще зрозуміти її функціональність та потреби користувачів. Цей аналіз допоміг визначити, які дані будуть введені в систему, як вони будуть оброблятися та які результати будуть виведені. Також було розглянуто різні сценарії використання, щоб зрозуміти, як система повинна реагувати на різні

ситуації. Також було розроблено модель роботи системи за допомогою UML діаграми. Ця діаграма візуалізує структуру та взаємодії між різними компонентами системи. Вона допомагає програмістам та інженерам краще розуміти архітектуру програмного продукту та його функціональність. Було розроблено метод формування стрічки рекомендацій користувача за допомогою штучного інтелекту. Також було розроблено алгоритм відслідковування геолокації користувачів. Крім того було розроблено діаграму сутностей. Також було розроблено дизайн інтерфейсу застосунку.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

При розробці програмного забезпечення для соціальної мережі з функцією відображення місцезнаходження людей поруч з користувачем, важливо добре обдумати вибір технологій та мови програмування, які будуть використовуватися.

Зважаючи на характеристики соціальної мережі з функцією відображення місцезнаходження людей поруч з користувачем, було проведено порівняльний аналіз мов програмування.

Результати порівняння наведено в таблиці 3.1.

Таблиця 3.1 – Порівняння мов програмування

Характеристика	Java	Kotlin	Python
Висока продуктивність	+	-	-
Підтримка Android	+	+	+
Підтримка багатопотоковості	+	+	-
Крос-платформеність	+	+	+
Масштабованість	+	+	-
Велика спільнота	+	+	+
Швидкодія	+	+	+
Загальна оцінка	7	6	4

За результатами наведеними у таблиці зроблено висновок, що мова програмування Java найкраще підходить для розв’язання поставлених задач при розробці програмного забезпечення для соціальної мережі з функцією відображення місцезнаходження людей поруч з користувачем.

Для розробки програмного продукту були обрані мова програмування Java, Spring Framework бібліотека Retrofit, та хмарний сервіс реляційної бази даних Amazon RDS.

Мова програмування Java є однією з найбільш популярних та широко використовуваних мов для розробки програмних засобів, особливо в сфері веб-розробки та мобільних застосунків. Її вибір для розробки програмних засобів для відображення місцезнаходження людей поруч з користувачем у соціальній мережі має кілька обґрунтованих причин.

Java є мовою з високим рівнем переносимості, що означає, що програмні засоби, розроблені на Java, можуть працювати на різних платформах без значних змін. Це дозволяє забезпечити широкий охоплення аудиторії користувачів соціальної мережі незалежно від їх пристрою чи операційної системи. Також Java має велику кількість бібліотек та фреймворків, які сприяють швидкому розвитку програм. Це дає можливість зручно використовувати різноманітні інструменти для реалізації складної функціональності, такої як відображення місцезнаходження користувачів у соціальній мережі.

Spring Framework відомий своєю гнучкістю та широким набором інструментів, що дозволяє легко створювати високоефективні та масштабовані веб-застосунки. Це дає можливість розробникам швидко реалізувати функціональність соціальної мережі та додавати нові функції без значних зусиль [13].

Бібліотека Retrofit обрана з метою забезпечення простоти, ефективності та надійності у взаємодії з сервером соціальної мережі. Вона надає зручний та зрозумілий спосіб виконання HTTP-запитів та обробки відповідей в форматах JSON або XML.

Завдяки Retrofit можна ефективно взаємодіяти з сервером мережі та обмінюватися даними з іншими користувачами. Це дозволяє спростити процес розробки, оскільки вона забезпечує широкі можливості для роботи з HTTP-запитами та відповідями.

Однією з основних переваг Retrofit є його зрозумілий та інтуїтивно

зрозумілий інтерфейс. Використання анотацій дозволяє швидко визначити параметри запиту, що робить процес розробки більш простим та зрозумілим.

Для розробки графічного інтерфейсу було обрано інструмент Layout Editor в Android Studio. Цей інструмент надає потрібний функціонал для створення і налаштування різноманітних елементів інтерфейсу користувача, включаючи компоненти для відображення місцезнаходження користувачів.

За допомогою Layout Editor можна легко розміщувати елементи інтерфейсу на екрані, налаштовувати їх вигляд та поведінку, щоб створити зручний та привабливий інтерфейс для користувачів. Це дозволяє забезпечити зручність у використанні застосунку і підвищити його привабливість для користувачів [14].

3.2 Вибір системи управління базами даних

Такий підхід дозволяє покращити якість взаємодії з сервером соціальної мережі, зменшити кількість помилок та підвищити ефективність роботи програми в цілому.Amazon Relational Database Service (Amazon RDS) – це веб-сервіс, який полегшує налаштування, експлуатацію та масштабування реляційної бази даних в хмарному середовищі AWS [15].

Amazon RDS – це потужний сервіс хмарної бази даних, який надає готові до використання реляційні бази даних, такі як MySQL, PostgreSQL та Oracle. Це дозволяє розробникам швидко налаштувати та масштабувати свої бази даних без необхідності турбуватися про апаратне забезпечення та програмне забезпечення. Однією з ключових переваг Amazon RDS є висока доступність та надійність. Система автоматично створює резервні копії даних, автоматично відновлює базу даних в разі виникнення неполадок та забезпечує можливість реплікації даних для забезпечення високої доступності та швидкодії. Це забезпечує неперервну роботу системи та захищає важливі дані від втрати.

Не менш важливою перевагою є зменшення витрат на обладнання та підтримку інфраструктури за рахунок використання хмарного сервісу. Це робить Amazon RDS привабливим вибором для стартапів та малих компаній, дозволяючи їм зосередитися на розвитку своїх продуктів, а не на управлінні інфраструктурою.

Загалом, використання Amazon RDS забезпечує безпеку та надійність зберігання важливої інформації про місцезнаходження.

PostgreSQL – це потужна реляційна база даних, яка є вільним та відкритим програмним забезпеченням. PostgreSQL – це надійний та гнучкий інструмент для зберігання та обробки даних.

Однією з основних переваг PostgreSQL є його багатофункціональність, спрямована на допомогу розробникам у будівництві застосунків та адміністраторам у забезпеченні цілісності даних та створенні надійних середовищ. Завдяки широкому набору функцій, PostgreSQL може задовольнити потреби різних проєктів, незалежно від їх розміру чи складності.

Також важливою перевагою є висока розширюваність PostgreSQL. Система підтримує розширення, які дозволяють розробникам розширювати функціональність бази даних за потреби проєкту, забезпечуючи високий рівень гнучкості та адаптивності.

Враховуючи всі ці переваги, PostgreSQL стає привабливим вибором для розробників, які шукають надійний та потужний інструмент для своїх проєктів. Його вільна та відкрита природа також сприяє зниженню витрат на ліцензії та забезпечує широку спільноту, яка може надати підтримку та поради. [16].

Отже, використання мови програмування Java, бібліотеки Retrofit та Amazon RDS є обґрунтованим вибором і надає всі необхідні інструменти для розробки програмних засобів для відображення місцезнаходження людей поруч з користувачем у соціальній мережі.

3.3 Розробка модулю для отримання інформації про геолокацію користувача через GPS

Головною функціональністю програмного продукту є отримання інформації про місцезнаходження користувача за допомогою GPS. Для досягнення цієї мети необхідно спочатку ініціалізувати «слухача» для відстеження змін у геолокації.

Це дозволить програмі автоматично отримувати оновлені координати користувача, коли він переміщується. Після ініціалізації «слухача» програмний

продукт буде готовий отримувати оновлені координати користувача та використовувати їх для подальших функцій, таких як відображення на мапі чи збереження у базі даних.

Цей процес забезпечує точність та надійність отримання інформації про місцезнаходження користувача, що є ключовим аспектом роботи програмного продукту.

Програмну реалізацію інтерфейсу «слухача» зображено на рисунку 3.1.

```
2 usages
public class MyLocationListener implements LocationListener {

    2 usages
    private LocalListenerInterface localListenerInterface;

    1 usage
    public void setLocalListenerInterface(LocalListenerInterface localListenerInterface) {
        this.localListenerInterface = localListenerInterface;
    }

    6 usages
    @Override
    public void onLocationChanged(@NonNull Location location) {
        localListenerInterface.onLocationChanged(location);
    }
```

Рисунок 3.1 – «Слухач» для відслідковування зміни в геолокації користувача

Реалізація екземпляра «слухача» для відстеження геолокації користувача шляхом імплементації вже існуючого інтерфейсу та перевизначення його методів у класі, що створюється, є досить поширеним підходом у програмуванні. Цей підхід дозволяє використовувати готові інтерфейси або абстрактні класи та адаптувати їх під конкретні потреби застосунку.

Інтерфейс може визначати методи для відстеження геолокації, наприклад, методи для отримання координат, визначення розташування на карті тощо. Перевизначення цих методів у класі дозволить забезпечити власну реалізацію функціональності відстеження геолокації, яка відповідає конкретним потребам застосунку.

Цей підхід дозволяє розділити логіку відстеження геолокації на окремий компонент, що полегшує його використання та підтримку у системі. Крім того,

він сприяє збереженню принципів об'єктно-орієнтованого програмування, а саме поліморфізм – об’єкти різних класів можуть мати однакові методи, але реалізовувати їх різними способами. Програма працює з усіма об’єктами, використовуючи загальний інтерфейс, що робить код більш гнучким і універсальним.

Після отримання інформації про місцезнаходження, необхідно її зашифрувати, за допомогою вже реалізованих інструментів в мові Java, що допоможе забезпечити конфіденційність та безпеку даних. Шифрування інформації про місцезнаходження є важливим кроком для захисту приватності користувачів та запобігання несанкціонованому доступу до їх особистої інформації. Під час шифрування дані перетворюються в криптографічно безпечний формат, який неможливо прочитати без відповідного ключа.

Це дозволяє зберігати інформацію про місцезнаходження в зашифрованому вигляді, що робить її недоступною для будь-яких несанкціонованих осіб чи програм.

Застосування сучасних методів шифрування, таких як AES (Advanced Encryption Standard), дозволяє надійно захистити дані. Наприклад, можна використовувати бібліотеки Java, такі як javax.crypto або java.security.

Програмну реалізацію процесу шифрування даних про місцезнаходження користувача зображено на рисунку 3.2.

```
no usages
public List<byte[]> encryptData(String lon, String lat){
    List<byte[]> encrypted = new ArrayList<>();
    try {
        Cipher cipher = Cipher.getInstance( transformation: "AES/CTR/NoPadding");
        SecretKey secretKey = new SecretKeySpec(secretString.getBytes(StandardCharsets.UTF_8),
                algorithm: "AES");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        encrypted.add(cipher.doFinal(lon.getBytes()));
        encrypted.add(cipher.doFinal(lat.getBytes()));
    } catch (InvalidKeyException | NoSuchPaddingException | NoSuchAlgorithmException |
             IllegalBlockSizeException | BadPaddingException e) {
        throw new RuntimeException(e);
    }

    return encrypted;
}
```

Рисунок 3.2 – Програмний код шифрування інформації про місцезнаходження

Алгоритм шифрування інформації про місцезнаходження реалізований за допомогою бібліотеки Cipher, яка надає різні алгоритми шифрування. Першим кроком є створення секретного ключа. Для обраного алгоритму шифрування AES ключ може мати різну довжину, наприклад, 128, 192 або 256 біт. Програмний продукт, що розробляється використовує алгоритм AES з довжиною секретного ключа 256 біт [17]. Потім ініціалізується об’єкт Cipher, який буде відповідальний за шифрування інформації. Після ініціалізації об'єкту Cipher можна використовувати методи класу для шифрування.

Цей підхід забезпечує високий рівень безпеки, оскільки алгоритм AES є одним з найбільш надійних і широко використовуваних алгоритмів шифрування.

Після шифрування отримані дані потрібно передати в сервіс (рис. 3.3), який відповідає за збереження та обробку місцезнаходження користувачів. Цей процес відбувається через відправку запиту на сервіс за допомогою Retrofit. Даний сервіс запущений на віддаленому сервері та відповідає за зберігання та оновлення даних, а також за забезпечення доступу до них відповідно до правил безпеки та приватності.

```
1 usage
public void sendLocationInfo(Double latitude, Double longitude){
    LocationDto locationDto = new LocationDto();
    locationDto.setLatitude(latitude);
    locationDto.setLongitude(longitude);

    Call<Void> call = locationServiceApi.postLocation(locationDto);

    try {
        call.execute();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

Рисунок 3.3 – Передача повідомлення на сервіс для роботи з місцезнаходженням

Використання бібліотеки Retrofit для передачі повідомлень між сервісами - це ефективний та зручний спосіб реалізації мережевого взаємодії у програмному

проєкті. Для того, щоб сервіс, який запущений на віддаленому сервері, отримував повідомлення, необхідно ініціалізувати об'єкт класу Retrofit та вказати базове посилання на сервер.

Важливою умовою для коректної роботи з'єднання є безпечність передачі даних. Для цього використовується протокол шифрування з'єднання Transport Layer Security (TLS), раніше відомий як Secure Sockets Layer (SSL). Цей протокол забезпечує захищеність передачі інформації шляхом шифрування даних між клієнтом та сервером.

Для реєстрації протоколу TLS можна використати сервіси, такі як CloudFlare, які надають можливість придбання доменів та реєстрації протоколів шифрування комунікацій. Крім того, CloudFlare надає корисну статистику про відвідування домену користувачами (рис. 3.4).

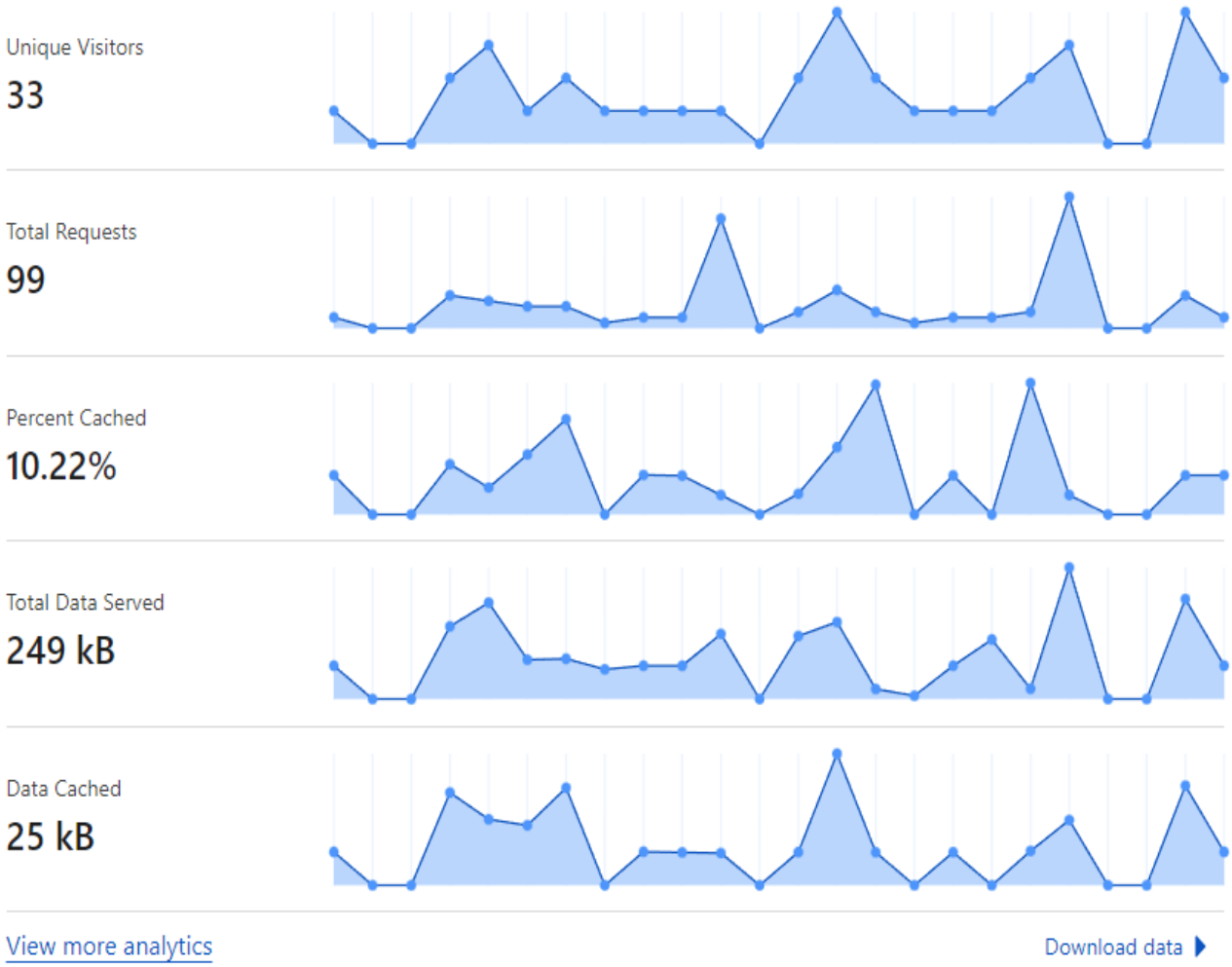

Рисунок 3.4 – Аналітика відвідин домену

Отже, інтеграція Retrofit з використанням протоколу TLS та реєстрація протоколу шифрування через сервіси, такі як CloudFlare, дозволять забезпечити безпечну та надійну передачу повідомлень між сервісами.

Для запуску модулю на віддаленому сервері було обрано сервіс Amazon EC2. Amazon EC2 надає широкий вибір обчислювальних можливостей з більш ніж 750 типами інстансів та вибором найновіших процесорів, сховищ, мереж, операційних систем та моделей придбання, щоб найкращим чином задовольнити потреби [18].

Цей підхід дозволяє забезпечити надійну та безпечну збереження та обробку даних про місцезнаходження користувачів, а також забезпечує можливість використання сучасних обчислювальних можливостей для оптимальної роботи програмного продукту.

Після успішного збереження даних у хмарному середовищі, система може відобразити користувачеві інформацію про місцезнаходження інших користувачів (рис. 3.5) та відобразити ці дані на мапі.

```
1 usage
@Override
public List<LocationDto> getLocations() {
    Call<List<LocationDto>> call = locationServiceApi.getLocations();
    Response<List<LocationDto>> response;
    try {
        response = call.execute();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
    return response.body();
}
```

Рисунок 3.5 – Отримання даних про місцезнаходження з сервісу

Після того, як всі перераховані вище кроки успішно виконають своє завдання, фінальним кроком буде розшифровка даних про місцезнаходження інших користувачів (рис. 3.6) та відображення цих даних на мапі.

```
no usages
public List<String> dencryptData(byte[] lon, byte[] lat){
    List<String> decrypted = new ArrayList<>();
    try {
        Cipher cipher = Cipher.getInstance( transformation: "AES/CTR/NoPadding");
        SecretKey secretKey = new SecretKeySpec(secretString.getBytes(StandardCharsets.UTF_8)
                , algorithm: "AES");
        cipher.init(Cipher.DECRYPT_MODE, secretKey);

        String lonString = new String(cipher.doFinal(lon), StandardCharsets.UTF_8);
        String latString = new String(cipher.doFinal(lat), StandardCharsets.UTF_8);

        decrypted.add(lonString);
        decrypted.add(latString);

    } catch (InvalidKeyException | NoSuchPaddingException | NoSuchAlgorithmException |
             IllegalBlockSizeException | BadPaddingException e) {
        throw new RuntimeException(e);
    }

    return decrypted;
}
```

Рисунок 3.6 – Дешифрування даних про місцезнаходження користувачів

Після отримання зашифрованих даних про місцезнаходження інших користувачів система використовує відповідний ключ для розшифрування цих даних. Цей процес перетворює зашифровану інформацію в зрозумілу форму, яку можна використовувати для подальшого аналізу та відображення.

Розшифровані дані дозволяють користувачеві бачити місцезнаходження інших користувачів та взаємодіяти з ними згідно з функціональними можливостями програмного продукту. Наприклад, користувач може бачити місце розташування своїх друзів на карті, обмінюватися повідомленнями або навіть планувати зустрічі на основі їхнього місцезнаходження.

Цей процес дозволяє створити зручне та ефективне середовище для користувачів програмного продукту, сприяючи взаємодії та обміну інформацією. Крім того, за допомогою шифрування та розшифрування забезпечується високий рівень безпеки та конфіденційності передаваних даних, що є важливим аспектом у сфері збереження приватності користувачів.

3.4 Висновки

У розділі було розглянуто вибір технологій для розробки модулів для реалізації в соціальній мережі функції відображення місцезнаходження людей

поруч з користувачем. Дослідивши потреби програмного продукту було зроблено висновки, що найкращим рішенням для розробки є мова програмування Java, використання бібліотеки Retrofit для обміну повідомленнями між сервісами та використання Amazon RDS як хмарного сховища даних, в якості системи управління базами даних було обрано PostgreSQL. Було розроблено модуль для отримання інформації про геолокацію користувачів, для цього було реалізовано функціонал збору даних про геолокацію, шифрування, та дешифрування цих даних.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Модульне тестування

Один із методів тестування програмного забезпечення це модульне тестування. Модульне тестування є одним із найважливіших аспектів забезпечення якості програмного забезпечення. Це метод тестування програмного забезпечення, при якому окремі модулі або компоненти програми перевіряються на коректність функціонування. – це метод тестування програмного забезпечення, який полягає в окремому (ізольованому) тестуванні кожного мінімально можливого для тестування компоненту програми. Модулем називають найменшу частину програми, яка може бути протестованою [19]. В різних мовах програмування модулі можуть бути представлені класами, функціями, процедурами чи методами.

Модульне тестування проводиться за принципом «білого ящика», тобто ґрунтується на знанні внутрішньої структури програми, і часто включає ті або інші методи аналізу покриття коду.

Перевагами модульного тестування є:

- виявлення помилок на ранніх стадіях;
- поліпшення вигляду коду: Написання тестів стимулює розробників до створення модульного, добре структурованого коду;
- полегшення внесення змін: Наявність тестів дозволяє безпечно вносити зміни та проводити рефакторинг коду.

Для створення якісних модульних тестів мають відповідати наступним критеріям:

- швидкі: Розробник не повинен зволікати при запуску програми на будь-якому етапі розробки, навіть якщо програма має більше тисячі тестів, розробник повинен отримати результат за лічені секунди;
- ізольовані: Тести повинні бути повністю незалежні так, що на результат їх виконання не впливали ніякі зовнішні фактори;

– повторювані: Результати виконання тесту не повинні відрізнятись при повторному запуску програми;
– само-затверджувальні: Розробник не повинен вручну перевіряти чи виконались тести;
– ретельні: Тести мають покривати всі ділянки коду модуля.

Фреймворком що використовувався для модульного тестування програмного забезпечення є JUnit. JUnit є одним з найбільш поширених інструментів для модульного тестування в екосистемі Java. Він забезпечує простий і зрозумілий інтерфейс для написання та виконання тестових випадків.

Код модульного тесту функції реєстрації зображено на рисунку 4.1.

```
@Test
void testSuccessfulRegistration() throws Exception {
    when(userRepository.findByEmail(validRequestDto.getEmail())).thenReturn( t: Optional.empty());
    when(userRepository.save(any(User.class))).thenReturn(user);

    File avatarsDir = new File( pathname: "avatars");
    avatarsDir.mkdir();

    doNothing().when(pictureService).setAvatar(any(), eq(user.getId()), eq(avatarsDir.getPath()));
    userService.register(validRequestDto);

    verify(userRepository, times( wantedNumberOfInvocations: 1)).findByEmail(validRequestDto.getEmail());
    verify(userRepository, times( wantedNumberOfInvocations: 1)).save(any(User.class));
    verify(pictureService, times( wantedNumberOfInvocations: 1)).
            setAvatar(any(), eq(user.getId()), eq(avatarsDir.getPath()));
    assertEquals(avatarsDir.getPath(), user.getAvatar());
}
```

Рисунок 4.1 – Модульне тестування реєстрації

Метод перевіряє успішну реєстрацію користувача в системі. Тест охоплює всі основні кроки, які повинні бути виконані під час реєстрації, і забезпечує правильність роботи методу реєстрації за умови, що всі введені дані є коректними.

Опис кроків тесту:

– мокування методів репозиторію та сервісів: Метод userRepository.save налаштовується на повернення об'єкта user, який представляє збереженого користувача, метод pictureService.setAvatar

налаштовується на виконання без дій, оскільки він просто встановлює аватар для користувача;

– виконання тестової дії: Викликається метод userService.register з в який передаються правильні дані користувача;
– перевірка очікуваних результатів: Перевірка викликів методів репозиторію, перевірка викликів методів сервісу, перевірка встановлення аватара.

Цей тест гарантує, що всі необхідні дії виконуються правильно під час успішної реєстрації користувача, і всі необхідні залежності працюють як очікується.

Код модульного тесту завантаження зображень зображено на рисунку 4.2.

```
@Test
void testSuccessfulImageUpload() {
    when(userRepository.findById(userId)).thenReturn(Optional.of(user));
    when(pictureRepository.save(any(Picture.class))).thenReturn(new Picture());

    File postDir = new File( pathname: "FOLDER_PATH/" + userId + "/posts/" + postId);
    if (!postDir.exists()) {
        postDir.mkdirs();
    }

    String result = imageUploadService.uploadImage(files, userId, postId);

    verify(userRepository, times( wantedNumberOfInvocations: 1)).findById(userId);
    verify(pictureRepository, times( wantedNumberOfInvocations: 2)).save(any(Picture.class));
    assertTrue(result.contains("FOLDER_PATH/" + userId + "/posts/" + postId));
    assertTrue(postDir.exists());

    for (File file : postDir.listFiles()) {
        file.delete();
    }
    postDir.delete();
}
```

Рисунок 4.2 – Модульне тестування завантаження зображень

Пояснення коду:

– налаштування моків: використання when() і thenReturn() для налаштування поведінки методів репозиторію, використання doNothing() та doThrow() для налаштування поведінки методів, які можуть викликати виключення.;

– виконання методу: викликається метод uploadImage() з відповідними параметрами;
– перевірка результатів: перевірка викликів методів з відповідними параметрами, перевірка коректності повернутого значення та викликів методів збереження зображень.

Цей тест гарантує, що всі необхідні дії виконуються правильно під час завантаження зображень, і всі необхідні залежності працюють як очікується.

Було проведено ретельне тестування системи за допомогою модульних тестів, результат перевірки зображено на рисунку 4.3.

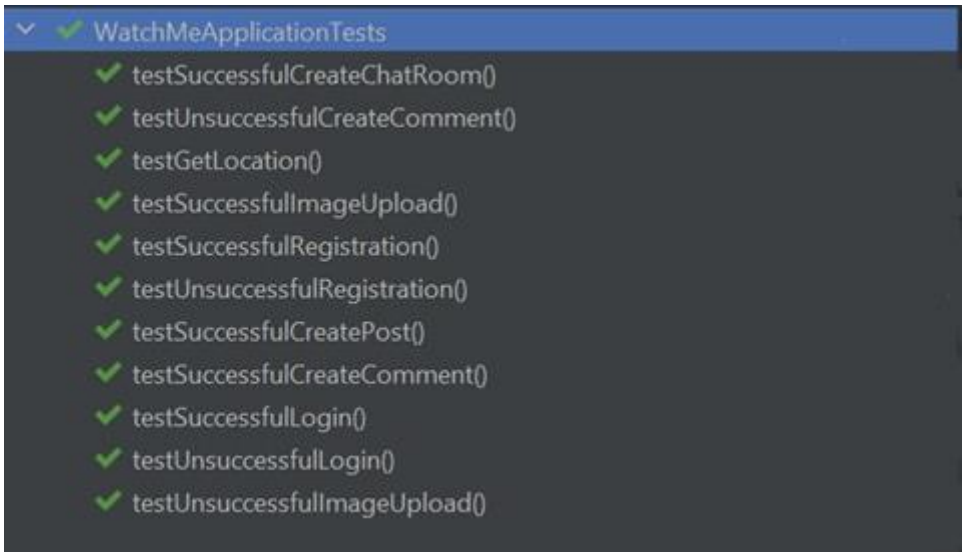

Рисунок 4.3 – Результат тестування системи

Процес модульного тестування програмного забезпечення для соціальної мережі з функцією відображення геолокації користувачів продемонстрував свою ефективність у виявленні та виправленні помилок на ранніх стадіях розробки. Використання автоматизованих тестових випадків забезпечило стабільність та надійність програми, що є критично важливим для її подальшої експлуатації. Модульне тестування дозволило не тільки підвищити якість програмного забезпечення, але й покращити процес розробки в цілому, сприяючи створенню більш структурованого та підтримуваного коду.

4.2 Ручне тестування системи

Для забезпечення надійності та правильної роботи програмних модулів, необхідно виконати ретельне ручне тестування. Цей процес включає в себе серію тестів, які виконують тестувальники, щоб переконатися, що програмне забезпечення працює належним чином.

Ручне тестування мобільних застосунків є невід'ємною частиною цього процесу. Під час ручного тестування тестувальники взаємодіють з застосунком так, як це роблять звичайні користувачі, з метою перевірки функціональності та продуктивності застосунку.

Ручне тестування передбачає активне використання людських здібностей та досвіду для пошуку потенційних проблем та помилок у роботі застосунку. Цей процес включає відтворення реальних сценаріїв використання, перевірку коректності відображення місцезнаходження користувачів поруч, а також оцінку загальної продуктивності та швидкодії програмного забезпечення.

Під час ручного тестування не використовуються жодні автоматизовані скрипти чи інструменти, оскільки саме людський фактор дозволяє виявити різноманітні проблеми та недоліки, які можуть залишитися непоміченими в автоматизованих тестах. Такий підхід дозволяє забезпечити більш високу якість програмного забезпечення та підвищити задоволення користувачів від його використання [20].

Застосування людського фактору в ручному тестуванні дозволяє тестувальникам краще зрозуміти потреби та очікування користувачів, що сприяє створенню більш коректного та відповідного їхнім потребам програмного забезпечення. Такий підхід також дозволяє швидше виявляти та виправляти помилки, що робить процес розробки та випуску продукту більш ефективним і ефективним.

В ході цього процесу були перевірені основні аспекти, включаючи:

Перевірка на можливість зареєструватись не заповнивши всі поля. У разі спроби зареєструватися без введення електронної пошти, ім’я користувача або

паролю має з’явитись повідомлення про те, що треба заповнити всі поля, результат тестування зображено на рисунку 4.4.

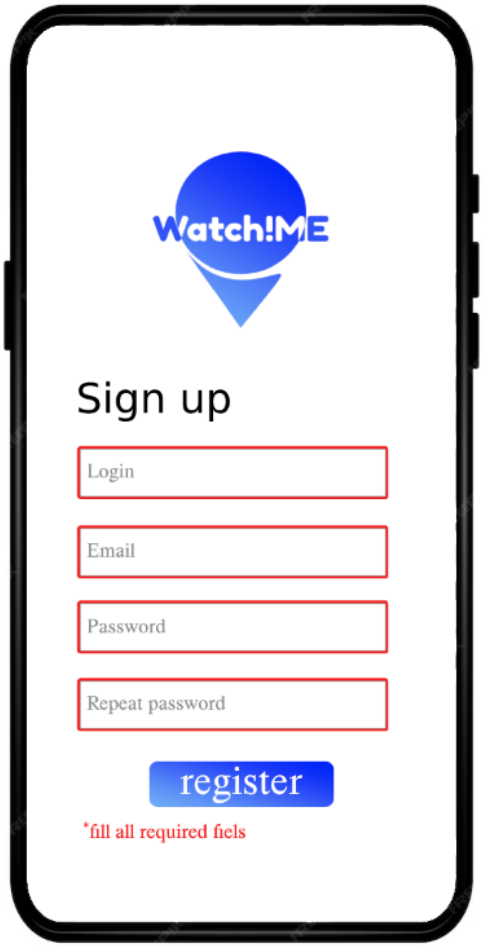

Рисунок 4.4 – Результат тестування можливості зареєструватись не заповнивши всі поля

Після того, як ми переконались, що система не дає змогу зареєструватись користувачу не заповнивши всі поля, протестуємо те як реагує програма на некоректні далі при спробі авторизуватись. Результат тестування зображено на рисунку 4.5.

Після вивчення та перевірки реакції системи на некоректно введені дані під час процесу реєстрації та автентифікації, наступним кроком є перевірка правильності реєстрації користувача за умови введення всіх необхідних даних. Успішна реєстрація повинна мати наслідком автоматичне перенаправлення

користувача на екран автентифікації, що свідчить про успішне створення облікового запису та готовність користувача до входу в систему з використанням введених облікових даних.

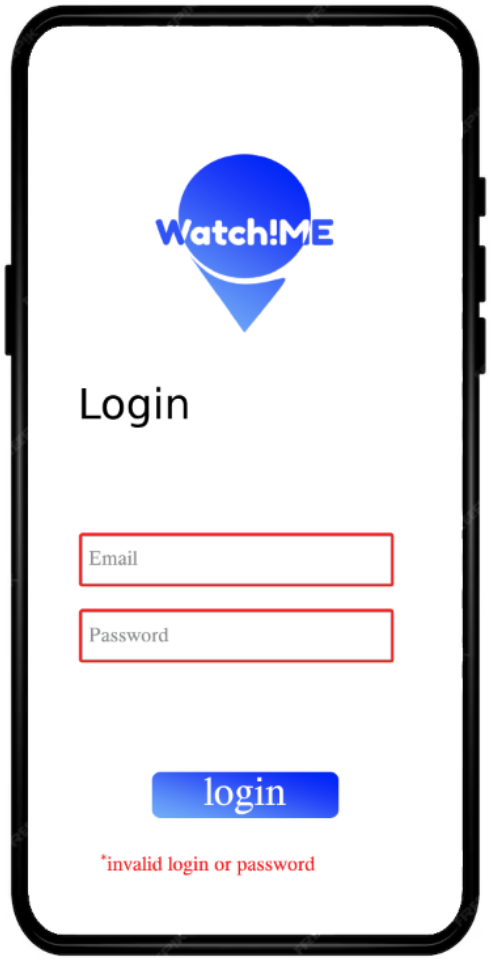

Рисунок 4.5 – Результат тестування некоректно введених даних при автентифікації

Цей етап перевірки є важливим для впевненості в тому, що система працює правильно та здатна обробляти коректно введені дані користувачів. Після успішної реєстрації користувачеві повинно бути забезпечено зручне та безпечне входження до системи, що є важливим аспектом забезпечення його позитивного досвіду використання сервісу.

Таким чином, після введення користувачем всіх необхідних даних та їх успішного підтвердження системою, автоматичне перенаправлення на екран

автентифікації буде важливим підтвердженням успішної реєстрації та готовності користувача до подальшого використання системи., результат тестування зображено на рисунку 4.6.

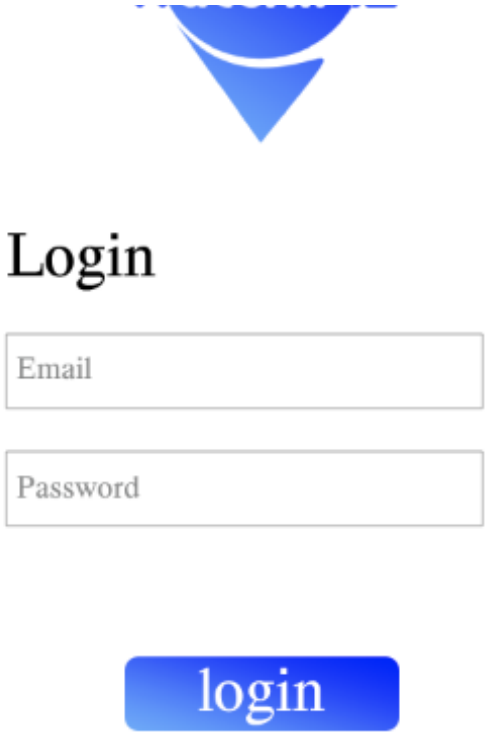

Рисунок 4.6 – Результат тестування реєстрації

Після успішної реєстрації та автентифікації користувач потряпляє на екран з мапою на якій будуть відображатись місцезнаходження людей поруч. Вигляд екрану зображено на рисунку 4.7.

Після тестування основних функцій, пов'язаних з реєстрацією та авторизацією, наступним кроком є перевірка взаємодії системи з віддаленим сервером. Це включає в себе перевірку того, як система взаємодіє з сервером під час передачі та отримання даних.

Після успішної автентифікації буде перевірено, чи коректно інформація зберігається в базі даних. Це включає в себе перевірку того, чи дані користувачів зберігаються правильно, чи відображається відповідна інформація про

місцезнаходження, а також чи виконуються всі вимоги безпеки даних. Результат тестування зображено на рисунку 4.8.

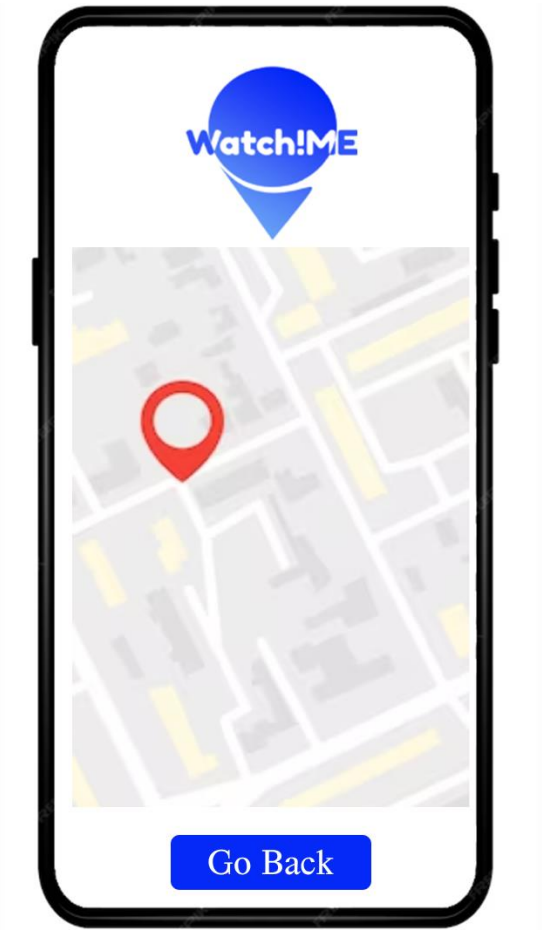

Рисунок 4.7 – Екран з мапою

	id [PK] bigint	user_id bigint	longitude text	latitude text
1	1	10	lPQrtNhJDgKG96nqhy1ezeaZEAz0ksCbH5y2/Q6Poss=	QE4ZEWl4DSc/SkQwq4/oD01dmxUAM8ia550JSlDvj04=
2	2	24	stdvyFUSjuxwo2GZneyXT01dmxUAM8ia550JSlDvj04=	wver2hjX12o/IcTqaCpn901dmxUAM8ia550JSlDvj04=
3	3	32	CPmVpT1Fbmlj0YHSWyk8CnZcrM/DktYF0cF5hx+DsGs=	enhn3w88P6g1kdtFmB4exsMBkj1gKW1XhHRWVwdz...
4	4	50	LnJHv8Myq6YeMwhatiQXQdg87nh6fWfyrPte7ZEPMmU=	ol1PeFGLZW3a+hrb9a3lDyVUy3KH5v89qJyCKZ/WaQY=
5	5	34	rU3bWvKS7Z+MGm7M9iGfXuc3Tkx0e2mMjXZw+DL6g...	jQF2BLYX5rFwvDpI1Dg9A1G6bdkh7J1en6ejKWUMCD...
6	6	40	V3kR4zBYgLrKcXqPJZgK7UioaUVd3WseGW60CMvab2c=	tZU7PuMx8P9Bzi6WENmYkQrQeGkOr84gHbgLcEkyYtI=
7	7	42	lC3bRnIe8X6mKxO7ZvQ9JpQw9nFJdCmWoGtafVzLhUk=	gX9jTtPqRvB3hZnVlMkI4ZiR6KwHfOaYbP5pLcNxZbY=
8	8	62	nW5fGyAmQbVrCsDoHsJkLzPeTzlbVkXxQaShCvEmXg...	zQ1vEoMfUdVxRlNwPyHmKoRtMsIaGcXlZnDbVwQpX...
9	9	17	aD8nWvRmJyUqPzNcFbEkPoVrXsZtGuYhQwlvDlMaKlE=	pZ4lXrAcGvVjRtYnUqEpWsZtCwQsYxBmNjLkMhFySbI=
10	10	67	dF5rNhUwYbXvZmKqJpHsBvQcWsEnRlOpTiAgCkVeFsU=	oR6gTbYnWpVuMsQcEpZwXrCvGtHfYbAjTkMnOlQiRlE=

Рисунок 4.8 – Результат тестування

Після завершення тестування спілкування між сервісами та збереження даних у базі даних отримано позитивний результат. Всі дані були зашифровані перед передачею та успішно збережені у базі даних.

Це свідчить про ефективну роботу механізмів захисту даних, які були розроблені для забезпечення безпеки та конфіденційності інформації. Шифрування даних перед їх передачею гарантує, що навіть у випадку несанкціонованого доступу до мережі, інформація залишається захищеною.

Збереження даних у базі даних без проблем свідчить про правильну інтеграцію сервісів та правильне функціонування бази даних. Це важливий крок у забезпеченні надійності та стійкості програмного забезпечення.

4.3 Розробка інструкції користувача

Інструкція користувача – це невід'ємна частина будь-якого продукту або сервісу, яка надає користувачам докладну інформацію про те, як використовувати його безперешкодно. Інструкція також відома як посібник користувача або керівництво користувача. Такі документи містять детальну інформацію про операції, стандарти та рекомендації, посібники з усунення неполадок, функціональні можливості та багато іншого [21]. Серед ключових аспектів, які передбачає інструкція користувача, є системні вимоги. Системні вимоги визначають технічні параметри, які необхідні для оптимального функціонування певного продукту або сервісу на пристрої користувача. Ці вимоги є ключовим елементом інструкцій, оскільки вони визначають, які конкретні характеристики має мати пристрій або система користувача для забезпечення ефективної роботи продукту або сервісу [22]. Серед цих системних вимог можуть бути такі параметри, як обсяг оперативної пам'яті, тип і версія операційної системи, доступ до Інтернету, обсяг вільного місця на диску, наявність необхідних драйверів або підтримка конкретних протоколів зв'язку. Вказівка правильних системних вимог дозволяє користувачам забезпечити собі належні умови для коректної роботи продукту чи сервісу, уникнути можливих проблем з сумісністю та забезпечити оптимальний досвід використання. Тому включення детальної і точної інформації

щодо системних вимог є необхідним етапом у створенні інструкцій для користувачів, перелік системних вимог наведено в таблицях 4.1 та 4.2

Таблиця 4.1 – Мінімальні системні вимоги

Операційна система	Android 7
Процесор	1 ГГц
Оперативна пам’ять	1 ГБ
Місце на пристрої	1 ГБ

Таблиця 4.2 – Рекомендовані системні вимоги

Операційна система	Android 9
Процесор	2 ГГц
Оперативна пам’ять	2 ГБ
Місце на пристрої	2 ГБ

Після запуску застосунку, першим кроком для користувача є процес автентифікації. Цей етап є необхідним для ідентифікації користувача та надання доступу до особистого облікового запису. Якщо користувач вже має обліковий запис у системі, він повинен ввести свої облікові дані, такі як логін та пароль, для автентифікації. Це дозволяє системі перевірити правильність введених даних і надати доступ до облікового запису користувача.

У випадку, якщо користувач ще не зареєстрований у системі, йому потрібно буде пройти процедуру реєстрації. Під час реєстрації можуть бути запропоновані для введення такі дані, як ім'я, електронна пошта та пароль. Після успішної реєстрації користувач матиме змогу ввести свої облікові дані та авторизуватись у системі, отримавши доступ до особистого облікового запису.

Цей процес дозволяє забезпечити безпеку та конфіденційність інформації користувача та забезпечити доступ лише автентифікованим користувачам до функціоналу застосунку.

Після успішної автентифікації, система може запитати доступ до геолокації користувача (рис. 4.9). Це може бути необхідно для функцій, пов'язаних з відображенням місцезнаходження користувача на карті або для надання послуг, які потребують доступу до геоданих. Зазвичай це робиться для поліпшення користувацького досвіду або для надання специфічних функціональних можливостей, пов'язаних з місцезнаходженням.

Отримання доступу до геолокації може допомогти покращити функціональність застосунку, наприклад, дозволяючи користувачам знаходити інших користувачів у їхній непосередній близькості або надавати інформацію про місцезнаходження для використання у послугах доставки чи навігації. Однак важливо забезпечити, щоб користувачі були повідомлені про цей запит і мали можливість зрозуміти, для яких цілей вони надають доступ до своїх геоданих.

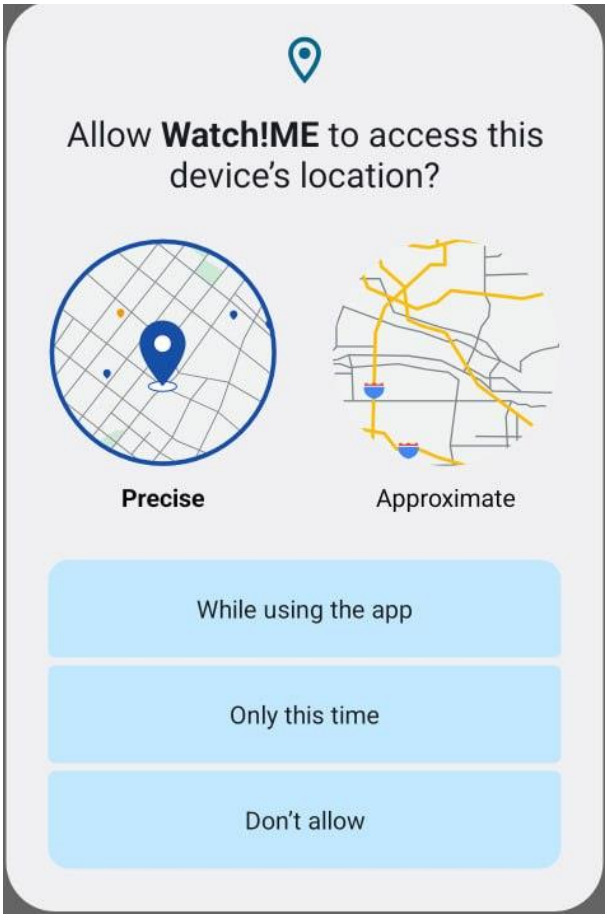

Рисунок 4.9 – Запит на отримання даних про геолокацію

Застосунки, що використовують відстеження місцезнаходження забов’язані запитувати дозвіл на отримання цієї інформації, цей механізм створений для захисту інформації користувача [23]. Отримання цієї інформації необхідно для відображення місцезнаходження користувача та інших людей у навколишній області. Лише після надання доступу до геолокації система може відобразити місцеположення людей поруч з ним на мапі.

4.4 Висновки

У цьому розділі було проведено модульне та ручне тестування системи. Було вручну протестовано основні функції застосунку, такі як: реакція системи на пусті поля при реєстрації; реакція системи на некоректні дані при реєстрації та реакція системи на реєстрацію з коректними даними. Також було проведено модульне тестування системи за допомогою фреймворка Junit. В ході тестування було виявлено, що система працює коректно. Також в цьому розділі були складені мінімальні та рекомендовані системні вимоги для роботи застосунку, та розроблено інструкцію користувача.

ВИСНОВКИ

Під час роботи над бакалаврською дипломною роботою було розроблено програмне забезпечення для соціальної мережі з функцією відображення геолокації користувачів. Метою програмного забезпечення є покращення користувацького досвіду взаємодії з соціальною мережею за рахунок розробки функції відображення геолокації користувачів.

Було розглянуто стан мобільних систем з використанням засобів для відстеження місцезнаходження та проведено порівняльний аналіз аналогів.

Було проаналізовано вхідні дані та розроблено модель системи вперше запропоновано метод формування стрічки рекомендацій користувача інтелекту, що покращить користувацький досвід для користувачів системи, було розроблено алгоритм відслідковування геолокації користувачів, розроблено діаграму сутностей бази даних, також було розроблено дизайн інтерфейсу прогрманого застосунку.

Було проведено порівняльний аналіз різних сов програмування, що часто використовуються для мобільної розробки і обґрунтовано вибір мови програмування Java. Описано розробку модулів: відслідковування зміни в геолокації користувача; шифрування інформації про місцезнаходження; передачі повідомлення на сервіс для роботи з місцезнаходженням; отримання даних про місцезнаходження з сервісу; дешифрування даних про місцезнаходження користувачів.

Було проведено тестування програми за допомогою модульного тестування сегментів системи. Модульне тестування дозволило не тільки підвищити якість програмного забезпечення, але й покращити процес розробки в цілому, сприяючи створенню більш структурованого та підтримуваного коду. Також було проведено ручне тестування системи. Було вручну протестовано основні функції застосунку, такі як: реакція системи на пусті поля при реєстрації; реакція системи на некоректні дані при реєстрації та реакція системи на реєстрацію з коректними даними.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кондратюк О.О., Романюк О.В. Дослідження різних API для відстеження місцезнаходження пристрою : Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів LIII Всеукраїнської науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії 20-22 березня 2024 р. – Вінниця: ВНТУ, 2024. с. 249–250.

2. Кондратюк О.О. Розробка системи для інтеграції в соціальній мережі функціоналу відстеження місцезнаходження/ О.О. Кондратюк, О.В. Романюк // Молодь в науці: дослідження, проблеми, перспективи, Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21667 (дата звернення: 15.03.2024)

3. Location tracking apps [Електронний ресурс] – Режим доступу до ресурсу: https://timesofindia.indiatimes.com/gadgets-news/location-tracking-apps-5-useful-apps-to-track-location-of-your-friends-and-family/articleshow/107257463.cms (дата звернення: 15.03.2024)

4. What is Life360? Popular family tracking app faces safety concerns [Електронний ресурс] – Режим доступу до ресурсу: https://www.standard.co.uk/news/tech/what-life360-tracker-app-price-features-safety-b1117821.html (дата звернення: 15.03.2024)

5. What is Glympse? [Електронний ресурс] – Режим доступу до ресурсу: https://glympse.zendesk.com/hc/en-us/articles/211602703-What-is-Glympse (дата звернення: 15.03.2024)

6. Why do people use Foursquare and Swarm? [Електронний ресурс] – Режим доступу до ресурсу: https://www.quora.com/Why-do-people-use-Foursquare-and-Swarm-What-are-the-major-current-use-cases (дата звернення: 15.03.2024)

7. JavaParser: Visited. Analyse, transform and generate your Java code base: Нью-Йорк, США: Leanpub, 2023. 63 с.

8. Retrofit library in Android [Електронний ресурс] – Режим доступу до ресурсу:https://www.topcoder.com/thrive/articles/retrofit-library-in-android (дата звернення: 17.03.2024)

9. What is GPS? [Електронний ресурс] – Режим доступу до ресурсу: https://www.gps.gov/systems/gps/ (дата звернення: 21.03.2024)

10. Mastering REST APIs: Boosting Your Web Development Journey with Advanced API Techniques: Нью-Йорк, США: Apress, 2024. 509 с.

11. What is UML Diagram? [Електронний ресурс] – Режим доступу до ресурсу: https://miro.com/diagramming/what-is-a-uml-diagram/ (дата звернення: 30.04.2024)

12. Introduction to Algorithms, fourth edition: Массачусетс, США: The MIT Press, 2024, 1312 с.

13. Spring in Action: Нью-Йорк, США: Manning Publications Co, 2022, 520 с.

14. Develop UI with Views [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/studio/write/layout-editor (дата звернення: 03.05.2024)

15. What is Amazon RDS [Електронний ресурс] – Режим доступу до ресурсу: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html (дата звернення: 05.05.2024)

16. Why use PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/about/ (дата звернення: 07.05.2024)

17. AES Encryption [Електронний ресурс] – Режим доступу до ресурсу: https://nordlayer.com/blog/aes-encryption (дата звернення: 10.05.2024)

18. Amazon EC2 [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ec2/ (дата звернення: 12.05.2024)

19. Авраменко А.С., Авраменко В.С. Косенюк Г.В. Тестування програмного забезпечення. Черкаси, 2017. 284 с.

20. A Complete Guide to Manual Mobile Testing [Електронний ресурс] – Режим доступу до ресурсу: https://katalon.com/resources-center/blog/manual-mobile-testing (дата звернення: 13.05.2024)

21. User Manual Guide [Електронний ресурс] – Режим доступу до ресурсу: https://document360.com/blog/creating-a-user-manual/ (дата звернення: 14.05.2024)

22. Requirements calculator [Електронний ресурс] – Режим доступу до ресурсу: https://academy.creatio.com/docs/requirements/calculator (дата звернення: 16.05.2024)

23. Request location permissions [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/develop/sensors-and-location/location/permissions (дата звернення: 18.05.2024)

ДОДАТКИ

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
к.т.н., проф. О. Н. Романюк
" 12 " березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу
«Розробка програмного забезпечення для соціальної мережі з функцією відображення геолокації користувачів»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
к.т.н., доц. каф. ПЗ О.В. Романюк
" 12 " березня 2024 р.

Виконав:
студент гр. 1ПІ-20б О.О. Кондратюк
" 12 " березня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка програмного забезпечення для соціальної мережі з функцією відображення геолокації користувачів».

Галузь застосування – мобільні технології.

2. Підстава для розробки.

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р по ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки.

Метою роботи є покращення користувацького досвіду взаємодії з соціальною мережею за рахунок розробки програмного забезпечення для соціальної мережі з функцією для відображення геолокації людей поруч з користувачем.

Призначення роботи – розробка та реалізація програмного забезпечення для соціальної мережі з функцією відображення геолокації користувачів.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР:

1. JavaParser: Visited. Analyse, transform and generate your Java code base: Нью-Йорк, США: Leanpub, 2023. 63 с.
2. Singh Ajit. Android Application Development. Сіетл, США : Amazon Publishing, 2022. 318 с.
3. Spring in Action: Нью-Йорк, США: Manning Publications Co, 2022, 520 с.

5. Технічні вимоги

Тип програми – мобільний застосунок; модель розробки – ітеративна; засоби організації даних програми – фреймворк Spring та бібліотека Retrofit;

вхідні дані: запит користувача, дані про місцезнаходженн; середовище розробки – Intellij IDEA; мова програмування – Java.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БДР.
2. Технічне завдання.
3. Лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

Ч.ч.	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану мобільних систем з використанням засобів для відстеження геолокації	14.03.24 – 15.03.24
2	Розробка моделі системи	21.03.24 – 31.03.24
3	Розробка методу формування стрічки рекомендацій користувача	31.03.24 – 05.04.24
4	Розробка алгоритму відслідковування геолокації користувачів	05.04.24 – 10.04.24
5	Розробка дизайну інтерфейсу програмного застосунку	15.04.24 – 30.04.24
6	Розробка програмного забезпечення	30.04.24 – 12.05.24
7	Тестування програмного забезпечення	12.05.24 – 19.05.24
8	Оформлення матеріалів до захисту БДР	19.05.24 – 30.05.24

10. Порядок контролю та прийняття

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Б

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програмного забезпечення для соціальної мережі з функцією відображення геолокації користувачів

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доцент кафедри ПЗ Романюк О.В.

Оригінальність	93,9 %
Схожість	6,1 %

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи Кондратюк О.О.

Керівник роботи Романюк О.В.

ДОДАТОК В

Лістинг програми

```
        private void init() {
            locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
            myLocationListener = new MyLocationListener();
            myLocationListener.setLocalListenerInterface(this);
            locationService = new LocationServiceImpl();
            button = findViewById(R.id.button);
            button.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    checkForPermissions();
                }
            });
            StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
            StrictMode.setThreadPolicy(policy);
        }
        @Override
        protected void onCreate(Bundle saveInstance) {
            super.onCreate(saveInstance);
            setContentView(R.layout.main);
            init();
        }
        @Override
        public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
```

```
            super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
            if (requestCode == 100 ) {
               checkForPermissions();
            } else {
               Toast.makeText(this, "No GPS Permission",
Toast.LENGTH_SHORT).show();
            }
         }
        private void checkForPermissions() {
            if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED
                  ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED
                  ActivityCompat.checkSelfPermission(this,
Manifest.permission.INTERNET) != PackageManager.PERMISSION_GRANTED) {
               requestPermissions(new
String[]{Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION,
Manifest.permission.INTERNET}, 100);
            } else {
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1, 1,
myLocationListener);
            }
         }
         @Override
         public void onLocationChanged(Location location) {
```

```
            locationService.sendLocationInfo(location.getLatitude(),
location.getLongitude());
            Timer timer = new Timer();
            timer.scheduleAtFixedRate(new TimerTask() {
                @Override
                public void run() {
                    List<LocationDto> locations = getLocations();
                }
            }, 0, 5 * 60 * 1000);
        }
        public List<LocationDto> getLocations(){
            return locationService.getLocations();
        }
    }
    public class MyLocationListener implements LocationListener {
        private LocalListenerInterface localListenerInterface;
        public void setLocalListenerInterface(LocalListenerInterface
localListenerInterface) {
            this.localListenerInterface = localListenerInterface;
        }
        @Override
        public void onLocationChanged(@NonNull Location location) {
            localListenerInterface.onLocationChanged(location);
        }
        @Override
        public void onLocationChanged(@NonNull List<Location> locations) {
            LocationListener.super.onLocationChanged(locations);
        }
        @Override
        public void onFlushComplete(int requestCode) {
```

```
        LocationListener.super.onFlushComplete(requestCode);
    }
    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
        LocationListener.super.onStatusChanged(provider, status, extras);
    }
    @Override
    public void onProviderEnabled(@NonNull String provider) {
        LocationListener.super.onProviderEnabled(provider);
    }
    @Override
    public void onProviderDisabled(@NonNull String provider) {
        LocationListener.super.onProviderDisabled(provider);
    }
}
public class RetrofitUtil {
    public static Retrofit getRetrofitInstance(String url) {
        OkHttpClient okHttpClient = new OkHttpClient.Builder()
                .connectTimeout(15, TimeUnit.SECONDS)
                .writeTimeout(60, TimeUnit.SECONDS)
                .readTimeout(60, TimeUnit.SECONDS)
                .addInterceptor(chain -> {
                    int retryCount = 0;
                    long retryDelay = 2000;
                    int maxRetries = 5;
                    while (retryCount < maxRetries) {
                        try {
                            Request request = chain.request();
                            Response response = chain.proceed(request);
                            if (response.isSuccessful()) {
```

```
                    return response;
                } else {
                    response.close();
                }
            } catch (IOException e) {
                if (retryCount + 1 >= maxRetries) {
                    throw e;
                }
                try {
                    Thread.sleep(retryDelay);
                } catch (InterruptedException ie) {
                    Thread.currentThread().interrupt();
                    throw new IOException("Interrupted during request", ie);
                }
                retryDelay *= 2;
            }
            retryCount++;
        }
        throw new IOException("Maximum number of attempts
exceeded");
    })
    .build();
return new Retrofit.Builder()
    .baseUrl(url)
    .client(okHttpClient)
    .addConverterFactory(GsonConverterFactory.create())
    .build();
  }
}
public class LocationEncryptor {
```

```
        private String secretString = "";
        public List<byte[]> encryptData(String lon, String lat){
            List<byte[]> encrypted = new ArrayList<>();
            try {
                Cipher cipher = Cipher.getInstance("AES/CTR/NoPadding");
                SecretKey secretKey = new
SecretKeySpec(secretString.getBytes(StandardCharsets.UTF_8),
                        "AES");
                cipher.init(Cipher.ENCRYPT_MODE, secretKey);
                encrypted.add(cipher.doFinal(lon.getBytes()));
                encrypted.add(cipher.doFinal(lat.getBytes()));
            } catch (InvalidKeyException | NoSuchPaddingException |
NoSuchAlgorithmException |
                    IllegalBlockSizeException | BadPaddingException e) {
                throw new RuntimeException(e);
            }
            return encrypted;
        }
    }
    public class LocationDecryptor {
        private String secretString = "";
        public List<String> dencryptData(byte[] lon, byte[] lat){
            List<String> decrypted = new ArrayList<>();
            try {
                Cipher cipher = Cipher.getInstance("AES/CTR/NoPadding");
                SecretKey secretKey = new
SecretKeySpec(secretString.getBytes(StandardCharsets.UTF_8)
                        ,"AES");
                cipher.init(Cipher.DECRYPT_MODE, secretKey);
```

```
                String lonString = new String(cipher.doFinal(lon),
StandardCharsets.UTF_8);
                String latString = new String(cipher.doFinal(lat),
StandardCharsets.UTF_8);
                decrypted.add(lonString);
                decrypted.add(latString);
            } catch (InvalidKeyException | NoSuchPaddingException |
NoSuchAlgorithmException |
                    IllegalBlockSizeException | BadPaddingException e) {
                throw new RuntimeException(e);
            }
            return decrypted;
        }
    }
    public class LocationServiceImpl implements LocationService {
        private final LocationServiceApi locationServiceApi;
        public LocationServiceImpl() {
            Retrofit retrofit =
RetrofitUtil.getRetrofitInstance("https://philosophers.party/");
            this.locationServiceApi = retrofit.create(LocationServiceApi.class);
        }
        public void sendLocationInfo(Double latitude, Double longitude){
            LocationDto locationDto = new LocationDto();
            locationDto.setLatitude(latitude);
            locationDto.setLongitude(longitude);
            Call<Void> call = locationServiceApi.postLocation(locationDto);
            try {
                call.execute();
            } catch (IOException e) {
```

```
                throw new RuntimeException(e);
            }
        }
        @Override
        public List<LocationDto> getLocations() {
            Call<List<LocationDto>> call = locationServiceApi.getLocations();
            Response<List<LocationDto>> response;
            try {
                response = call.execute();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
            return response.body();
        }
    }
    public class LocationDto {
        private Double latitude;
        private Double longitude;
        public void setLatitude(Double latitude) {
            this.latitude = latitude;
        }
    public void setLongitude(Double longitude) {
            this.longitude = longitude;
        }
    }
    <manifest xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools" >
        <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/><uses-
permissionandroid:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

```
        <uses-permission android:name="android.permission.INTERNET"/>
        <application
            android:allowBackup="true"
            android:dataExtractionRules="@xml/data_extraction_rules"
            android:fullBackupContent="@xml/backup_rules"
            android:icon="@mipmap/ic_launcher"
            android:label="@string/app_name"
            android:roundIcon="@mipmap/ic_launcher_round"
            android:supportsRtl="true"
            android:theme="@style/Theme.PeopleNerby"
            tools:targetApi="31" >
            <activity
                android:name=".MainActivity"
                android:exported="true"
                android:label="@string/app_name"
                android:theme="@style/Theme.PeopleNerby" >
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />
                    <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
            </activity>
        </application>
</manifest>
@Service
@Slf4j
public class UserServiceImpl implements UserService {
    private final UserRepository userRepository;
    private final BCryptPasswordEncoder passwordEncoder;
    private final AuthenticationManager authenticationManager;
    private final JwtTokenProvider jwtTokenProvider;
```

```
    private final PictureService pictureService;
    private final SubscriptionRepository subscriptionRepository;
    private final PostRepository postRepository;
    private final String DIR = "C:\\Users\\alex\\Documents\\images";
@Autowired
    public UserServiceImpl(UserRepository userRepository, BCryptPasswordEncoder
passwordEncoder,
                           AuthenticationManager authenticationManager, JwtTokenProvider
jwtTokenProvider,
                           PictureService pictureService, SubscriptionRepository
subscriptionRepository,
                           PostRepository postRepository) {
        this.userRepository = userRepository;
        this.passwordEncoder = passwordEncoder;
        this.authenticationManager = authenticationManager;
        this.jwtTokenProvider = jwtTokenProvider;
        this.pictureService = pictureService;
        this.subscriptionRepository = subscriptionRepository;
        this.postRepository = postRepository;
    }
@Override
    @Transactional(rollbackFor = {ErrorCodeException.class})
    public void register(UserRegistrationRequestDto requestDto)
            throws EmailIsAlreadyTakenException, InvalidParamException, IOException
{
        log.info("Registering user");
        if (userRepository.findByEmail(requestDto.getEmail()).isPresent()) {
            throw new EmailIsAlreadyTakenException("Email is taken");
        }
```

```
        if (requestDto.getPassword() == null || requestDto.getEmail() == null ||
requestDto.getUsername() == null) {
            throw new InvalidParamException("Non of the fields can be blank");
        }
        User user = buildUser(requestDto);
        File avatarsDir = initFiles(user);
        if (!requestDto.getAvatar().isEmpty()&&requestDto.getAvatar()!=null) {
            pictureService.setAvatar(requestDto.getAvatar(), user.getId(),
avatarsDir.getPath());
        }
        user.setAvatar(avatarsDir.getPath());
    }
    private File initFiles(User user) {
        File userDir = new File(DIR + "/" + user.getId());
        userDir.mkdir();
        File postsDir = new File(DIR + "/" + user.getId() + "/" + "posts");
        postsDir.mkdir();
        File avatarsDir = new File(DIR + "/" + user.getId() + "/" + "avatars");
        avatarsDir.mkdir();
        return avatarsDir;
    }
    private User buildUser(UserRegistrationRequestDto requestDto) {
        log.info("Building user");
        User user = new User();
        user.setRole(Role.USER);
        user.setUsername(requestDto.getUsername());
        user.setPassword(passwordEncoder.encode(requestDto.getPassword()));
        user.setEmail(requestDto.getEmail());
        userRepository.save(user);
        log.info("User {} was saved", user);
```

```
        return user;
    }
    @Override
    public UserAuthorizationResponseDto login(UserAuthorizationRequestDto
requestDto) throws UserNotFoundException {
        log.info("Logging in");
        String email = requestDto.getEmail();
        User user = userRepository.findByEmail(email).orElseThrow(() -> new
UserNotFoundException("User with email: " + email + " not found"));
        authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(email, requestDto.getPassword()));
        String token = jwtTokenProvider.createToken(email, user.getRole());
        UserAuthorizationResponseDto response = new
UserAuthorizationResponseDto();
        response.setToken(token);
        log.info("User {} logged in", user);
        return response;
    }
    @Override
    public User update(UserUpdateRequestDto requestDto, Long userId) throws
InvalidParamException, IOException {
        log.info("Updating user");
        User user = userRepository.findById(userId).orElseThrow(() ->
                new UsernameNotFoundException("User with id: " + userId + " not
found"));
        if (requestDto.getEmail() == null || requestDto.getUsername() == null) {
            throw new InvalidParamException("Exception");
        }
        user.setEmail(requestDto.getEmail());
        user.setUsername(requestDto.getUsername());
```

```
        File avatarDir = new File(DIR + "/" + user.getId() + "/" + "avatars");
        if (!requestDto.getAvatar().isEmpty()) {
            pictureService.setAvatar(requestDto.getAvatar(), user.getId(),
avatarDir.getPath());
        }
        log.info("User {} updated", user);
        return user;
    }
    @Override
    @Transactional
    public void deleteUser(Long userId) {
        log.info("Deleting user");
        User user = userRepository.getById(userId);
//        List<Post> posts = postRepository.findAllByUserId(user.getId()).orElse(new
ArrayList<>());
//        if (!posts.isEmpty()) {
//            for (Post post : posts) {
//                postRepository.delete(post);
//                List<Comment> comments =
commentRepository.findAllByContentTypeAndContentId(ContentType.POST,
post.getId()).get();
//                comments.addAll(comments.stream().flatMap(c ->
commentRepository.findAllByContentTypeAndContentId(ContentType.COMMENT,
c.getId()).orElse(new ArrayList<>()).stream()).collect(Collectors.toList()));
//                commentRepository.deleteAll(comments);
//            }
//        }
//        userRepository.delete(user);
        user.setUsername("DELETED");
```

```
        user.setEmail("DELETED");
        log.info("User {} was deleted", user);
    }
    @Override
    @Transactional
    public void deleteUserById(Role role, Long userId) throws
UserNotFoundException {
        log.info("Deleting user by id");
        if (role.equals(Role.ADMIN)) {
            User userToDelete = userRepository.findById(userId).orElseThrow(() -> new
UserNotFoundException("User with id: " + userId + " not found"));
            userToDelete.setEmail("DELETED");
            userToDelete.setUsername("DELETED");
            log.info("User {} was deleted", userToDelete);
        }
    }
    @Override
    public UserResponseDto showUserById(Long userId) throws
UserNotFoundException {
        log.info("Showing user by id");
        User user = userRepository.findById(userId).orElseThrow(() -> new
UserNotFoundException("User with id: " + userId + " not found"));
        UserResponseDto userResponseDto = new UserResponseDto();
        userResponseDto.setId(user.getId());
        userResponseDto.setUsername(user.getUsername());
        userResponseDto.setEmail(user.getEmail());
        userResponseDto.setAvatar(user.getAvatar());
        userResponseDto.setRole(user.getRole());
userResponseDto.setSubscribers(subscriptionRepository.countAllBySubscribedToId(u
serId));
```

```
userResponseDto.setSubscriptions(subscriptionRepository.countAllBySubscriberId(us
erId));
        userResponseDto.setPosts(postRepository.countAllByUserId(userId));
        log.info("User {} was shown", user);
        return userResponseDto;
    }
    @Override
    public User findByEmail(String email) throws UserNotFoundException {
        log.info("Finding user by email");
        User  user = userRepository.findByEmail(email).orElseThrow(() ->
                new UserNotFoundException("User with email: " + email + " not found"));
        log.info("User {} was found", user);
        return user;
    }
    @Override
    public User findById(Long id) throws UserNotFoundException {
        log.info("Finding user by id");
        User  user = userRepository.findById(id).orElseThrow(() ->
                new UserNotFoundException("User with id: " + id + " not found"));
        log.info("User {} was found", user);
        return user;
    }
}
@Service
@Slf4j
public class NotificationServiceImpl implements NotificationService {
    private final NotificationRepository notificationRepository;
    private final SponsoredPostRepository sponsoredPostRepository;
    private final UserRepository userRepository;
    @Autowired
```

```
    public NotificationServiceImpl(NotificationRepository notificationRepository,
SponsoredPostRepository sponsoredPostRepository, UserRepository userRepository) {
        this.notificationRepository = notificationRepository;
        this.sponsoredPostRepository = sponsoredPostRepository;
        this.userRepository = userRepository;
    }
    @Override
    public List<NotificationResponseDto> getNotifications(Long userId) {
        log.info("Getting notifications for user with id: {} ", userId);
        List<Notification> notifications =
notificationRepository.findByRecipientId(userId);
        List<NotificationResponseDto> responseDtos = new ArrayList<>();
        for (Notification notification : notifications) {
            NotificationResponseDto responseDto = new NotificationResponseDto();
            responseDto.setMessage(notification.getMessage());
            responseDto.setId(notification.getId());
            responseDto.setDate(notification.getDate());
            responseDtos.add(responseDto);
        }
        log.info("Notifications for user with id: {} are: {}", userId, responseDtos);
        return responseDtos;
    }
    @Override
    public void notificationCommentLike(Long commentId, String username, Long
recipientId) throws UserNotFoundException {
        log.info("Creating notification about comment like");
        User recipient = userRepository.findById(recipientId).orElseThrow(() -> new
UserNotFoundException("User with id: " + recipientId + " not found"));
        Notification notification = new Notification();
        notification.setMessage(username + " liked your comment");
```

```
        notification.setRecipient(recipient);
        notification.setDate(LocalDate.now());
        notificationRepository.save(notification);
        log.info("Notification about comment like {} was created",notification);
    }
    @Override
    public void notificationPostLike(Long postId, String username, Long recipientId)
throws UserNotFoundException {
        log.info("Creating notification about post like");
        User recipient = userRepository.findById(recipientId).orElseThrow(() -> new
UserNotFoundException("User with id: " + recipientId + " not found"));
        Notification notification = new Notification();
        notification.setMessage(username + " liked your post");
        notification.setRecipient(recipient);
        notification.setDate(LocalDate.now());
        notificationRepository.save(notification);
        if (sponsoredPostRepository.existsByPostId(postId)) {
            SponsoredPost sponsoredPost =
sponsoredPostRepository.findByPostId(postId).get();
            Notification notificationToSponsor = new Notification();
            notificationToSponsor.setMessage(username + " liked post you sponsored");
            notificationToSponsor.setRecipient(sponsoredPost.getSponsor());
            notificationToSponsor.setDate(LocalDate.now());
            notificationRepository.save(notificationToSponsor);
        }
        log.info("Notification about post like {} was created",notification);
    }
    public void sendNotificationCommentPost(String commentatorUsername, User
recipient, Long postId) {
        log.info("Creating notification about comment post");
```

```
        Notification notification = new Notification();
        notification.setMessage(commentatorUsername + " commented your post");
        notification.setRecipient(recipient);
        notification.setDate(LocalDate.now());
        notificationRepository.save(notification);
        if (sponsoredPostRepository.existsByPostId(postId)) {
            SponsoredPost sponsoredPost =
sponsoredPostRepository.findByPostId(postId).get();
            Notification notificationToSponsor = new Notification();
            notificationToSponsor.setMessage(commentatorUsername + " commented post
you sponsored");
            notificationToSponsor.setRecipient(sponsoredPost.getSponsor());
            notificationToSponsor.setDate(LocalDate.now());
            notificationRepository.save(notificationToSponsor);
        }
        log.info("Notification about comment post {} was created",notification);
    }
    public void sendNotificationCommentComment(String commentatorUsername,
User recipient) {
        log.info("Creating notification about comment comment");
        Notification notification = new Notification();
        notification.setMessage(commentatorUsername + " commented your comment");
        notification.setRecipient(recipient);
        notification.setDate(LocalDate.now());
        notificationRepository.save(notification);
        log.info("Notification about comment comment {} was created",notification);
    }
}
```

ДОДАТОК Г

Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СОЦІАЛЬНОЇ МЕРЕЖІ З ФУНКЦІЄЮ ВІДОБРАЖЕННЯ ГЕОЛОКАЦІЇ КОРИСТУВАЧІВ

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмногозебезпечення

Бакалаврська дипломна робота на тему:

«Розробка програмного забезпечення для соціальної мережі з функцією відображення геолокації користувачів»

Автор: ст. гр. 1ПІ20б Кондратюк О.О.
Науковий керівник:к.т.н., доц. каф. ПЗ Романюк О.В.

Вінниця – 2024

Рисунок Г.1 – Титульний слайд

Актуальність теми

У сучасному цифровому світі соціальні мережі відіграють важливу роль у спілкуванні та взаємодії між людьми. З кожним днем їх вплив на наше життя лише зростає, надаючи нові можливості для спілкування, обміну інформацією та координації дій. У цьому контексті актуальною стає розробка програмних засобів, які розширюють можливості соціальних мереж та забезпечують користувачів новими функціями .

Однією з таких потенційно цікавих функцій є відображення місцезнаходження людей поруч з користувачем. Це може стати корисним інструментом для планування зустрічей, організації подій або просто для того, щоб знати, де перебувають ваші друзі чи колеги.

Рисунок Г.2 – Актуальність теми

Можливості, що відкриваються

1. Покращення взаємодії користувачів: Забезпечення можливості відображення місцезнаходження інших користувачів поруч з поточним користувачем дозволить покращити комунікацію та взаємодію між ними. Це особливо важливо для організації зустрічей, подій чи спільних заходів.
2. Підвищення безпеки та контролю: Можливість відслідковувати місцезнаходження друзів чи родичів може бути корисною для забезпечення їх безпеки та контролю за їхнім розташуванням, особливо у випадку екстрених ситуацій.
3. Створення нових сервісів та функцій: Розробка програмного засобу для цієї функції відкриває широкі можливості для створення нових сервісів та функцій в соціальних мережах, які можуть бути корисними для різних груп користувачів, включаючи бізнес, туризм, освіту тощо.

Рисунок Г.3 – Можливості, що відкриваються

Мета, об’єкт та предмет дослідження

Метоюроботи є покращення користувацького досвіду взаємодії з соціальною мережею за рахунок розробки програмного забезпечення для соціальної мережі з функцією відображення геолокації користувачів

Об’єктом дослідження є процес розробки програмних модулів для програмного забезпечення для соціальної мережі з функцією відображення геолокації користувачів.

Предметомдослідження є методи і засоби реалізації програмного забезпечення для реалізації в соціальній мережі функції відображення геолокації людей поруч з користувачем.

Рисунок Г.4 – Мета, об’єкт та предмет дослідження

Задачі дослідження

- здійснити аналіз стану питання та методів розв'язання задачі;
- спроектувати модель мобільного застосунку;
- розробити метод формування стрічки рекомендацій користувача;
- розробити алгоритм відслідковування геолокації користувачів та загальний алгоритм роботи програми;
- розробити базу даних, яка буде забезпечувати збереження інформації про користувачів та їх місцезнаходження.
- розробити графічний інтерфейс мобільного застосунку;
- розробити модуль для відображення місцезнаходження, який буде отримувати дані про місцезнаходження користувачів і відображати їх на графічному інтерфейсі;
- розробити модуль соціальної мережі;
- провести тестування програмного застосунку;
- розробити інструкцію користувача та описати системні вимоги програми для відстеження читання.

Рисунок Г.5 – Задачі дослідження

Новизна і практична цінність отриманих результатів

Новизною отриманих результатів :

1. Удосконалено систему відслідковування геолокації користувачів, яка, на відміну від відомих, реалізована у вигляді окремого модуля на окремому сервері, що дало можливість підвищити гнучкість її інтеграції, швидкість роботи та користувацький досвід.
2. Уперше запропоновано метод формування стрічки рекомендацій користувача, особливістю якого є використання штучного інтелекту, що дало можливість підвищити релевантність рекомендацій відповідно до інтересів і потреб користувачів.

Практична цінність отриманих результатів полягає в тому, щоб на основі отриманих в ході виконання бакалаврської дипломної роботи теоретичних положень розроблено алгоритми і програмне забезпечення для соціальної мережі з функцією відображення місцезнаходження людей поруч з користувачем.

Рисунок Г.6 – Новизна і практична цінність отриманих результатів

Порівняльний аналіз аналогів

Критерії	Life360	Glympse	Foursquare Swarm	Власний програмний засіб
Простота використання	+	+	-	+
Споживання ресурсів	-	+	+	+
Відстеження точного місцезнаходження	+	-	-	+
Функціонал характерний соціальній мережі	-	-	+	+
Доступність всього функціоналу безкоштовно	-	-	+	+
Швидкість роботи	+	-	+	+
Загальна оцінка	3	2	4	6

Рисунок Г.7 – Порівняльний аналіз аналогів

Діаграма сутностей

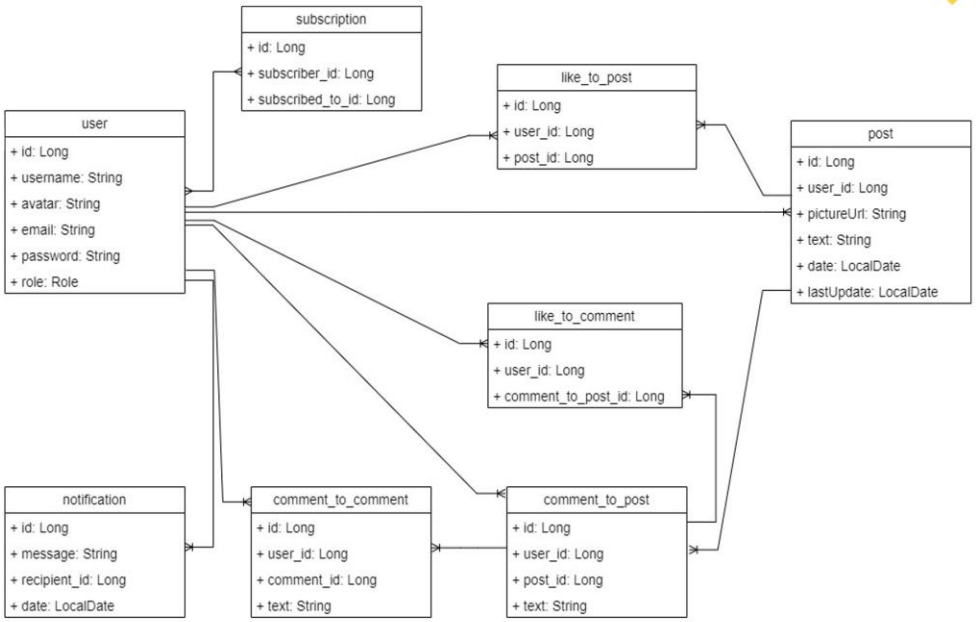

Рисунок Г.8 – Діаграма сутностей

Алгоритм відстеження геолокації

Згідно з діаграмою, перший крок – це автентифікація користувача, щоб вони могли увійти до системи і скористатися її функціоналом.
Після успішної автентифікації система перевіряє, чи надав користувач доступ до своєї геолокації .
Якщо доступ до геолокації надано, користувач може перейти на екран з мапою, на якій показані місцезнаходження інших користувачів.

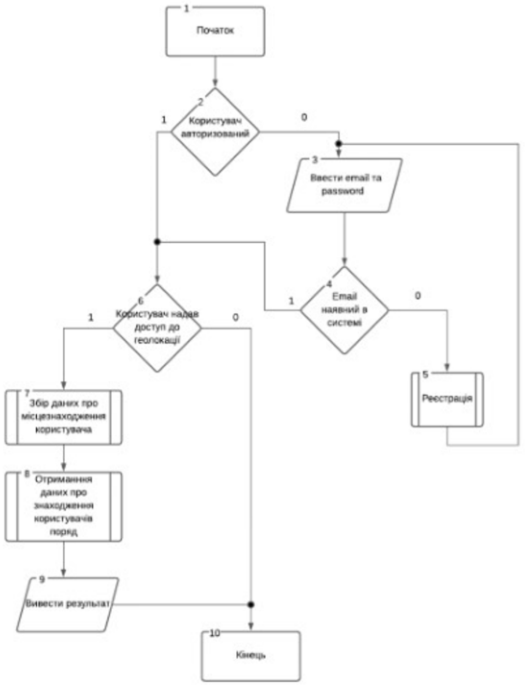

Рисунок Г.9 – Алгоритм відстеження геолокації

Алгоритм формування стрічки рекомендацій користувача

Штучний інтелект отримує інформацію про дописи, що були створені за останній час, а також інформація про вподобання користувача, яка формується на основі його активності. Після аналізу цих даних штучний інтелект надає список тих дописів, що будуть цікаві користувачеві.

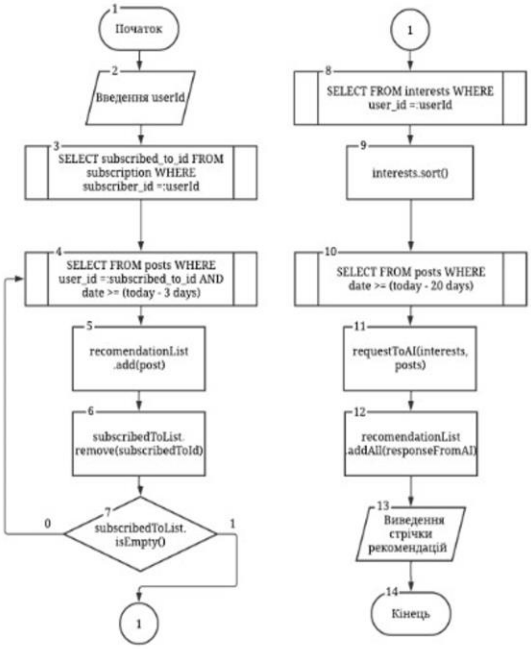

Рисунок Г.10 – Алгоритм формування стрічки рекомендацій користувача

Технологічний стек

Для розробки програмного продукту були обрані мова програмування Java, Spring Framework бібліотека Retrofit, та хмарні сервіси AWS.

Java виявляється ідеальним вибором, оскільки вона забезпечує потужність та гнучкість для створення високопродуктивних та масштабованих систем.

Бібліотека Retrofit обрана з метою забезпечення простоти, ефективності та надійності у взаємодії з сервером соціальної мережі. Вона надає зручний та зрозумілий спосіб виконання HTTP-запитів та обробки відповідей в форматах JSON або XML.

Фреймворк Spring забезпечує високорівневу інфраструктурну підтримку, включаючи управління залежностями, управління транзакціями, взаємодію з базами даних, роботу з веб-застосунками та інші необхідні функції .

Рисунок Г.11 – Технологічний стек

Екран реєстрації застосунку

Екран реєстрації є ключовою частиною будь-якого додатку, тому що він дозволяє користувачам створювати облікові записи і отримувати доступ до функціональності системи, у застосунку було спроектовано та імлементовано екран реєстрації.

Обмеження та перевірки на екрані реєстрації:
- перевірка правильності формату електронної пошти;
- перевірка унікальності електронної пошти;
- перевірка наявності обов'язкових полів.

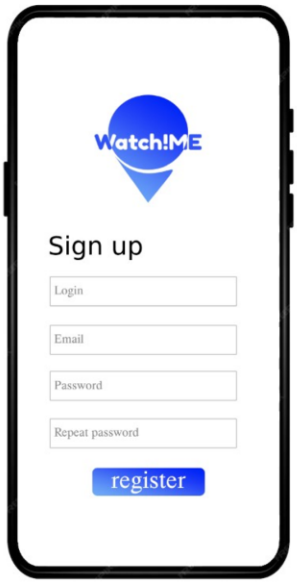

Рисунок Г.12 – Екран реєстрації застосунку

Екран з мапою

Після того як користувач надав застосунку дозвіл для отримання даних про геолокацію він може перейти на екран з мапою, на якому буде відображатись інформація про геолокацію інших користувачів.

Користувач може бачити місцезнаходження тільки тих користувачів, що надали йому для цього доступ.

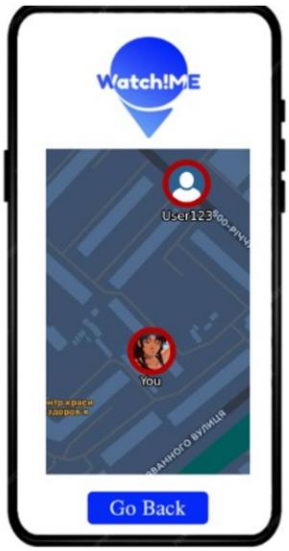

Рисунок Г.13 – Екран з мапою

Тестування програми

Було проведено модульне тестування програми. Один із методів тестування програмного забезпечення це модульне тестування. Модульне тестування є одним із найважливіших аспектів забезпечення якості програмного забезпечення. Це метод тестування програмного забезпечення, при якому окремі модулі або компоненти програми перевіряються на коректність функціонування. Модульне тестування дозволяє виявити помилки на ранніх стадіях розробки, що зменшує витрати на їх виправлення в майбутньому.

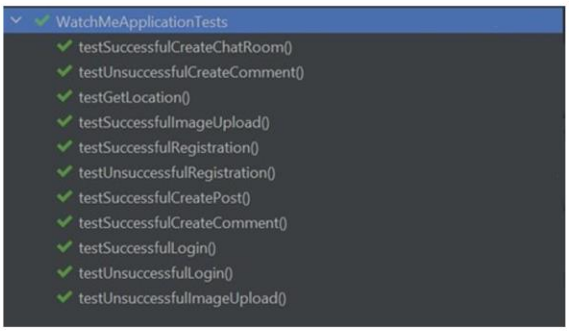

Рисунок Г.14 – Тестування програми

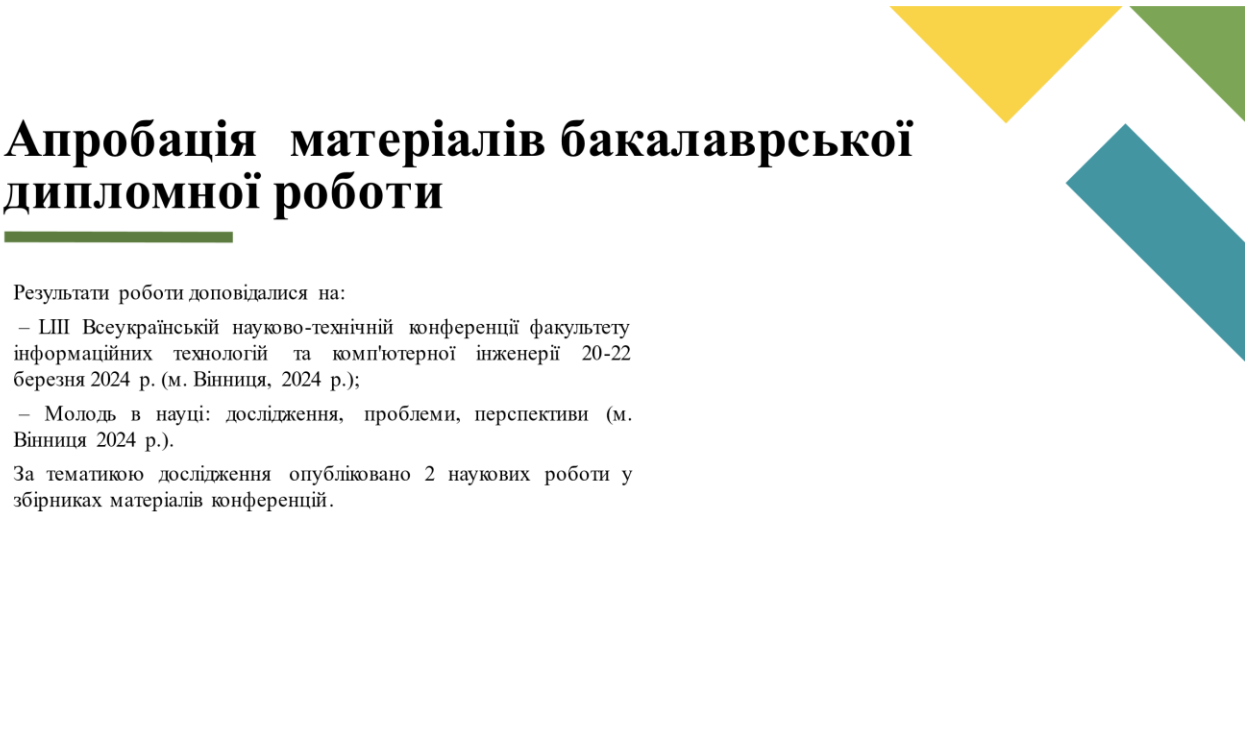

Рисунок Г.15 – Апробація матеріалів бакалаврської дипломної роботи

Рисунок Г.16 – Фінальний слайд