МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Мобільний додаток з пошуку шляху до найближчого укриття»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м-12 Меший Тихон Сергійович

Кваліфікаційну роботу
захищено на засіданні ЕК
з оцінкою _____ «___» грудня 2022 р.

Науковий керівник _____ к.т.н., доц., Баранова І.В.
(підпис)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2022

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В.о. зав. кафедри ІТ

С. М. Ващенко
«___» _____ 2022 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Меший Тихон Сергійович
(прізвище, ім'я, по батькові)

1 Тема проекту *Мобільний додаток з пошуку шляху до найближчого укриття*

затверджена наказом по університету від «_04_» _11__2022 р. № 01013-VI

2 Термін здачі студентом закінченого проекту «___» ___грудня___ 2022 р.

3 Вхідні дані до проекту Необхідний функціонал на розробку мобільного додатку з пошуку шляху до найближчого укриття

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз програмних продуктів – аналогів, постановка задачі, засоби реалізації, створення додатку з пошуку маршруту,

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність роботи, постановка задачі, аналіз програмних продуктів – аналогів, IDEF0-діаграма та її декомпозиція, діаграма варіантів використання, практична реалізація, демонстрація додатку

6 Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання 27.08.2022.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Аналіз предметної області	01.10.2022	
2	Аналіз потреб	03.10.2022	
3	Аналіз конкурентів	05.10.2022	
4	Аналіз актуальності	07.10.2022	
5	Створення проектної частини	09.10.2022	
6	Написання чек-листів	13.10.2022	
7	Побудова діаграми Ганта	15.10.2022	
8	Планування ризиків	20.10.2022	
9	Написання коду	22.10.2022	
10	Пошук найкоротшого шляху	08.11.2022	
11	Тестування додатку	10.11.2022	
12	Тестування найкоротшого шляху	26.11.2022	

Магістрант _____ Меший Т.С.

Керівник роботи _____ к.т.н., доц. Баранова І.В.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Мобільний додаток з пошуку шляху до найближчого укриття».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 30 найменувань, додатків. Загальний обсяг роботи − 128 сторінок, у тому числі 112 сторінок основного тексту, 2 сторінки списку використаних джерел, 6 сторінок додатків.

Кваліфікаційну роботу магістра присвячено розробці мобільного додатку для пошуку шляху до найближчого укриття.

В роботі проведено аналіз аналогічних застосунків та необхідних функцій додатку.

Визначено які технології використовувати при створення додатку.

Проведено роботу з написання мобільного додатку на мові java.

Результатом проведеної роботи є повністю функціонуючий мобільний додаток.

Практичне значення роботи полягає наданні найближчого маршруту до бомбосховища.

Ключові слова: мобільний додаток, пошук шляху, сховище, java.

ЗМІСТ

ВСТУП

Смартфони та інші мобільні пристрої стали не лише частиною нашого повсякденного життя, але й його повним продовженням. За допомогою мобільних телефонів ми можемо не тільки спілкуватися один з одним, але й робити замовлення в магазинах, купувати квитки, бронювати житло, викликати таксі, орієнтуватися за допомогою мобільних телефонів, фотографувати та знімати відео, читати, онлайн-банкінг і просто як вид задоволення та спосіб провести час. Згідно зі статистикою, опублікованою Datareportal, 67% дорослих у світі щодня користуються смартфоном, а це майже 5,19 мільярда людей (із загального населення 7,75 мільярда) [1].

Тренд від простих мобільних пристроїв до багатофункціональних смартфонів зростає з кожним роком. Припустимо, що в недалекому майбутньому людина буде напряму підключена до свого смартфона, тобто всі функції і вся інформація наших гаджетів потраплять прямо в наш мозок.

Поки багато винахідників працюють над цим, час витрачається на, здавалося б, непотрібні додатки, такі як низькочастотні репеленти від комарів, індикатори стиглості кавунів, симулятори бульбашок або візуалізація екранів смартфонів у вигляді додатків для чашок пива.

Коротше кажучи, актуальність розробки мобільних додатків зростає не лише щороку, а щомісяця. Щодня на онлайн-платформах з’являються сотні нових мобільних додатків. Може здатися, що важко придумати щось нове, але хороший розвиток у поєднанні з хорошим маркетинговим партнерством може принести багато грошей творцям, про що свідчить гучний старт TikTok і постійне стрімке зростання глядачів.

В сучасному світі дуже цінується людське життя, а зараз в Україні йде війна і гине багато людей. Наш ворог завдає удари не тільки по військовим, а й по

цивільному населенню. І багато з них не знає про наявні укриття для цивільного населення.

В період сьогодення в Україні є актуальним додаток пошуку укриття. У ці важкі часи ІТ-підрозділи намагаються функціонувати, захищати націю та виконувати роботу, необхідну для вирішення проблем сучасного життя. Оскільки мало хто знає як поводити себе підчас обстрілу.

Метою є створення мобільного додатку для пошуку укриття.

Для досягнення мети необхідно вирішити такі задачі:

– провести аналіз предметної області;

– сформулювати постановку задачі;

– виконати проектування всіх необхідних діаграм та моделей використання;

– розробити інтерфейс та функціонал мобільного додатку з пошуку шляху до найближчого укриття.

Об’єкт дослідження: пошук шляху до найближчого укриття.

Предмет дослідження: мобільний додаток для пошуку шляху.

Практичне значення роботи: в результаті використання розробленого додатку людина швидше знайде найближче укриття.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень та публікацій

Розглянемо питання, що таке пошук шляху. Пошук шляху – це використання комп’ютерної програми для побудови найкоротшого шляху між двома точками. Це більш практичний варіант вирішення задач лабіринту. Ця галузь дослідження базується на алгоритмі Дейкстри для пошуку найкоротших шляхів на зважених графах [2].

Алгоритм Дейкстри – алгоритм на графах, який знаходить найкоротший шлях від однієї вершини графу до всіх інших вершин. Класичний алгоритм Дейкстри працює тільки для графів без ребер від'ємної довжини [3].

Для пошуку оптимального шляху потрібно дві точки: початок та кінець. Початкову точку буде задавати геолокація.

Геолокація – Одна з найпопулярніших функцій мобільних додатків. Якщо звернути увагу, використовуються не тільки карти і навігатори, але навіть камери мобільних телефонів використовують цю функцію. Крім того, є багато програм, де геолокація є не доповненням, а основною функцією.

Для визначення розташування користувача пристрій може використовувати різні послуги: GPS, LBS, WiFi, Cell ID, BLE тощо [4].

Розробка сервісів з геолокацією потребує особливої уваги та підходу, адже це може виявитися непростим завданням. Щоб користувацький досвід був позитивним і додаток приносив прибуток, необхідно ретельно продумати деталі та врахувати всі можливі труднощі. Серед аспектів, що вимагають окремої уваги при розробці, виділимо такі [4]:

– □ Використання ресурсів пристрою: програми геолокації можуть бути

погано оптимізовані та працювати надто повільно. Це негативно впливає на досвід користувача;

– Автономність роботи: при виборі технології геолокації необхідно враховувати «живучість» смартфона – акумулятор і енергоефективність. GPS-модуль може швидко розрядити пристрій, тому необхідно передбачити можливість запуску програми без підключення до мережі;

– Деталі інтерфейсу: оскільки деякі програми геолокації (наприклад, карти) містять велику кількість інформації в різних масштабах, дизайн таких інтерфейсів вимагає особливої уваги;

– Оперативна перевірка: часто важко перевірити роботу геолокації до пізніх етапів розробки. В результаті тестування тривало навіть після релізу, що ускладнювало перші місяці роботи та спілкування з користувачами;

– Кросплатформенність: розробка загальної програми геолокації для Android та IOS може бути ускладнена проблемами інтеграції з рідними API карт. Іноді найкращим варіантом є локальна розробка для кожної платформи [4].

1.2 Аналіз існуючих аналогів

На початку роботи проведено аналіз існуючих проектів, в ході якого були виявлені їх переваги та недоліки.

Перший аналог “Київ цифровий” [5]. Додаток містить корисні посилання на карти аварійних укриттів, інструкції щодо дій під час артобстрілу тощо. Додаток дозволяє користувачам заряджати свої транспортні картки, купувати QR-квитки, оплачувати паркування та повертати евакуйовані транспортні засоби зі своїх

смартфонів. Отримуйте сповіщення про житлово-комунальні послуги, голосуйте за проекти благоустрою міста тощо.Станом на квітень 2022 року у додатку, окрім загальноміської інформації, з'явилася низка функціоналу, пов'язана з військовими діями на території України (рис. 1.1). Серед нововведень:

- мапа укриттів;
- мапа працюючого бізнесу;
- волонтерська допомога армії;
- волонтерська допомога киянам;
- посилання на офіційні джерела інформації;
- повідомлення тривоги у додатку.

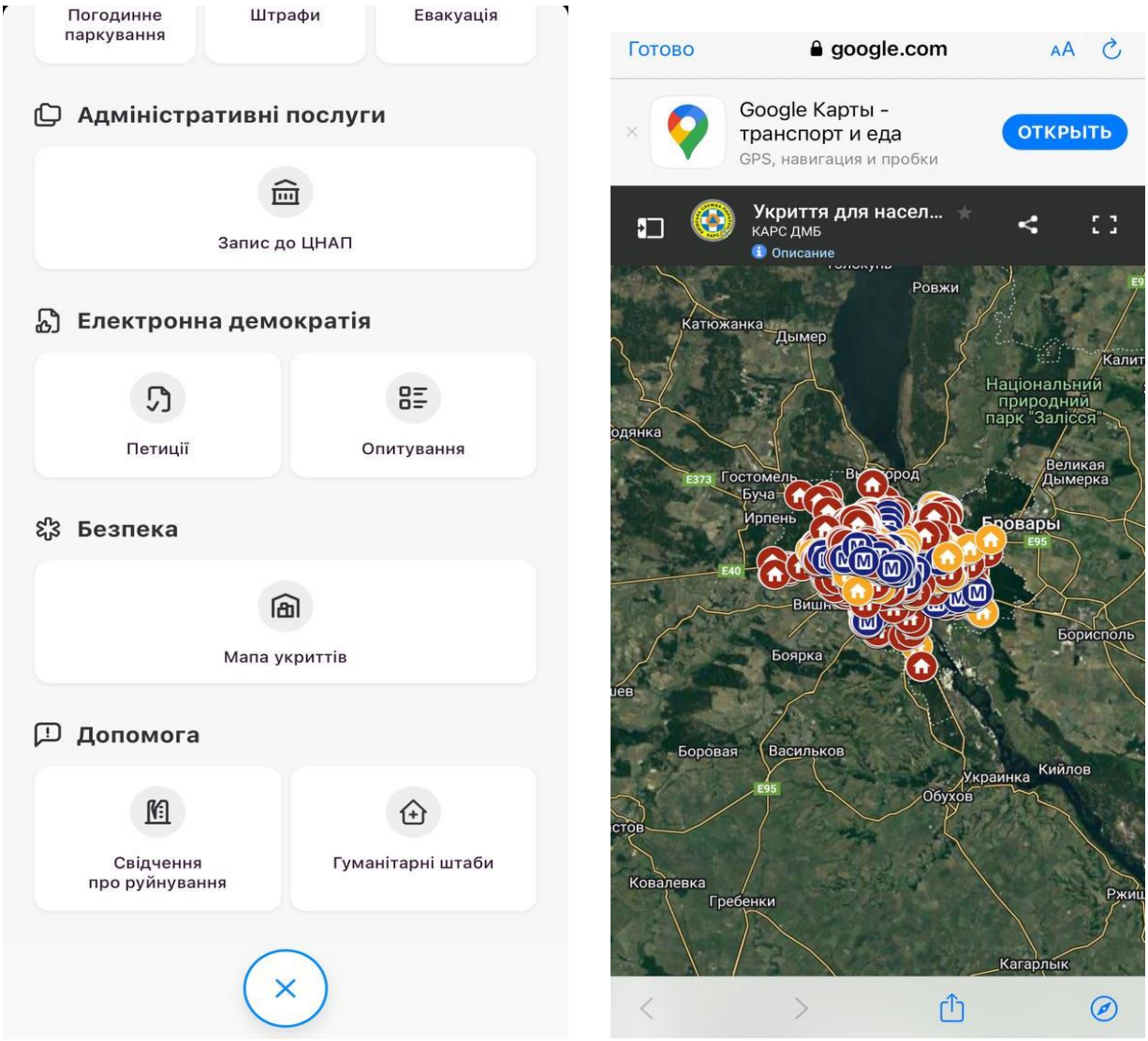

Рисунок 1.1 – Загальний план мобільного додатку

Після аналізу функціоналу додатку можна сказати, що перевагою його є великий спектр функцій, є карта укриттів. Недоліком є те, що при роботі додаток має потребу в з’єднанні з інтернетом, та задіює багато ресурсів.

Інший аналог – мобільний додаток «Прилуки –карта захисних споруд» [6] з місцями розташування захисних споруд в місті. Встановивши застосунок на смартфон, у випадку повітряної тривоги люди Прилук зможуть швидко визначити, де є найближче укриття, та як до нього дістатися найкоротшим шляхом. Додаток розраховує відстань та час, який знадобиться користувачу, щоб дістатися до

сховища. Дані щодо розташування укриттів автор брав з офіційного сайту Прилуцької міської ради та департаменту цивільного захисту обласної військової адміністрації. Клопіт мав лише з окремими адресами сховищ.

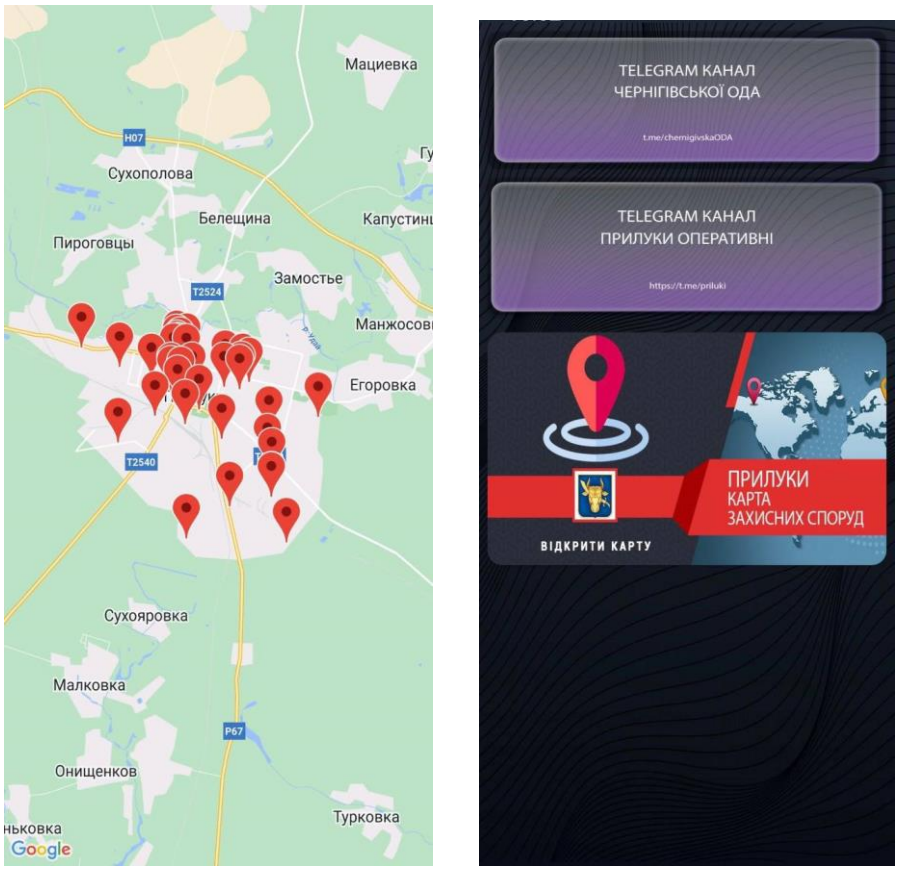

Рисунок 1.2 – Загальний план мобільного додатку

Перевагою цього додатку є те, що має тільки мапу укриттів, швидко працює, не потребує багато ресурсів. Недоліком є те, що працює тільки за допомогою інтернету та є тільки на платформі Android.

Результат порівняння аналогів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

№	Назва критерію	Назва платформи		
		Київ-	Прилуки –карта	Мапа укриттів

		цифровий	захисних споруд	Суми
1	Швидкість роботи	+	+	+
2	Вказує вашу геолокацію	+	-	+
3	Наявність карти міста Суми	-	-	+
4	Ресурсозатратність	+	-	-

В результаті аналізу предметної області та аналогів зроблено висновок про необхідність розробки власного мобільного додатку. Розроблюваний додаток “Мапа укриттів Суми” не потребуватиме багато ресурсів, матиме мапу укриттів, буде швидко працювати.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Постановка задачі

Мета роботи полягає у розробці мобільного додатку для пошуку шляху до найближчого укриття для жителів міста Суми.

Для досягнення поставлених цілей необхідно поетапно вирішити наступні окремі завдання:

- Проаналізувати існуючі публікації та подібні проекти, щоб визначити характеристики дизайну;

- сформулювати необхідні задачі, визначити програмні продукти та технології потрібні для реалізації проекту;

- провести структурно-функціональний аналіз виконання проекту, розробити контекстну діаграму, діаграму декомпозиції та варіантів використання;

- розробити структуру додатку та створити сам додаток ;

Для досягнення поставленої мети додаток має включати такий функціонал:

– визначення місця геолокації користувача;

– прокладання маршруту до найближчого укриття;

– зворотній зв'язок;

– наявність карти сховищ та укриттів.

2.2 Технології та інструменти реалізації

Будь-яка мобільна програма на основі визначення місцезнаходження може працювати лише з актуальною картою та службами визначення місцезнаходження.

Для більш точної геолокації потрібно реалізувати декілька технологій. Розглянемо їх [7-8]:

– GPS модуль. Термін «GPS» означає глобальну систему позиціонування – технологію супутникової навігації, яка надає дані про геолокацію та час. Найновіша технологія GPS пропонує точні геолокаційні дані з точністю до кількох метрів. Можна завантажити навігатори з вбудованими GPS координатами.

– Стільниковий ідентифікатор. Стільниковий ідентифікатор (ідентифікація) є важливим компонентом геолокації, оскільки він унікальний для кожного пристрою. І навіть за відсутності живих даних з мобільного пристрою, інформація зі стільникових веж може надати приблизне місцезнаходження. Тому вам не потрібне з'єднання Wi-Fi, щоб знайти ваше поточне місцезнаходження.

– Допоміжний GPS. Допоміжний або доповнений GPS (A-GPS) – це окрема система, яка покращує загальну продуктивність технології позиціонування GPS. Поєднання Cell ID і A-GPS забезпечує більш точне відстеження місцезнаходження, яке замінює продуктивність стандартного GPS.

– Геозонування. Геозона – це віртуальна межа, у межах якої програма виконує просту або складну заздалегідь запрограмовану дію. Такі компанії, як Uber, використовують техніку геозонування. Подібним чином програмне забезпечення для моніторингу дітей використовує адаптовану версію геозонування для відстеження пересування маленьких дітей.

– iBeacon і Eddystone. iBeacon – це датчик наближення з низьким енергоспоживанням на основі Bluetooth, який передає унікальний сигнал від маяків на приймач (зазвичай додаток). Eddystone – це версія iBeacon від Google. Технологія названа на честь однойменного маяка в Англії [9].

Google об'єднав всі свої API в карти, маршрути та місця. Це рішення спростило інтеграцію рішень геолокації в розробку програмного забезпечення. Але розробники можуть отримати доступ до цих API лише за допомогою унікального ключа автентифікатора, який надають лише адміністратори від Google [10].

2.2.1 Google Maps API

Google Maps API дозволяє додавати карти до програм. Карти також гарантують, що ви можете змінювати відображення за допомогою вбудованого панорамного перегляду [10].

Функціональність рішення:

– Maps SDK – цей API на базі Android дозволяє додавати дані з Карт Google у вашу програму. Maps SDK забезпечує автоматичний доступ до серверів Google, реагування на дії з картою та завантаження.

– Maps JavaScript API – цей унікальний API дозволяє розробникам додавати власний графічний вміст на свої пристрої та веб-платформи. JavaScript API містить чотири категорії карт, які можуть бути налаштовані: супутник, дорожня карта, рельєф і гібрид.

– Maps Embed API – за допомогою цього API ви можете додати інтерактивну карту на свій веб-сайт за допомогою базового HTTP-запиту. Maps Embed API відрізняється від JavaScript API, оскільки ви можете налаштувати його без жодного досвіду Javascript.

– Maps Static API – цей API дозволяє додавати будь-яку графіку Google Maps на ваш веб-сайт без динамічного завантаження сторінки. Все, що вам потрібно, це надіслати HTTP-запит, який повертає збіг, який ви можете відобразити на своїй платформі.

– Street View Static API –цей API надає вам доступ до оновлених реальних зображень місць. Ви можете вставити API як мініатюру на свій сайт.

– URL- адреси карт –цей інструмент генерує міжплатформне посилання, яке може отримувати маршрути та відображати панорамні види.

2.2.2 Маршрути

Цей API дозволяє отримувати точні вказівки та активні маршрути між заданими місцями на карті. Ви також можете використовувати цей інструмент для моніторингу оновлених даних про затори на маршруті та аварій у реальному часі.

– Directions API – за допомогою Google Maps Direction API ви можете отримати маршрути залежно від обраного способу транспорту. Сервіс також обчислює відстань між місцями за допомогою HTTP-запиту, створеного на основі даних користувача.

– Roads API – цей інструмент надає конкретні дані про пройдені дороги, як-от обмеження швидкості та альтернативні маршрути.

– Distance Matrix API – цей сервіс розраховує відстань між маршрутними точками на карті та скільки часу знадобиться для її подолання. Google Distance Matrix API працює разом з Directions API, щоб надавати точні оцінки часу відправлення та прибуття.

2.2.3 Places API

Places API надає користувачам доступ до понад 100 мільйонів місць. Це також дозволяє їм знаходити місця за допомогою GPS-адрес і номерів телефонів. Користувачі фотолокатора також можуть скористатися перевагами цього API для навігації поза приміщеннями.

– API геокодування – ця служба перетворює довгі фізичні адреси в географічні висоти та координати (широти та довготи). По суті, ця функція дозволяє користувачам поставити точку в додатку та отримати повну адресу.

– API геолокації – цей ефективний інструмент геолокації перевіряє дзвінки стільникового зв’язку та використовує інформацію, щоб визначити місцезнаходження в межах визначеного периметра.

– API часового поясу – цей інтерфейс дозволяє користувачам дізнаватися часові пояси різних місць у всьому світі. Цей API надає часовий пояс і поточний час у UTC і літній час.

– Бібліотека місць – цей API дозволяє програмам шукати місця та визначні пам’ятки. Функції автозаповнення надають відповідні пропозиції для кращої взаємодії з користувачем.

2.2.4 Засоби реалізації

Для реалізації додатку буде використовуватись програма Android Studio [11].

Android Studio – це офіційне інтегроване середовище розробки (IDE) для розробки додатків Android на основі IntelliJ IDEA [12]. Окрім потужного редактора коду IntelliJ та інструментів розробника, Android Studio пропонує ще більше функцій, які підвищують продуктивність під час створення програм для Android, наприклад:

– гнучка система збирання на основі Gradle;

– швидкий і багатофункціональний емулятор;

– уніфіковане середовище, де можна розробляти застосунки для всіх пристроїв Android;

– можливість застосувати зміни до push-коду та зміни ресурсів до запущеної програми без перезапуску програми;

– шаблони коду та інтеграція GitHub, які допоможуть створювати загальні функції програми та імпортувати зразок коду;

– інструменти та фреймворки тестування;

– інструменти Lint для визначення продуктивності, зручності використання, сумісності версій та інших проблем;

– підтримка C++ і NDK.

3 ПРОЄКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ

3.1 Моделювання в IDEF0

Методологія IDEF0 показує побудову ієрархічної системи діаграм – одиничних описів фрагментів системи [13]. Спочатку йде опис системи в цілому та її взаємодії з навколишнім світом (контекстна діаграма), а потім функціональна декомпозиція. Система поділена на підсистеми, і кожна підсистема описана окремо (деталізований вигляд). Потім розділіть кожну підсистему на менші підсистеми, доки не досягнете бажаного рівня деталізації.

Тому процес проектування мобільного додатку необхідно розпочинати з розробки контекстної діаграми IDEF0 [13]. До контекстної діаграми входять:

- Вхідні дані – це інформація чи матеріали, спожиті або змінені під час роботи.
- Результат – інформація або матеріал, створений роботою.
- Менеджмент – Процедури, правила, політика або стандарти для управління роботою.
- Механізми – ресурси (люди, обладнання, пристрої тощо) для виконання роботи. Після аналізу основних елементів контекстної діаграми було створено наступний список даних.
- Введення: потрібен root.
- Результат: Найкоротший шлях.
- Управління: інформація про укриття, технології пошуку найближчого шляху.
- Механізм: користувачі, програмне забезпечення, технічна підтримка.

На основі цих даних була розроблена контекстна діаграма у програмному продукті AllFusion Process Modeler [14], що представлена на рис. 3.1.

Рисунок 3.1 – Контекстна діаграма

Проаналізувавши отриману діаграму, можна отримати лише взаємодію процесів у загальному вигляді, виключивши зайві деталі. Для більш точного опису логіки та порядку роботи модель слід декомпозувати.

Діаграма декомпозиції представлена на рисунку 3.2.

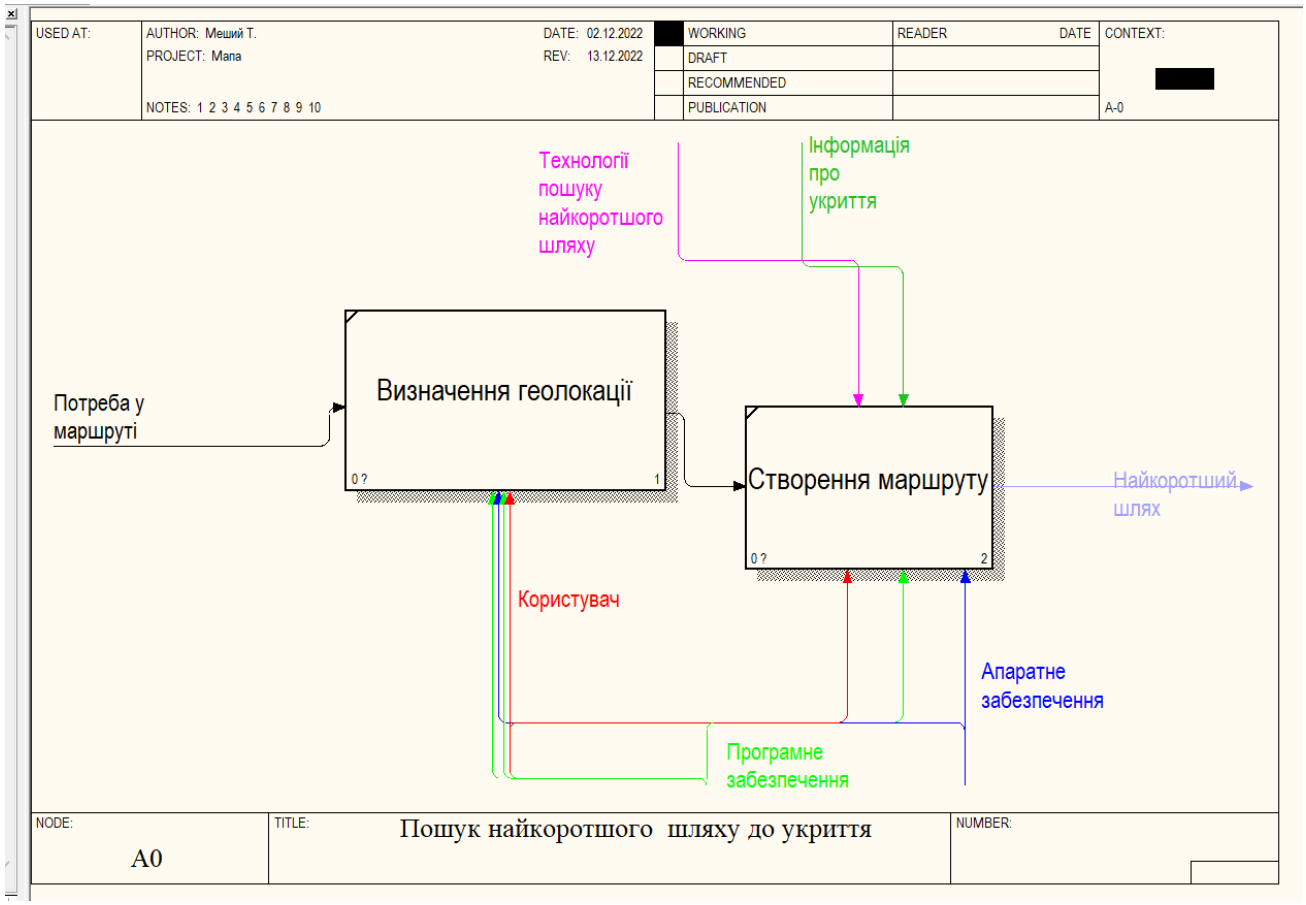

Рисунок 3.2 – Декомпозиція діаграми A0

3.2 Моделювання варіантів використання

Наступним кроком після створення контекстної діаграми є створення діаграми варіантів використання. Ця фігура є графічним зображенням усіх акторів, артефактів, способів використання та зв’язків між ними [15].

Для розробки системи були визначені такі учасники:

– Користувач.

Після визначення акторів в системі, формується перелік варіантів використання, а саме:

– Сховища;

– Маршрут;

– Зворотній зв'язок ;

На основі сформованих даних про варіанти використання та акторів, було розроблено Use Case діаграму, яка представлена рисунку 3.3.

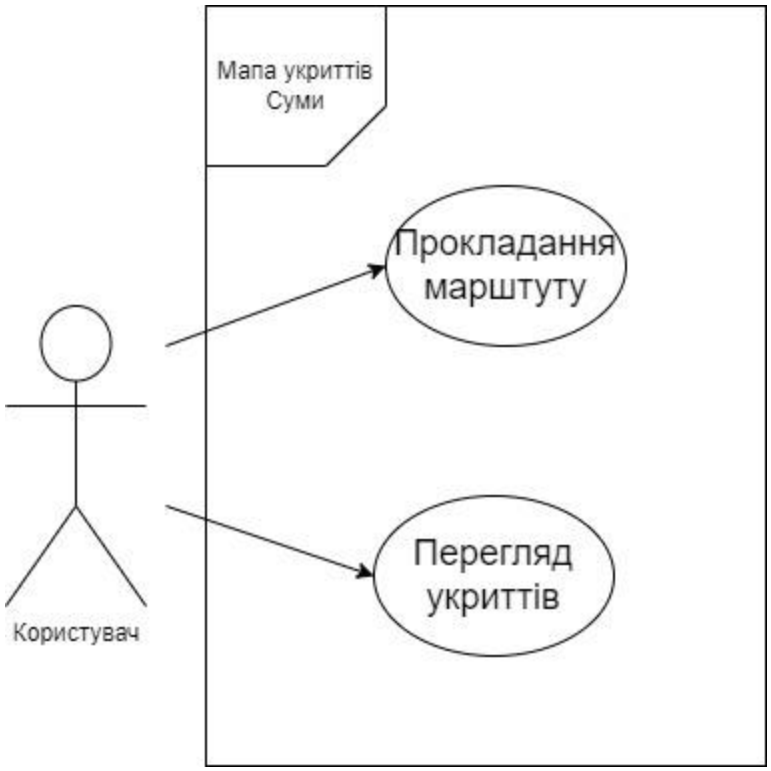

Рисунок 3.3 – Use Case Diagram

Розроблені діаграми дозволять більш ясно представити процес створення.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ДОДАТКА

Додаток має просту структуру з інтуїтивним управлінням. Складається додаток з декількох модулів:

- модуль дозволу на використання геолокації;
- модулю пошуку геолокації ;
- модулю відображення мапи ;
- модулю відображення міток ;
- модуль натискання по карті;
- модуль складання маршруту.

Іконка додатку виглядає як звичайна іконка створено додатку створеного в Android Studio. Головний інтерфейс виглядає просто. Він має три кнопки:

- геолокації,
- знаходження шляху;
- очищення маршруту.

Інтерфейс додатка продемонстровано на рисунку 4.1.

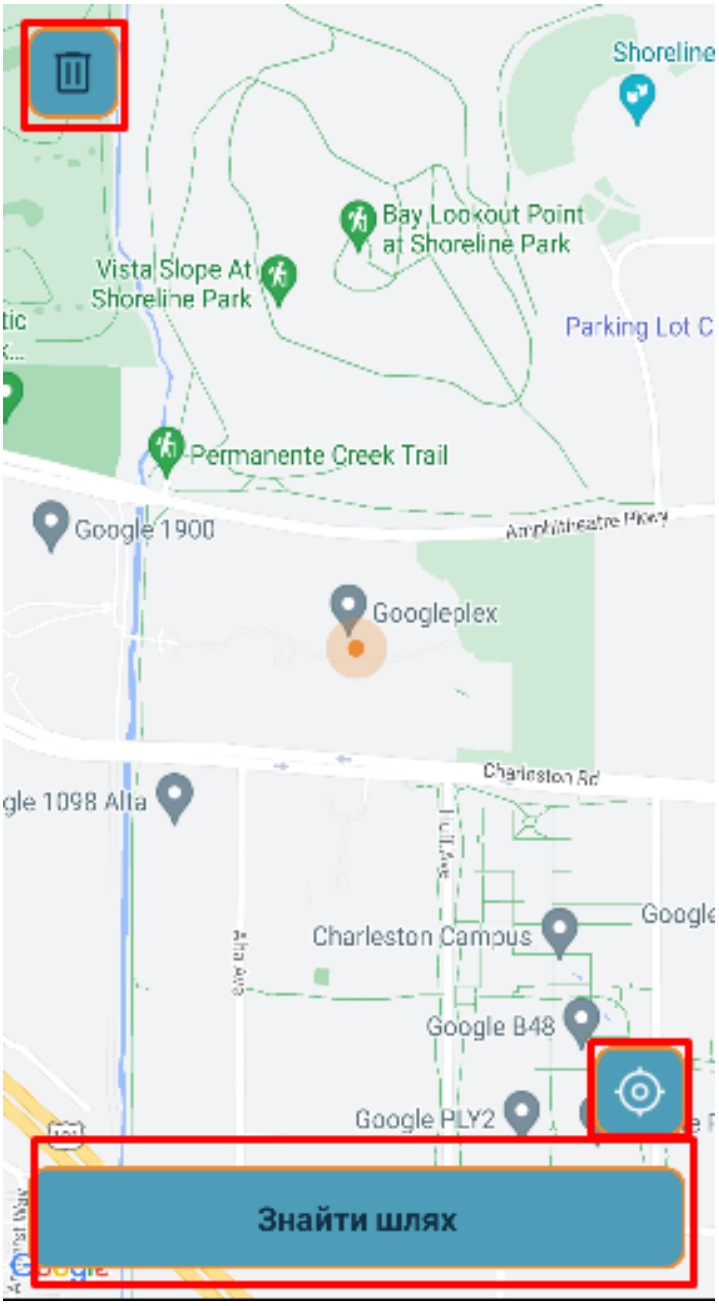

Рисунок 4.1 – Інтерфейс додатка

Далі розроблено вигляд міток користувача та сховищ. Помаранчевим позначено мітку користувача, а синім позначено мітку сховища, як продемонстровано на рисунку 4.2.

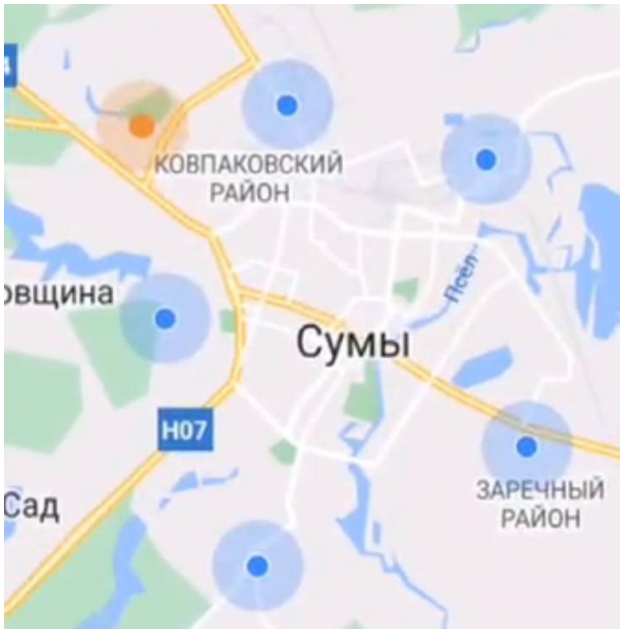

Рисунок 4.2 - Демонстрація міток

Далі розглянемо створення проєкта. По-перше, створюємо в Android Studio новий проект, як представлено на рисунку 4.3.

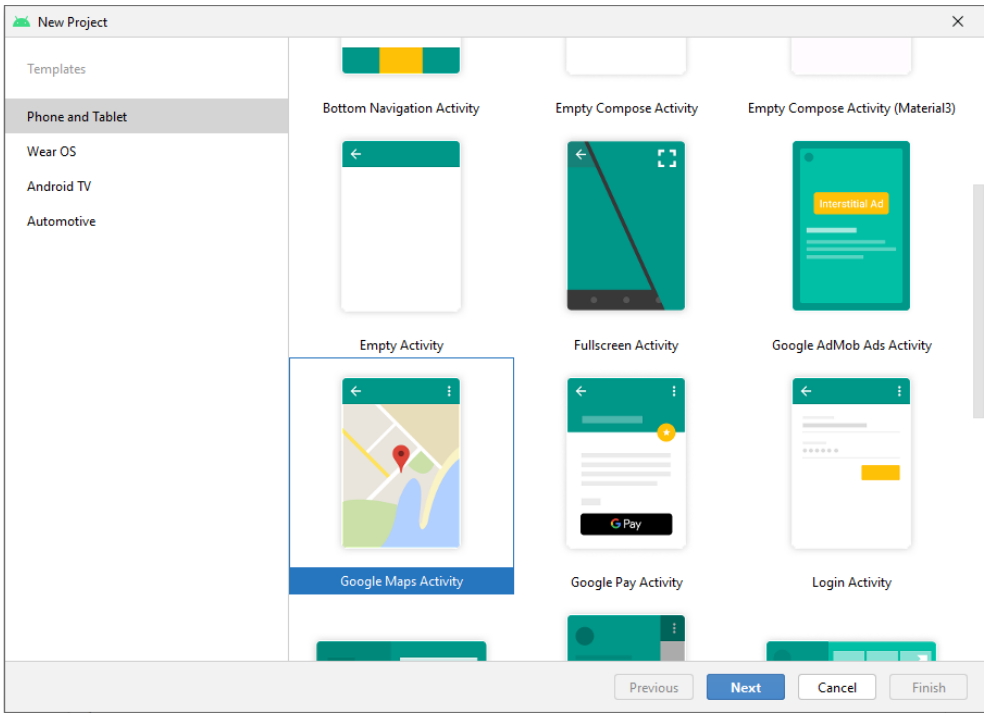

Рисунок 4.3-Створення проекту

Gradle-плагіни є потужним інструментом для доповнення логіки складання необхідними завданнями. На жаль, Gradle API має недостатньо документації, але наявна дозволяє скористатися потрібними функціями для додатку.

Додаємо необхідні залежності у файл build.gradle :

- 'androidx.appcompat:appcompat:1.1.0' – дозволяє отримати доступ до нових API на старих версіях API платформи [24];

- 'androidx.constraintlayout:constraintlayout:1.1.3' – розміщує та розмірує віджети за допомогою відносного позиціонування [25];

- 'com.google.android.gms:play-services-maps:16.1.0' – потрібен для додавання карти в додатку [26];

- 'com.google.android.material:material:1.1.0' – переносить кодову базу з бібліотеки підтримки дизайну в Material Components для Android [27];

- 'io.reactivex.rxjava2:rxjava:2.2.4' – використовує бібліотеку для створення асинхронних програм і програм на основі подій за допомогою спостережуваних послідовностей.

Потім отримуємо ключ Google API, та додаємо наступні дозволи і теги метаданих у тег програми у файлі Androidmenifest:

- android.permission.INTERNET – щоб виконувати мережеві операції у програмі [28];

- android.permission.ACCESS_COARSE_LOCATION — дозволяє API повертати приблизне розташування пристрою. Дозвіл надає оцінку розташування пристрою від служб позиціонування, як описано в документації про приблизну точність позиціонування [29];

- android.permission.ACCESS_FINE_LOCATION — дозволяє API максимально точно визначати місцезнаходження від доступних провайдерів позиціонування, включно з глобальною системою позиціонування (GPS), а також дані WiFi та мобільних стільникових мереж [29];

- com.google.android.gms.version – залежність сервісів Google Play для SDK Maps для Android [30];

- android.intent.action.MAIN,android.intent.category.LAUNCHER – забезпечує вхід верхнього рівня в програму NotePad: стандартна дія MAIN є основною точкою входу (не потребує жодної іншої інформації в Intent), а категорія LAUNCHER говорить, що ця точка входу має бути вказана в панелі запуску програм [31].

Android Studio дозволяє працювати з візуальним інтерфейсом як у режимі коду, так і у графічному режимі. Так, за замовчуванням файл відкритий у графічному режимі, і можна побачити, як приблизно буде виглядати екран програми. Можна ще й додати з панелі інструментів елементи управління, наприклад, кнопки чи текстові поля.

Але також ми можемо працювати з файлом у режимі коду, оскільки activity_main.xml – це звичайний текстовий файл із розміткою xml. Для перемикання в режим коду натиснемо на кнопку Code над графічним представленням (рис. 4.4).

Рисунок 4.4 – Перемикання в режим коду

Далі використаємо метод *checkSelfPermission* інтерфейсу Notification, який запитує дозвіл від користувача для отримання геолокації та відображення сповіщень, як наведено на рисунку 4.5.

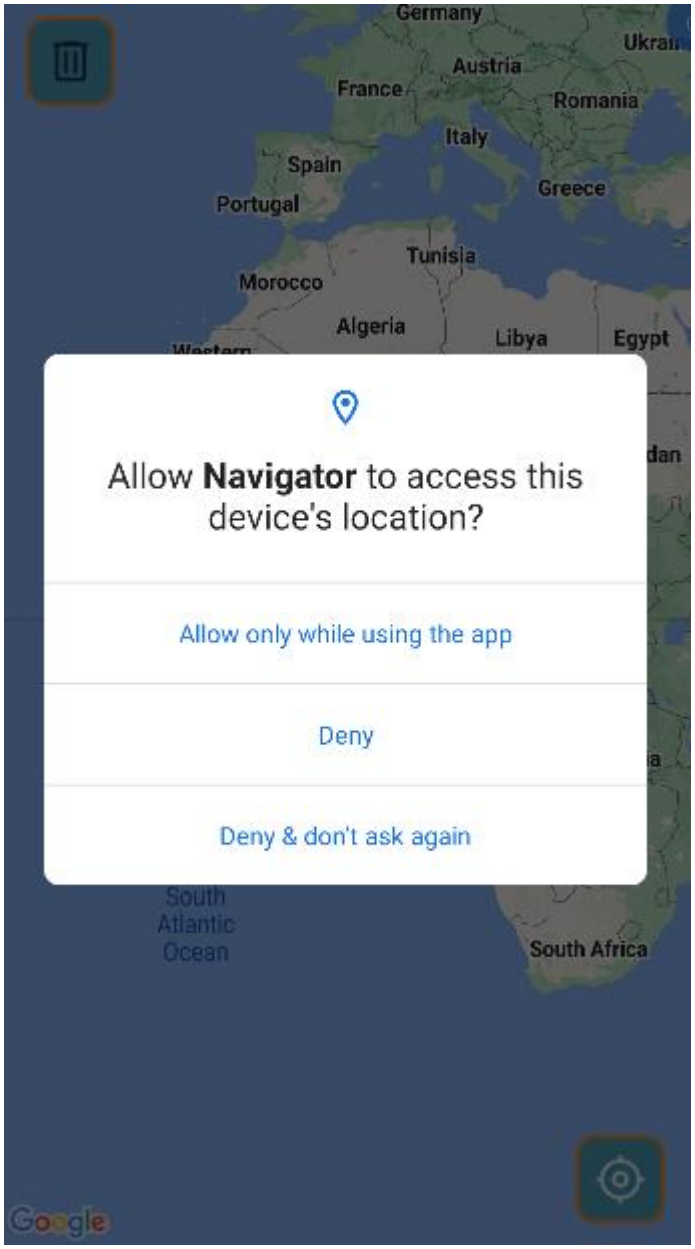

Рисунок 4.5 – Запит дозволу на геолокацію

Для отримання місцезнаходження користувача виконуємо метод getMyLocation. Для виконання методу потрібно отримати API ключ. Також можна сказати, що метод викликається, коли карта готова до використання (рис. 4.6).

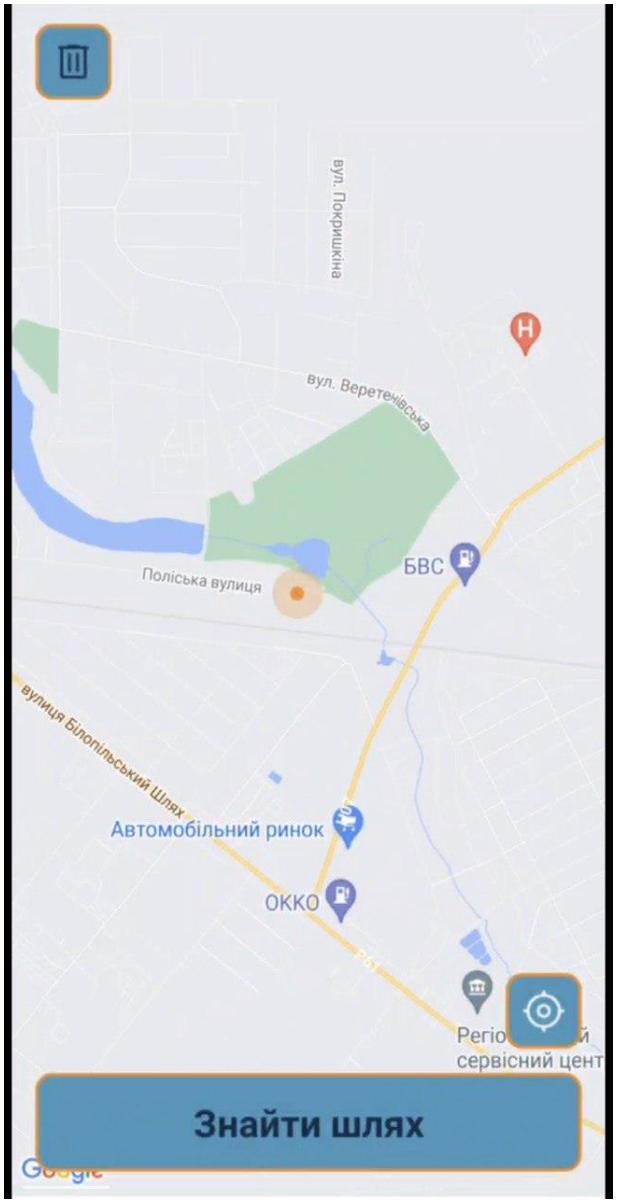

Рисунок 4.6 – Геолокація користувача

Для отримання місця призначення виконується метод публічного статичного інтерфейса GoogleMap.OnMapClickListener. Він використовується для встановлення маркера на клацнуте місце та збереження цього розташування в ArrayList.Інтерфейс зворотного виклику, коли користувач торкається карти.

Щоб відобразити мітку місцезнаходження користувача та сховища використовується метод addMarker. Метод додає маркер на вказані координати (рис. 4.6).

Для отримання найкоротшого шляху ми вимірюємо відстань до сховищ, заносимо їх до матриці, а потім порівнюємо та шукаємо найближчу точку. Якщо пошук маршруту успішний, то додаємо маршрут до карти за допомогою полілінії від маркера позиції користувача (початкова точка маршруту) до маркера сховища (кінцева позиція маршруту), тобто будуємо до неї маршрут (рис. 4.7).

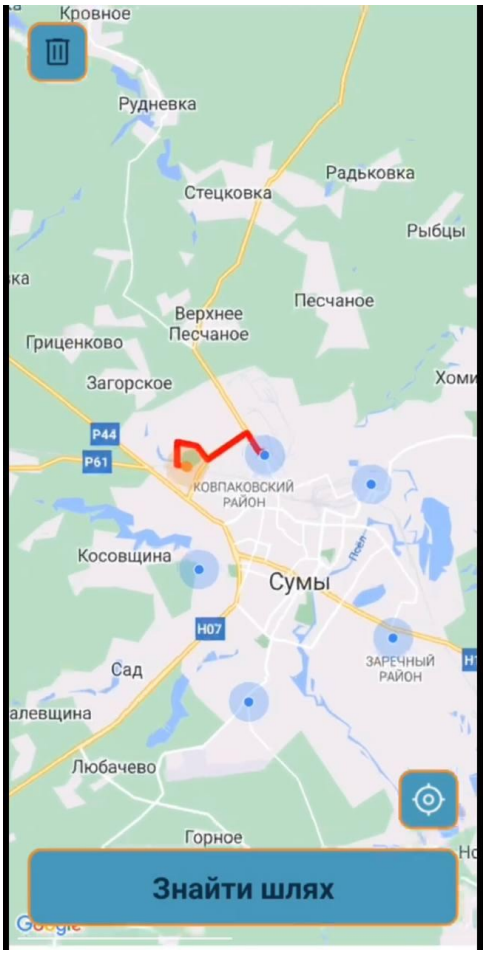

Рисунок 4.7 - Отримання місця призначення

Таким чином ми можемо побачити що маршрут будується до найближчого сховища.

ВИСНОВКИ

У рамках магістерської роботи були досліджені та проаналізовані важливі аспекти розробки додатків. Крім того, було проведено аналіз предметної галузі з використанням літературних джерел для визначення актуальності даної роботи. Пізніше я вирішив розробити мобільний додаток для пошуку найкоротшого шляху до притулку, оскільки він дуже відповідав темі, яку я обрав сьогодні. Крім того, проведено аналіз наявних програмних продуктів. Його функціональність безпосередньо пов'язана з додатком, що створюється. Потім описуються засоби реалізації та обґрунтовується вибір. Виконано структурно-функціональне моделювання процесу розробки моделі. Діаграми варіантів використання були створені з використанням ресурсів програмного забезпечення AllFusion, діаграм контексту та діаграм декомпозиції IDEF0. Створено діаграми WBS та OBS і на їх основі створено матрицю обов’язків людей. Також була створена діаграма Ганта, яка показує терміни та порядок виконання всієї роботи. Крім того, тепер можливо визначити потенційні ризики, допомогти знайти шляхи їх уникнення та скласти план дій щодо усунення факторів їх прояву з розрахованим ступенем впливу на реалізацію проекту. Розроблено інтерфейс програми. Створено модуль визначення геолокації та пошуку найкоротшого шляху. Для реалізації програми використовувалися стандартні бібліотеки Android Studio. Розробили мобільний додаток, який допоможе користувачам додатка міста Суми знайти дорогу до найближчого сховища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. DIGITAL 2016: ГЛОБАЛЬНЕ ВИКОРИСТАННЯ МОБІЛЬНИХ ПРИСТРОЇВ, ЛЮТИЙ 2016 Р [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://datareportal.com/reports/digital-2016-global-mobile-users-february-2016?rq=mobile.

2. АЛГОРИТМ ПОШУКУ НАЙКОРОТШОГО ШЛЯХУ [Електронний ресурс] – Режим доступу до ресурсу: https://jak.koshachek.com/articles/algoritm-poshuku-najkorotshogo-shljahu.html.

3. Алгоритм Дейкстри [Електронний ресурс] – Режим доступу до ресурсу: http://choippo.cn.sch.in.ua/Files/downloadcenter/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%20%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D0%B8.%20%D0%A2%D0%B5%D0%BE%D1%80%D1%96%D1%8F.pdf.

4. Development of geolocation applications [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://wezom.com.ua/blog/razrabotka-geolokatsionnyh-prilozhenij.

5. Де ховатися від ворожих атак: карта бомбосховищ столиці [Електронний ресурс]. – Режим доступу до ресурсу: https://kyivmaps.com/ua/blog/de-hovatisa-vid-vorozih-atak-karta-bomboshovis-stolici.

6. Де найближче укриття та як до нього дістатися: житель Прилук створив мобільну карту [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://suspilne.media/257479-de-najblizce-ukritta-ta-ak-do-nogo-distatisa-zitel-priluk-stvoriv-mobilnu-kartu/.

7. Як зробити геолокацію на телефоні. [Електронний ресурс] – Режим доступу до ресурсу: https://www.zahidknyga.com.ua/robota/jak-zrobiti-geolokaciju-na-telefoni.html.

8. Як керувати доступом до геоданих на пристроях Android. [Електронний ресурс] – Режим доступу до ресурсу: https://support.google.com/accounts/answer/3467281?hl=uk.

9. ЯК СТВОРИТИ ПРОГРАМУ НА ОСНОВІ МІСЦЕЗНАХОДЖЕННЯ ДЛЯ ANDROID ТА IOS [Електронний ресурс] – Режим доступу до ресурсу: https://theappsolutions.com/blog/development/develop-app-with-geolocation/.

10. Як отримати ключ API для Google Maps [Електронний ресурс] – Режим доступу до ресурсу: https://www.bmb.com.ua/2020/12/yak-otrimati-klyuch-api-dlya-google-maps.html.

11. Android Studio [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/studio.

12. Intellij Idea [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/idea/download/#section=windows.

13. Методологія IDEF0 – URL: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema6/tema6_2 (дата звернення: 28.04.2021).

14. Allfusion Process Modeler [Електронний ресурс] – Режим доступу до ресурсу: https://ca-allfusion-process-modeler.software.informer.com/7.2/.

15. Уніфікована мова моделювання UML [Електронний ресурс] – Режим доступу до ресурсу: http://www.znannya.org/?view=uml.

16. Захисні споруди м. Суми 11.03.2016 - Google My Maps. [Електронний ресурс] – Режим доступу до ресурсу: https://www.google.com/maps/d/u/0/viewer?mid=1U6JH6sTiCW37oU3nITeUTx5C5-k&ll=50.90281827493573%2C34.82437405000002&z=13

17. USE CASE-діаграма. Приклади використання [Електронний ресурс] – Режим доступу до ресурсу: http://hi-news.pp.ua/kompyuteri/8924-use-case-dagrama-prikladi-vikoristannya.html.

18. SMART [Електронний ресурс] – Режим доступу до ресурсу: https://cikavoznaty.com.ua/2021/05/09/cili-za-tehnikou-smart/.

19. WBS [Електронний ресурс] – Режим доступу до ресурсу: http://um.co.ua/7/7-2/7-29918.html.

20. Організаційна структура проекту (OBS) [Електронний ресурс] – Режим доступу до ресурсу: https://studopedia.com.ua/1_243503_organizatsiyna-struktura-proektu-oBS.html.

21. Розробка мобільних додатків від А до Я: повний гайд.: [Електронний ресурс]. – Режим доступу до ресурсу: https://dan-it.com.ua/uk/blog/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/

22. Укриття [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://dovidka.info/ukryttya/.

23. Проектування інформаційних системи на основі уніфікованої мови моделювання – URL: https://sites.google.com/site/analizvimogdopz/ lekciie/uml .

24. Appcompat [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/jetpack/androidx/releases/appcompat.

25. Constraintlayout [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/jetpack/androidx/releases/constraintlayout.

26. Maps SDK for Android overview [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/maps/documentation/android-sdk/overview?hl=en.

27. Getting started with Material Components for Android [Електронний ресурс] – Режим доступу до ресурсу: https://m2.material.io/develop/android/docs/getting-started.

28. Connect to the network [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/training/basics/network-ops/connecting.

29. Дані про місцезнаходження [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/maps/documentation/android-sdk/location.

30. Налаштування проекту Android Studio [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/maps/documentation/android-

sdk/config#specify_the_google_play_services_version_number.

31. Intent [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/reference/android/content/Intent.

ДОДАТОК А

А.1 Ідентифікація мети ІТ-проекту

Метою проекту є розробка мобільного додатку для пошуку шляху до найближчого укриття для жителів міста Суми.

Деталізація мети методом SMART [18]:

S – Метою проекту є розробка мобільного додатку для пошуку шляху до найближчого укриття для жителів міста Суми.

M – Система яка зможе стати відкритою платформою, в котрій за можливості кожний зможе використати для своїх потреб.

A – Головною ціллю є створення відкритої для всіх системи для рекомендації музики.

R – Проект має потенціал для монетизації завдяки пожертвам людей.

T – на розробку потрібно – 15 тижнів, при роботі по 6 годин на день.

А.2 Планування змісту структури робіт інформаційної системи

WBS (рис. А.1) у проектному менеджменті та системотехніці є орієнтованою на доконане виконання проєкту декомпозицією проекту на менші частки. Структура декомпозиції робіт є ключовою часткою робіт по проекту, яка організовує командну роботу по проекту у керовані частини. PMBOK визначає структуру декомпозиції робіт як «ієрархічну декомпозицію робіт, що має бути виконаною командою проекту та орієнтована на успішне завершення проекту» [19].

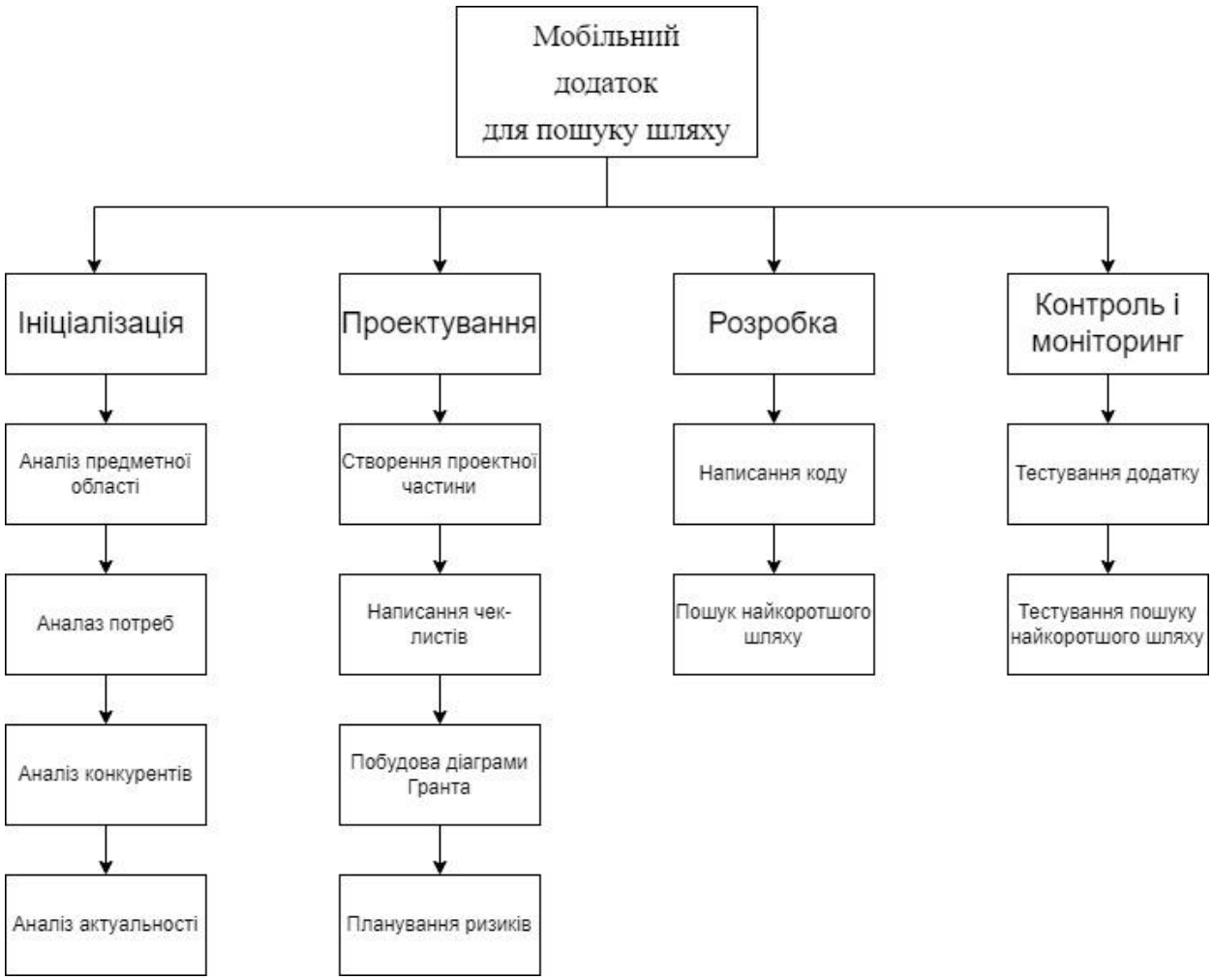

Рисунок А.1 – WBS діаграма

Після створення WBS-структури проекту, потрібно створити організаційну структуру OBS – графічне зображення учасників проекту та осіб, залучених до реалізації проекту [20] (рис.А.2).

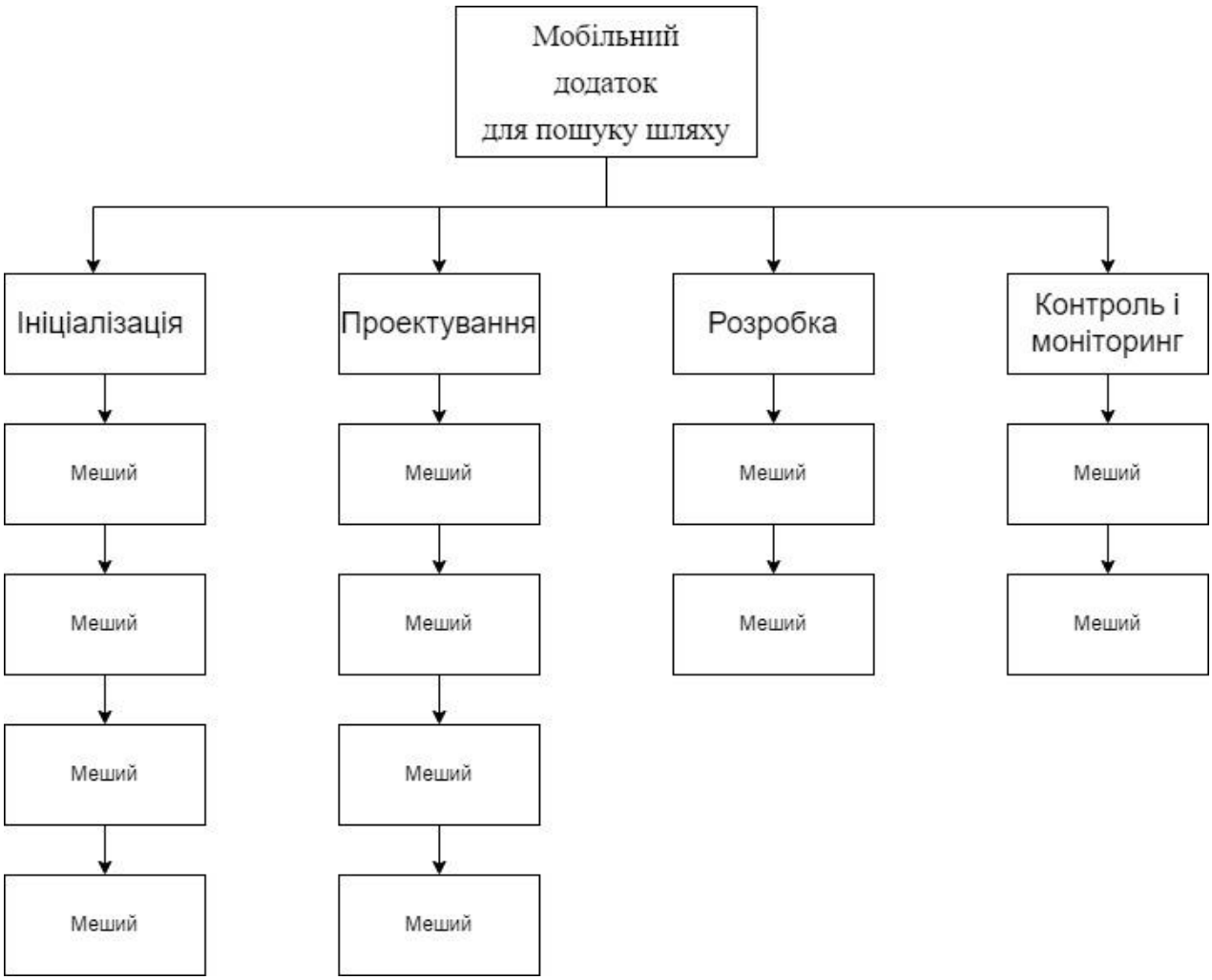

Рисунок А.2 – OBS діаграма

А.3 Побудова календарного графіку виконання інформаційної системи

Діаграма Ганта – лінійна діаграма, яка показує задачі проекту, що представляються часовими відрізками, характеризуються датами початку та закінчення робіт. Діаграма дозволяє відслідковувати відсоток робіт виконаних по кожному завданню. Цей графік представлено за допомогою програмного засобу MS Project.

Календарний план виконання робіт для діаграми Ганта представлений на рисунку А.3. Список робіт для побудови діаграми Ганта представлений на

рисунку А.4.

Здадача	Дата	Кількість днів	
Аналіз предметної області	01.10.2022	1	02.10.2022
Аналіз потреб	03.10.2022	1	04.10.2022
Аналіз конкурентів	05.10.2022	1	06.10.2022
Аналіз актуальності	07.10.2022	1	08.10.2022
Створення проектної частини	09.10.2022	3	12.10.2022
Написання чек-листів	13.10.2022	1	14.10.2022
Побудова діаграми Ганта	15.10.2022	4	19.10.2022
Планування ризиків	20.10.2022	1	21.10.2022
Написання коду	22.10.2022	16	07.11.2022
Пошук найкоротшого шляху	08.11.2022	1	09.11.2022
Тестування додатку	10.11.2022	15	25.11.2022
Тестування найкоротшого шляху	26.11.2022	8	04.12.2022

Рисунок А.3 –Календарний план для діаграми Ганта

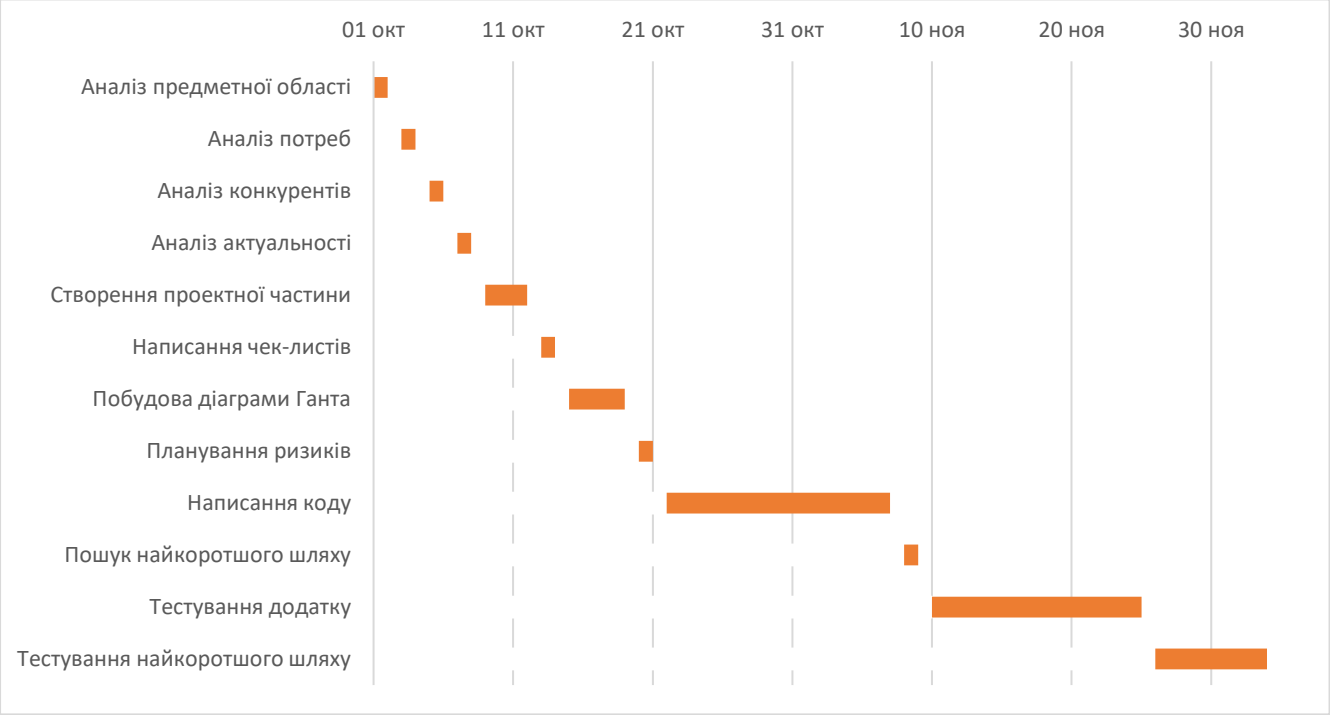

Рисунок А.4 –Діаграма Ганта

А.4 Планування ризиків проекту

Ризики – це негативні події, які можуть статися та вплинути на проект. Наприклад, держава випустить новий закон чи розробник тимчасово не зможе працювати над проектом.

Якщо негативна подія станеться у будь-якому разі, це не ризик, а завдання. Наприклад, менеджер знає, що новому QA-інженеру потрібно вдвічі більше часу на тестування продукту. Його завдання –зважити на факт і закласти достатньо часу на цей етап.

Під час аналізу для визначення значень можливостей появи ризику, була застосована методика експертної оцінки. Виходячи з цих оцінок ми можемо знайти ранг нашого ризику.

R = P * L, де

- R – ранг ризику;
- P – ймовірність виникнення;
- L – ступінь впливу.

Шкала оцінки ризику може відповідати емпіричній шкалі оцінки ризику:

- 5 балів - критичний ризик (0,81 - 1);
- 4 бали - максимальний ризик (0,61 - 0,8);
- 3 бали - високий ризик (0,41 - 0,6);
- 2 бали - нормальний ризик (0,31 - 0,4);
- 1 бал - малий ризик (0 - 0,3).

Ризики проєкту наведені у таблиці А.1.

Таблиця А.1 – Визначені ризики проекту

N	Назва ризику	Опис	P	L	R	План реагування
1	Збільшення кількості функціоналу	при спробі додати нові функції ми можемо стикнутися з тим, що маємо більше працювати над проектом .	0,7	0,8	0,56	У разі настання такого ризику бюджет компанії може бути збільшений для найму нових робітників для пришвидшення роботи над проектом
2	Вимкнення світла	Через війну можливо таке, що може зникнути світло	1	1	1	На це неможливо реагувати
3						

ДОДАТОК Б

Коди модулів проєкту

MainActivity

```
package com.simple.navigator;

import static
com.google.android.gms.location.LocationServices.getFusedLocationProviderClient;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.DialogFragment;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;

import android.Manifest;
import android.animation.AnimatorSet;
import android.animation.ValueAnimator;
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.location.LocationManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Looper;
import android.view.View;
import android.view.animation.LinearInterpolator;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.location.LocationCallback;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationResult;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.location.LocationSettingsRequest;
import com.google.android.gms.location.SettingsClient;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.CameraPosition;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.PolylineOptions;
```

```
import com.simple.navigator.api.APIService;
import com.simple.navigator.model.TargetPoint;
import com.simple.navigator.model.response.MapResponseDistance;
import com.simple.navigator.other.DialogLoading;
import com.simple.navigator.other.DirectionsJSONParser;

import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Objects;
import java.util.concurrent.TimeUnit;

import io.reactivex.Observable;
import io.reactivex.android.schedulers.AndroidSchedulers;
import io.reactivex.functions.Consumer;
import io.reactivex.schedulers.Schedulers;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class MainActivity extends AppCompatActivity implements OnMapReadyCallback {

    // карта
    private GoogleMap mMap;

    // маркер поточного місцезнаходження користувача
    private Marker markerCurLocation;

    // поточні координати
    LatLng curLocation = null;

    // анімація зміни поточного місцезнаходження
    private AnimatorSet animatorSet = new AnimatorSet();

    // кнопки меню
    TextView findPath;
    ImageView ivShowCurPos, ivClearRoute;

    // массив відстаней між поточними координатами та цільвими точками
    ArrayList<MapResponseDistance> distances = new ArrayList<>();

    // індекс цільвої точки для якої будемо робити запит на отримання відстані до
неї
    int curDistanceIndex = 0;

    // діалог завантаження інформації з сервера, парсингу маршруту
    DialogLoading dialogLoading = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        findPath = findViewById(R.id.tv_find_path);
        findPath.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View view) {
                distances.clear();
                curDistanceIndex = 0;

                // перевіряємо наявність дозволу. у випадку відсутності запитуємо
його
                if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
                    if
(ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this,
Manifest.permission.ACCESS_FINE_LOCATION)) {
                        ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 1);
                    } else {
                        ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 1);
                    }
                } else {
                    // якщо дозвіл є, то шукаємо відстані
                    getAllDistances();
                    startLocationUpdates();
                    subscribeToTimer();
                }

            }
        });


        ivClearRoute = findViewById(R.id.iv_clear_route);
        ivClearRoute.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // очищаємо карту від усіх елементів, та повторно відображаємо
цільві точки
                // потрібно для очищення маршруту
                if (mMap != null) {
                    mMap.clear();
                    addTargetPointsToMap();
                    if(curLocation!=null){
                        addMarker(curLocation);
                    }
                }
            }
        });

        ivShowCurPos = findViewById(R.id.iv_show_cur_pos);
        ivShowCurPos.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // анімуємо карту до поточного місцезнаходження
                if (curLocation != null) {
                    updateCurPosMarker(curLocation);
                }

                if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
                    if
(ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this,
Manifest.permission.ACCESS_FINE_LOCATION)) {
```

```
                        ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 1);
                    } else {
                        ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 1);
                    }
                }
            }
        });

        // ініціалізуємо карту у фоновому режимі
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map);
        Objects.requireNonNull(mapFragment).getMapAsync(this);


    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        // після ініціалізації карти додаємо цільві точки
        mMap = googleMap;
        addTargetPointsToMap();

        // залажно від дозволу відстежуємо поточне місцезнаходження
        if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
            startLocationUpdates();
            subscribeToTimer();
        }
        // робимо видимою кнопку очистки карти
        ivClearRoute.setVisibility(View.VISIBLE);
    }

    @SuppressLint("MissingPermission")
    protected void startLocationUpdates() {

        // Створюємо запит місцезнаходження, щоб почати отримувати оновлення
        LocationRequest mLocationRequest = new LocationRequest();
        mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        mLocationRequest.setSmallestDisplacement(10f);
        mLocationRequest.setInterval(0);
        mLocationRequest.setFastestInterval(0);

        // Створюємо об’єкт LocationSettingsRequest за допомогою запиту про
місцезнаходження
        LocationSettingsRequest.Builder builder = new
LocationSettingsRequest.Builder();
        builder.addLocationRequest(mLocationRequest);
        LocationSettingsRequest locationSettingsRequest = builder.build();

        SettingsClient settingsClient = LocationServices.getSettingsClient(this);
        settingsClient.checkLocationSettings(locationSettingsRequest);


getFusedLocationProviderClient(this).requestLocationUpdates(mLocationRequest, new
LocationCallback() {
                    @Override
                    public void onLocationResult(LocationResult locationResult) {
```

```
                        // зберігаємо координати корустувача, та робими видимими
кнопки в нижній частині екрану
                        // інакше може виникнути ситуація, коли користувач буде
шукати маршрут, а програмі ще невідомі поточні координати
                        curLocation = new
LatLng(locationResult.getLastLocation().getLatitude(),
locationResult.getLastLocation().getLongitude());
                        findPath.setVisibility(View.VISIBLE);
                    }
                },
                Looper.myLooper());
    }

    // анімуємо карту до поточних координат
    private void subscribeToTimer() {
        Observable.interval(3, TimeUnit.SECONDS)
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(new Consumer<Long>() {
                    @Override
                    public void accept(Long aLong) throws Exception {
                        sendNextPoints();
                    }
                });
    }


    private void sendNextPoints() {
        if (!animatorSet.isRunning() && curLocation != null) {
            updateCurPosMarker(curLocation);
        }
    }

    // анімуємо карту до поточних координат
    private void updateCurPosMarker(LatLng newlatlng) {

        if (markerCurLocation != null) {
            markerCurLocation.setAnchor(0.5f, 0.5f);
            animatorSet = new AnimatorSet();
            animatorSet.play(moveMarker(newlatlng,
markerCurLocation.getPosition()));
            animatorSet.start();
        } else {
            addMarker(newlatlng);
        }
        mMap.animateCamera(CameraUpdateFactory.newCameraPosition
                (new CameraPosition.Builder().target(newlatlng)
                        .zoom(16f).build()));
    }

    // додаємо маркер поточного місцезнаходження
    private void addMarker(LatLng position) {
        MarkerOptions markerOptions = new
MarkerOptions().position(position).flat(true).icon(BitmapDescriptorFactory.fromReso
urce(R.drawable.ic_user_location));
        markerCurLocation = mMap.addMarker(markerOptions);
    }

    // додаємо маркери цільвих точок
```

```
    private void addTargetPointsToMap() {
        ArrayList<TargetPoint> points = Utils.getTargetPoints();
        for (TargetPoint p : points) {
            MarkerOptions markerOptions = new MarkerOptions().position(new
LatLng(p.getLat(), p.getLon()))
                    .flat(true)
                    .anchor(0.5f, 0.5f)
                    .title(getResources().getString(p.getName()))

.icon(BitmapDescriptorFactory.fromResource(R.drawable.target_point_map_icon));

            mMap.addMarker(markerOptions);
        }
    }

    // анімація переміщення маркера
    public synchronized ValueAnimator moveMarker(final LatLng finalPosition, final
LatLng startPosition) {
        ValueAnimator valueAnimator = ValueAnimator.ofFloat(0, 1);
        valueAnimator.setInterpolator(new LinearInterpolator());
        valueAnimator.setDuration(3000);
        valueAnimator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener()
{
            @Override
            public void onAnimationUpdate(ValueAnimator valueAnimator) {
                float t =
Float.parseFloat(valueAnimator.getAnimatedValue().toString());

                LatLng currentPosition = new LatLng(
                        startPosition.latitude * (1 - t) + (finalPosition.latitude)
* t,
                        startPosition.longitude * (1 - t) +
(finalPosition.longitude) * t);
                markerCurLocation.setPosition(currentPosition);
            }
        });
        return valueAnimator;
    }

    // отримуємо відстані до всіх цільвих точок
    private void getAllDistances() {
        ArrayList<TargetPoint> targetPoints = Utils.getTargetPoints();
        if (curDistanceIndex < targetPoints.size() && curLocation != null) {

            showDialogLoading();
            if (mMap != null) {
                mMap.clear();
                addTargetPointsToMap();
                addMarker(curLocation);
            }

            try {
                String start = curLocation.latitude + "," + curLocation.longitude;
                String apiKey = APIService.API_KEY;

                String end = targetPoints.get(curDistanceIndex).getLat() + "," +
targetPoints.get(curDistanceIndex).getLon();
                App.apiService.getDistance(start, end, apiKey).enqueue(new
Callback<MapResponseDistance>() {
```

```
                    @Override
                    public void onResponse(Call<MapResponseDistance> call,
Response<MapResponseDistance> response) {

                        try {
                            // якщо вдалось отримати відстань, то збільшуємо індекс
та переходимо до наступної цільвої точки
                            MapResponseDistance resp = response.body();
                            if
(resp.getRows().get(0).getElements().get(0).getDistance() != null) {
                                distances.add(resp);
                                curDistanceIndex++;
                                getAllDistances();
                            } else {
                                Toast.makeText(MainActivity.this,
getString(R.string.error_retrieving_distance_to_,

getResources().getString(targetPoints.get(curDistanceIndex).getName())),
Toast.LENGTH_LONG).show();
                                dismissDialogLoading();
                            }

                        } catch (Exception e) {
                            Toast.makeText(MainActivity.this,
getString(R.string.error_retrieving_distance_to_,

getResources().getString(targetPoints.get(curDistanceIndex).getName())),
Toast.LENGTH_LONG).show();
                            dismissDialogLoading();
                        }
                    }

                    @Override
                    public void onFailure(Call<MapResponseDistance> call, Throwable
t) {
                        Toast.makeText(MainActivity.this,
getString(R.string.error_retrieving_distance_to_,

getResources().getString(targetPoints.get(curDistanceIndex).getName())),
Toast.LENGTH_LONG).show();
                        dismissDialogLoading();
                    }
                });
            } catch (Exception e) {
                Toast.makeText(MainActivity.this,
getString(R.string.error_retrieving_distance_to_,

getResources().getString(targetPoints.get(curDistanceIndex).getName())),
Toast.LENGTH_LONG).show();
                dismissDialogLoading();
            }
        } else {
            // після отриманні відсаней до всіх точок переходимо до пошуку маршруту
            getRoute();
        }
    }

    private void getRoute() {
        String start = curLocation.latitude + "," + curLocation.longitude;
```

```
        String apiKey = APIService.API_KEY;
        long minDistance = Long.MAX_VALUE;
        String end = "";
        // шукаємо найближчу цільову точку та її координати
        for (int i = 0; i < distances.size(); i++) {
            MapResponseDistance d = distances.get(i);
            try {
                if
(d.getRows().get(0).getElements().get(0).getDistance().getValue() < minDistance) {
                    minDistance =
d.getRows().get(0).getElements().get(0).getDistance().getValue();
                    TargetPoint p = Utils.getTargetPoints().get(i);
                    end = p.getLat() + "," + p.getLon();
                }
            } catch (Exception e) {
            }
        }

        // здіснюємо запит на отримання маршруту до найближчої цільвої точки
        App.apiService.getPath(start, end, apiKey).enqueue(new Callback<String>() {
            @Override
            public void onResponse(Call<String> call, Response<String> response) {
                try {
                    // декодуємо відповідь від сервера
                    String s = response.body();
                    ParserTask parserTask = new ParserTask();
                    parserTask.execute(s);
                } catch (Exception e) {
                    Toast.makeText(MainActivity.this,
R.string.error_finding_distance, Toast.LENGTH_LONG).show();
                    dismissDialogLoading();
                }
            }

            @Override
            public void onFailure(Call<String> call, Throwable t) {
                Toast.makeText(MainActivity.this, R.string.error_finding_distance,
Toast.LENGTH_LONG).show();
                dismissDialogLoading();
            }
        });
    }

    // відображаєм діалог завантаження
    void showDialogLoading() {
        if (dialogLoading == null || !dialogLoading.isVisible()) {
            dialogLoading = new DialogLoading();
            dialogLoading.setStyle(DialogFragment.STYLE_NO_TITLE,
R.style.DialogFullScreen);
            dialogLoading.setCancelable(false);
            dialogLoading.show(getSupportFragmentManager(), "dialog");
        }
    }

    // приховуємо діалог завантаження
    void dismissDialogLoading() {
        if (dialogLoading != null) {
            dialogLoading.dismiss();
        }
```

```
        FragmentManager fm = getSupportFragmentManager();
        Fragment dialog = fm.findFragmentByTag("dialog");
        if (dialog != null) {
            fm.beginTransaction().remove(dialog).commit();
        }
    }


    private class ParserTask extends AsyncTask<String, Integer,
List<List<HashMap<String, String>>>> {

        // декодуємо маршрут у фоновому режимі
        @Override
        protected List<List<HashMap<String, String>>> doInBackground(String...
jsonData) {

            JSONObject jObject;
            List<List<HashMap<String, String>>> routes = null;

            try {
                jObject = new JSONObject(jsonData[0]);
                DirectionsJSONParser parser = new DirectionsJSONParser();

                routes = parser.parse(jObject);
            } catch (Exception e) {
                // у випадку винекнення проблем із декодуванням показуємо
відповідне повідомлення та приховуємо діалог завантаження
                Toast.makeText(MainActivity.this, R.string.error_finding_distance,
Toast.LENGTH_LONG).show();
                dismissDialogLoading();
            }
            return routes;
        }

        @Override
        protected void onPostExecute(List<List<HashMap<String, String>>> result) {
            ArrayList points = null;
            PolylineOptions lineOptions = null;

            for (int i = 0; i < result.size(); i++) {
                points = new ArrayList();
                lineOptions = new PolylineOptions();

                List<HashMap<String, String>> path = result.get(i);

                for (int j = 0; j < path.size(); j++) {
                    HashMap<String, String> point = path.get(j);

                    double lat = Double.parseDouble(point.get("lat"));
                    double lng = Double.parseDouble(point.get("lng"));
                    LatLng position = new LatLng(lat, lng);

                    points.add(position);
                }

                lineOptions.addAll(points);
                // налаштовуємо вигляд маршруту
                lineOptions.width(12);
                lineOptions.color(Color.RED);
```

```
                lineOptions.geodesic(true);
            }

            // додаємо маршрут на карту
            mMap.addPolyline(lineOptions);
            // приховуємо діалог завантаження
            dismissDialogLoading();
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        switch (requestCode) {
            case 1: {
                if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                    // якщо користувач дав дозвіл, то починаємо відстежувати
поточне місцезнаходження
                    if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
                        startLocationUpdates();
                        subscribeToTimer();
                    }
                } else {
                    // дозвіл не отримали. показуємо відповідне повідомлення
                    Toast.makeText(this, R.string.permission_denied,
Toast.LENGTH_SHORT).show();
                }
                return;
            }
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        LocationManager manager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
        if (!manager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
            Toast.makeText(this, R.string.enable_gps_for_use_app,
Toast.LENGTH_SHORT).show();
        }
    }
}
```

AndroidManifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.simple.navigator">

    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION"
/>
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:name=".App"
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@drawable/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Navigator"
        tools:targetApi="31"
        android:usesCleartextTraffic="true">
        <uses-library android:name="org.apache.http.legacy"
android:required="false" />

        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyC89LIki6skYEULJv8UnYSRbKB6thTnBUc" />

    </application>

</manifest>
```

Activity_main

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <ImageView
        android:id="@+id/iv_clear_route"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_margin="16dp"
        android:background="@drawable/btn_bg"
        android:padding="10dp"
        android:src="@drawable/ic_clear_route" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="bottom|end"
        android:orientation="vertical">


        <ImageView
            android:id="@+id/iv_show_cur_pos"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_margin="16dp"
            android:layout_marginStart="16dp"
            android:layout_marginTop="16dp"
            android:background="@drawable/btn_bg"
            android:padding="10dp"
            android:src="@drawable/ic_my_location" />

        <TextView
            android:id="@+id/tv_find_path"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="16dp"
            android:layout_marginRight="16dp"
            android:layout_marginBottom="16dp"
            android:background="@drawable/btn_bg"
            android:gravity="center"
            android:paddingVertical="16dp"
            android:text="@string/find_path"
            android:textColor="@color/blue_dark"
            android:textSize="18sp"
            android:textStyle="bold"
            android:visibility="gone" />
    </LinearLayout>
```

```
</RelativeLayout>
```

App

```
package com.simple.navigator;

import android.app.Application;

import com.simple.navigator.api.APIService;

import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;
import retrofit2.converter.scalars.ScalarsConverterFactory;

public class App extends Application {

    public static APIService apiService;


    @Override
    public void onCreate() {
        super.onCreate();

        // ініціалізація ретрофіта для відправлення запитів до сервера
        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl(APIService.BASE_URL)
                .addConverterFactory(ScalarsConverterFactory.create())
                .addConverterFactory(GsonConverterFactory.create())
                .build();
        apiService = retrofit.create(APIService.class);
    }
}
```

Utils

```
package com.simple.navigator;

import com.simple.navigator.model.TargetPoint;

import java.util.ArrayList;

public class Utils {

    // додаємо цільві точки
    public static ArrayList<TargetPoint> getTargetPoints(){
        ArrayList<TargetPoint> points = new ArrayList<>();
        points.add(new TargetPoint(R.string.target_point_1_name,
50.892558061019606, 34.83642459735727));
        points.add(new TargetPoint(R.string.target_point_2_name,
50.875803312174554, 34.77668686229881));
        points.add(new TargetPoint(R.string.target_point_3_name, 50.91055910366983,
34.75611374273554));
        points.add(new TargetPoint(R.string.target_point_4_name, 50.94050452395689,
34.78326267999344));
        points.add(new TargetPoint(R.string.target_point_5_name, 50.93287004588913,
34.827378617801735));
        return points;
    }
}
```

APIServise

```
package com.simple.navigator.api;

import com.simple.navigator.model.response.MapResponseDistance;
import retrofit2.Call;
import retrofit2.http.GET;
import retrofit2.http.Query;

public interface APIService {

    public static final String BASE_URL = "https://maps.googleapis.com/";
    public static final String API_KEY = "AIzaSyC89LIki6skYEULJv8UnYSRbKB6thTnBUc";

    // запит на отримання відстаней між точками
    @GET("maps/api/distancematrix/json?units=metric")
    Call<MapResponseDistance> getDistance(@Query(value="origins") String start,
                                          @Query(value="destinations") String end,
                                          @Query(value="key") String key);


    // запит на отримання маршруту між точками
    @GET("maps/api/directions/json?units=metric")
    Call<String> getPath(@Query(value="origin") String start,
                         @Query(value="destination") String end,
                         @Query(value="key") String key);
}
```

build.gradle

```
plugins {
    id 'com.android.application'
    id 'com.google.android.libraries.mapsplatform.secrets-gradle-plugin'
}

android {
    compileSdk 32

    defaultConfig {
        applicationId "com.simple.navigator"
        minSdk 23
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {

    implementation 'androidx.appcompat:appcompat:1.5.1'
    implementation 'com.google.android.material:material:1.7.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.4'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.0'

    implementation 'com.google.android.gms:play-services-maps:16.0.0'
    implementation 'com.google.android.gms:play-services-location:16.0.0'
    implementation 'io.reactivex.rxjava2:rxjava:2.2.4'
    implementation 'io.reactivex.rxjava2:rxandroid:2.1.0'

    implementation 'com.squareup.retrofit2:retrofit:2.5.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.3.0'
    implementation 'com.squareup.retrofit2:converter-scalars:2.1.0'

   // implementation 'com.google.maps.android:android-maps-utils:0.5'

   // implementation 'com.github.AntonioHReyes:EasyRoutes:1.0.11'
}
```

Target.point

```
package com.simple.navigator.model;

// модель, що описує цільву точку (до якої будуємо маршрут)
public class TargetPoint {

    int name;
    double lat;
    double lon;

    public TargetPoint() {}

    public TargetPoint(int name, double lat, double lon) {
        this.name = name;
        this.lat = lat;
        this.lon = lon;
    }

    public int getName() {
        return name;
    }

    public void setName(int name) {
        this.name = name;
    }

    public double getLat() {
        return lat;
    }

    public void setLat(double lat) {
        this.lat = lat;
    }

    public double getLon() {
        return lon;
    }

    public void setLon(double lon) {
        this.lon = lon;
    }
}
```