Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмного засобу-навігатора з доповненою реальністю»

Виконав: студент IV курсу
групи 2ПІ–18б (д/ф)
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Кагальняк Р.Ю.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф ПЗ Бабюк Н.П.
(прізвище та ініціали)

Рецензент: к.т.н., доц. каф. КН Крилик Л.В.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«____» _____ 2022 р.

ВНТУ– 2022

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 кафедрою ПЗ
д.т.н, професор Романюк О. Н.
25 березня 2022 р.

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

Кагальняку Руслану Юрійовичу

1. Тема роботи: «Розробка програмного засобу-навігатора з доповненою реальністю»

Керівник роботи: Бабюк Наталя. Петрівна. к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від "24" березня 2022 року № 66.

2. Строк подання студентом роботи "10" червня 2022 року

3. Вихідні дані до роботи: середовище розробки – Android Studio, Мови розробки – Kotlin, Java, Операційна система – Windows 7/8/8.1/10.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз та постановка задачі; розробка архітектури та алгоритмів програмного додатка; розробка програмного додатку; тестування додатку, висновки; список використаних джерел, додатки, графічна частина.

5. Перелік графічного матеріалу: демонстрація роботи аналогів, блок-схема функціонування програмного продукту, блок-схема взаємодії модулів програмного продукту, загальний алгоритм роботи додатку та тестування додатку.

6. Консультанти розділів бакалаврської дипломної роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1–4	к.т.н. Бабюк Н. П., доцент кафедри ПЗ	26.03.2022	10.06.2022

7. Дата видачі завдання ____10.06.2022р._____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз проблеми, обґрунтування актуальності розробки системи та постановка задачі	26.03.22-14.04.22	Вик.
2	Розробка архітектури та алгоритмів роботи системи	15.04.22-25.04.22	Вик.
3	Вибір середовища та мови розробки	26.04.22-12.05.22	Вик.
4	Розробка програмного продукту	13.05.22-30.05.22	Вик.
5	Тестування роботи системи	31.05.22-03.06.22	Вик.
6	Оформлення матеріалів до захисту БДР	03.06.22-10.06.22	Вик.

Студент _____ **Кагальняк Р.Ю.**
(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ **Бабюк Н.П.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У бакалаврській роботі було виконано аналіз стану галузі мобільних навігаційних програм, а також сформовано завдання та мету, предмет і об’єкт дослідження. Також було проведено порівняння аналогів, і вирішено, що дана розробка є актуальною. Система буде надавати користувачам навігаційні функції з використанням технології доповненої реальності.

Було розроблено алгоритми, блок-схеми, для найбільш великих операцій. А також розроблено адаптивний інтерфейс підходящий для будь-якого Android смартфону.

Мобільний додаток було написано на мовах програмування Kotlin, Java, а також за допомогою Android SDK і GoogleMaps API. Середовище розробки було обрано Android Studio. Кінцевий програмний продукт відповідає завданням та меті. У підсумку було отримано мобільний додаток, швидкодія та коректність роботи, якого була перевірена.

ANNOTATION

In the bachelor's thesis the analysis in the field of mobile navigation programs was performed, as well as the tasks and purpose, subject and object of research were formed. A comparison of analogues was also made, and it was decided that this development is relevant. The system will provide users with navigation features using augmented reality technology.

Algorithms, block diagrams, have been developed for the largest operations. And also developed an adaptive interface suitable for any Android smartphone.

The mobile application was written in Kotlin, Java, and Android SDK and GoogleMaps API. The development environment was chosen Android Studio. The final software product meets the objectives and goals. As a result, a mobile application was received, the speed and correctness of the work, which was tested.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. В наш час досить стрімко збільшуються площі міст і будуються нові населені пункти, також значно збільшується кількість приватного транспорту. Тож проблема пошуку маршруту завжди залишається актуальною.

З цього виникає потреба у системах навігації , які були б зрозумілі будь-якому користувачу, та були б зручні як для пішоходів, так і для водіїв.

Навігатор з доповненою реальністю (мобільний додаток) – надає можливість користувачу бачити маршрут , прокладений прямо на екрані камери. За рахунок цього користувачу не обов'язково знатись на топографії щоб знайти маршрут до потрібної точки, а лиш достатньо увімкнути камеру та слідувати покажчикам які прокладені на зображенні яке камера транслює на екран смартфона. Таким чином навігація стає зрозуміла навіть дітям.

Чим більше функціональних можливостей містить мобільний навігатор, тим ефективніше його використання та ширша варіативність для різних випадків. Основна функція розроблюваного додатку це можливість нанесення маршруту безпосередньо на зображення яке транслюється через камеру. Також основна вимога будь-якого навігатора це перегляд карти певної області. Пошук по назві дозволяє знайти об'єкт не знаючи його місцезнаходження і в подальшому прокласти маршрут. Також одною з основних функцій навігатора є пошук свого місцезнаходження на карті. Проте це вимагає наявну та ввімкнену систему GPS.

Отже, розробка програмного засобу - навігатора з доповненою реальністю є актуальною, завдяки функціональності, доступності та варіативності.

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

Мета та завдання дослідження. Метою бакалаврської роботи є спрощення процесу навігації, завдяки розробці навігатора з доповненою реальністю.

- Аналіз стану галузі мобільних навігаторів
- Порівняльний аналіз аналогів
- Аналіз методів розв’язання задачі
- Постановка задач для мобільного додатку навігації з доповненою реальністю
- Розробку структури і алгоритмів програмного продукту
- Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу
- Розробка додатку – навігатора з доповненою реальністю
- Тестування додатку

Об’єктом дослідження є процес створення структури та алгоритмів для навігатора з доповненою реальністю

Предметом дослідження є методи та засоби для реалізація навігатора з доповненою реальністю.

Методи дослідження. Протягом дослідження було використано: методи розробки клієнтського інтерфейсу; методи навчання; методи оптимізації клієнтського інтерфейсу та мобільного додатка в цілому; методи алгоритмізації модулів програмної системи.

Новизна отриманих результатів.

Подальшого розвитку отримав метод нанесення навігаційних покажчиків за допомогою технології AR, який відрізняється від досліджуваних зрозумілістю отриманого маршруту і, таким чином, полегшує процес навігації.

Практична цінність отриманих результатів. Розроблено алгоритми та програмні засоби для навігатора з доповненою реальністю.

Особистий внесок здобувача. Наукові результати, які містяться у бакалаврській роботі, були отримані індивідуально автором. У працях, опублікованих у співавторстві, автор має такі результати: архітектура клієнтських інтерфейсів; алгоритми роботи освітнього порталу.

Апробація і публікації. Основні положення бакалаврської дипломної роботи доповідалися та обговорювалися на міжнародних та всеукраїнських конференціях: LI Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2022), Молодь в науці: дослідження, проблеми, перспективи (МН-2022).

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану галузі мобільних навігаторів

Населення нашої планети постійно збільшується і, відповідно, будуються нові населені пункти або ж збільшуються існуючі. Тільки за останній рік показник природного приросту населення світу становив 10,8% і він постійно збільшується. Також постійно збільшується показник урбанізації країн , який залежно від країни становить від 33% до 71%. Разом з цим збільшується кількість приватного транспорту. Тільки за 2021 рік українці придбали 945 тисяч легкових авто. То ж проблема пошуку місцезнаходження та навігації залишається актуальною.

Особливо важливу роль в вирішенні цієї проблеми зіграла поява технології GPS. Global Positioning System або система глобального позиціювання — це сукупність радіоелектронних засобів, що дозволяє визначати положення та швидкість руху об'єкта на поверхні Землі або в атмосфері [1]. Першим мобільним телефоном з підтримкою GPS став Benefon ESC який з’явився в 2000 році, проте широку популярність здобула наступна версія телефону - Benefon Twig , який був випущений в 2006 році. Не являючись смартфоном він мав вбудований набір карт та можливість пошуку свого місцезнаходження.

З появою смартфонів стало набагато легше орієнтуватись на місцевості і знаходити потрібні локації. Зараз кожен користувач може завантажити спеціалізовані додатки чи набори карт і за допомогою Інтернету знайти будь-яку локацію в будь-якій точці планети. Сучасні навігаційні системи володіють величезним набором функцій, таких як: перегляд класичних карт, перегляд супутникових знімків, аналіз ландшафту, погодні карти, пошук населених пунктів та закладів, прокладення маршруту і обрахунок оптимального транспорту для певних випадків, тощо.

Проте не для кожного користувача класичний інтерфейс може бути зручним та зрозумілим. Особливо звичні карти незрозумілі для дітей. Також для більшості навігаційних додатків необхідно мати персональний аккаунт і всі дані користувача зберігаються на серверах корпорацій, що потенційно може призвести до витоку персональних даних користувачі.

Як результат вищесказаного , було вирішено створити мобільний додаток – навігатор з доповненою реальністю, який виправить недоліки існуючих мобільних навігаторів та зробить користування додатком більш зручним та зрозумілим для усіх користувачів. Даний мобільний додаток буде розповсюджуватись безкоштовно .

1.2 Порівняльний аналіз аналогів розроблюваного додатку

Для визначення актуальності розробки мобільного додатку потрібно провести аналіз аналогів. Завдяки критеріям було визначено доцільність розробки даного мобільного додатку.

Оскільки мобільна розробка умовно поділилась на 3 види– розробка додатків для мобільних телефонів на основі операційної системи Android, iOS, та кросплатформенна розробка, для аналізу аналогів було обрано додатки з усіх видів.

Розглянемо декілька найпопулярніших мобільних навігаторів.

Google Maps (рис. 1.1) – один з найпопулярніших мобільних навігаторів, частково за рахунок того що додаток заздалегідь встановлений в кожному Android смартфоні [2]. Мобільний додаток розроблений корпорацією Google. Додаток має функції карт, перегляд ландшафту, аналіз стану міського трафіку, вимірювання дистанції між заданими точками, перегляд супутникових знімків місцевості, перегляд вулиць та AR навігація. Даний додаток має зв’язок з персональним акаунтом, що дозволяє завантажувати дані, збережені на будь-якому іншому девайсі, з’єднаним з цим же акаунтом.

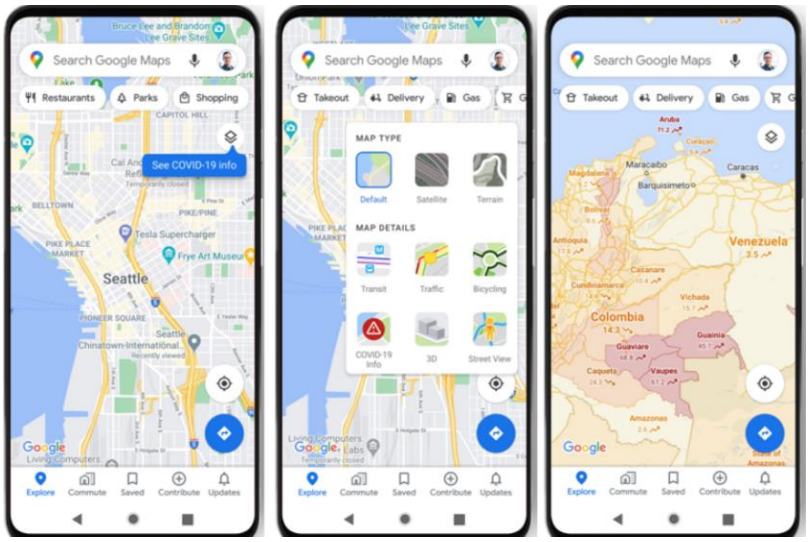

Рисунок 1.1 – Інтерфейс мобільного навігатора Google Maps

Apple Maps (рис. 1.2) – мобільний додаток розроблений компанією Apple для пристроїв на основі iOS для перегляду карт та навігації [3]. Додаток має звичні функції навігатора, такі як перегляд карт, визначення власного місцезнаходження, пошук локації за назвою та за координатами, а також, власне, прокладання маршруту. У одному з останніх оновлень додатку було додано функцію перегляду вулиць в режимі доповненої реальності. У додатку є можливість збереження даних у своєму персональному акаунті, за рахунок чого збережені дані майже неможливо втратити.

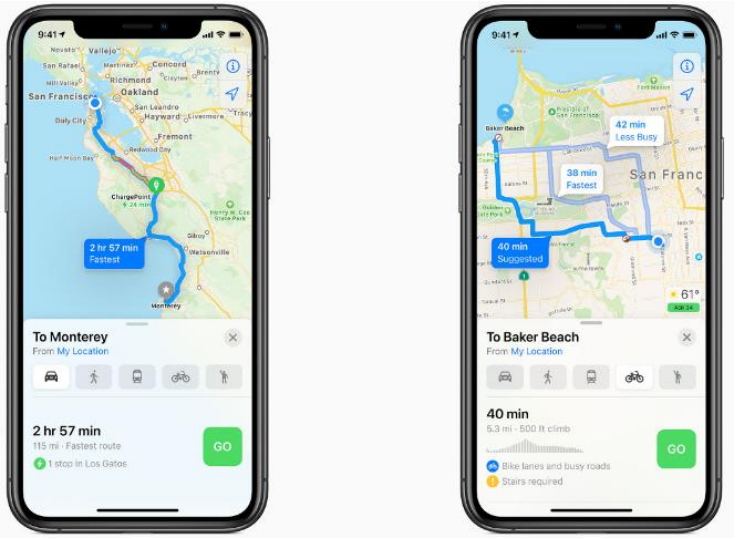

Рисунок 1.2 – Інтерфейс мобільного навігатора Apple Maps

ARCity (рис. 1.3) – мобільний додаток створений компанією Blippar для мобільних телефонів на основі операційної системи iOS [4]. Даний додаток призваний спростити використання навігатора шляхом інтеграції технології доповненої реальності. За рахунок цього користувач може бачити прокладений маршрут та підказки прямо на екрані камери свого телефона.

Рисунок 1.3 – Приклад роботи додатку ARCity

Sygic (рис. 1.4) – навігаційна система, розроблена однойменною словацькою компанією Sygic. Компанія була заснована в 2004 році зі штаб-квартирою в Братиславі, Словаччина. Вона стала першою компанією, яка запропонувала навігацію для iPhone і другою для Android. Його можна використовувати як онлайн, так і офлайн, працює на операційних системах Android, Android Auto, iOS, Windows Phone та Symbian, пропонує карти для більш ніж 200 країн світу та працює більш ніж 30 мовами. Користувачі навігації Sygic GPS можуть завантажувати карти на свої пристрої та використовувати їх, коли їм потрібна навігація, але не мають підключення до Інтернету. Sygic оптимізує розміри завантаження даних, щоб користувачі могли використовувати карти в автономному режимі, використовуючи мінімальний обсяг пам’яті на своїх пристроях.

Рисунок 1.4 – Мобільний додаток Sygic

Результати аналізу і порівняння аналогів показано в табл. 1.1.

Таблиця 1.1 – Переваги та недоліки аналогів

Критерії	Google Maps	Apple Maps	ARCity	Sygic	Розроблюваний додаток
Кросплатформенність	-	-	-	+	+
Незалежність від персонального аккаунта	-	-	-	-	+
Прокладення маршрутів	+	+	+	+	+
Наявність AR технології	+	+	+	-	+

Дана таблиця містить 4 критеріїв, які характеризують дані аналоги, та створюваний у дипломній роботі мобільний додаток [5].

Отже, розробка мобільного навігатора з доповненою реальністю є актуальною на даний час, так як площі міст постійно збільшуються і будуються нові міста і потреба в пошуку маршрутів завжди залишається актуальною. Програмне забезпечення має особливості, які не спостерігаються у аналогічних додатках та які виводять продукт на новий рівень – балансоване поєднання простоти та функціональності. Один з найголовніших та критичних факторів є повна незалежність додатку від операційної системи мобільного пристрою, таким чином додаток можу бути запущений на будь-якому смартфоні.

1.3 Аналіз методів розв’язання поставленої задачі

Аналіз методів для розв’язання задачі є дуже важливим етапом. Від даного етапу залежить наскільки розроблювальний програмний продукт буде якісним, конкурентоспроможним на ринку, а також яку кількість ресурсів потрібно для розробки даного продукту. Можна застосовувати різні технології, бібліотеки для розв’язання задачі, які можуть змінити час розробки і якість. Існує декілька варіантів розв’язання поставленої задачі.

Було вирішено створювати мобільний додаток за допомогою Android SDK, використовуючи мови програмування Java та Kotlin, IDE – Android Studio [6]. Для забезпечення доступу мобільного додатку до карт було обрано комплекс програмних засобів Google Maps API, для реалізації роботи додатку в режимі доповненої реальності було вирішено використовувати набір інструментів Navigation SDK від Mapbox [7]. Додаток буде створений за принципом Single Activity Application, дотримуючись принципів Clean Architecture та SOLID, з використанням рекомендованих інструментів з переліку Jetpack.

Такий підхід дозволить забезпечити найвищу швидкодію та працездатність мобільного додатку, а також пришвидшить та полегшить розробку і підтримку даного додатку.

1.4 Постановка задач для програмного засобу - навігатора з доповненою реальністю

Після аналізу та дослідження по темі розробка програмного засобу - навігатора з доповненою реальністю, було виділено задачі, які забезпечують якість та працедію модулів.

- визначити найбільш ефективний підхід до організації мобільного навігатора;
- розробити алгоритм завантаження , парсингу та відображення мапи та маршрутів;
- розробити графічний інтерфейс користувача для взаємодії із різними модулями мобільному додатку;
- розробити модуль для екрану з доповненою реальністю;
- здійснити тестування програмного модуля.

1.5 Висновки

У першому розділі було розглянуто тему мобільних навігаційних систем.

Після дослідження та проведення аналіз аналогів, які існують на даний момент, було вирішено розробляти мобільний навігатор, оскільки дана тему є дуже актуальна на даний період часу.

Було обрано метод самостійної розробки модулів мобільного додатку, оскільки, саме цей метод має найбільше переваг. Було обрано і визначено задачі, які потрібно виконати для успішної розробки додатку.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Відображення карти у додатку

Кожен навігатор повинен відображати карту. Для організації доступу та взаємодії користувача з картою в даному додатку було використано мережевий протокол http та GoogleMaps API [8]. Робота з картою влаштована наступним чином: екран містить об’єкт в який в подальшому буде завантажено карту. Для цього в проект було імпортовано SDK. Тепер використовуючи попередньо створений та розміщений у layout файлах UI-об’єкт карта завантажується в цей об’єкт і відмальовується на екрані(рис. 2.1).

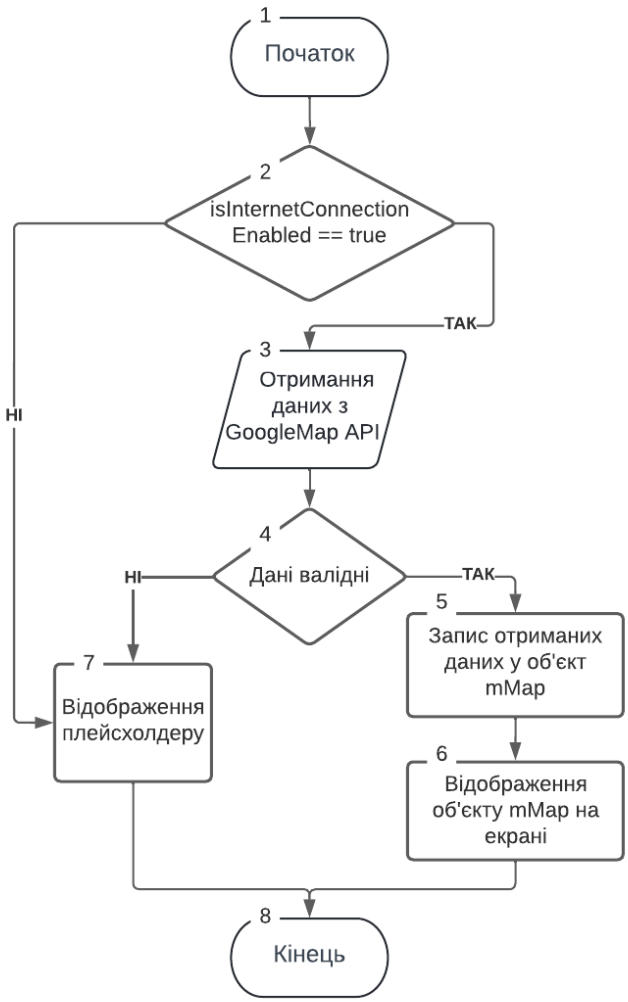

Рисунок 2.1 – Діаграма завантаження карти у додаток

2.2 Розробка алгоритму нанесення навігаційних покажчиків за допомогою технології AR

Основною функцією розроблюваного додатку є інтеграція технології доповненої реальності для перегляду маршрутів в реальному часі за допомогою камери смартфона. Для цього було використано сервіс Mapbox та набір засобів Navigation API. Для цього у додатку було реалізовано окремий екран для більш зручного user experience. Також було розроблено алгоритм для створення і коректного відображення AR покажчиків (рис. 2.2).

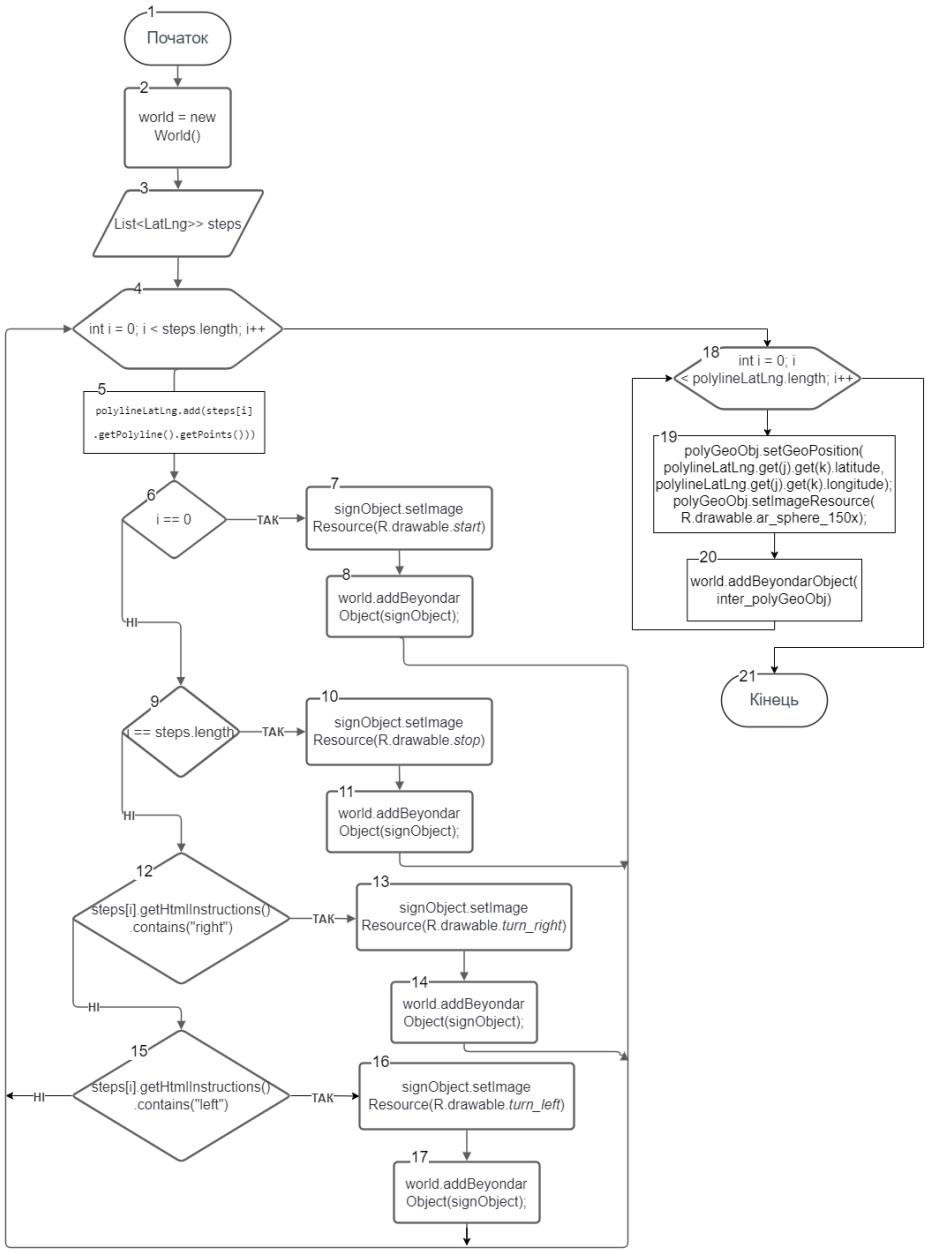

Рисунок 2.2 – Алгоритм для створення і позиціонування AR покажчиків

2.3 Розробка структури інтерфейсу мобільного навігатора

Так як додаток повинен бути адаптивним для смартфона з будь якою діагоналлю розробці інтерфейсу користувача була приділена особлива увага. Інтерфейс додатку побудовано за принципами Material Design, з використанням рекомендованого набору інструментів таких як : Jetpack Navigation, Material, ConstraintLayout. Для верстки дизайну було використано мову розмітки XML. Було виконано дизайн екрану з основною картою (рис. 2.3).

Рисунок 2.3 – Дизайн основного екрану додатку з мапою та панеллю навігації

Всю площу екрану займає карта, з якою користувач може взаємодіяти. У верхній частині екрану розміщена стрічка пошуку, в яку користувач може ввести назву країни, міста, чи будь-якої іншої локації. Після введення назви

локації і натискання кнопки пошуку карта буде переведена для показу обраної локації. В правому нижньому куті розміщена кнопка меню, при натисканні на яку відкриваються додаткові 3 пункти: навігація за допомогою AR, POI пошук, інформація. При натисканні на пункт AR Navigation відкриється екран з налаштуванням навігації з доповненою реальністю. При натисканні на пункт POI Browser користувач буде перенаправлений на екран з пошуком POI локацій [9]. При натисканні на пункт About відкриється екран з інформацією про програму та розробника.

Також було створено допоміжний екран для введення початкової та шуканої точки , з варіантами прокладення маршруту між ними (рис. 2.4).

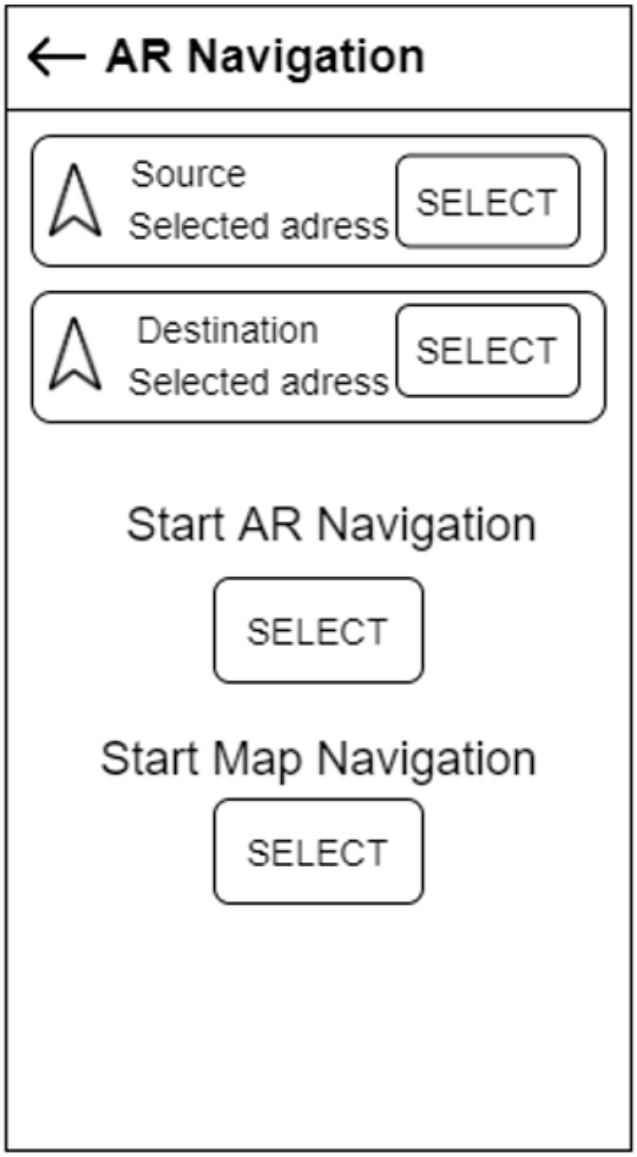

Рисунок 2.4 – Дизайн допоміжного екрану

Зверху на екрані розміщений тулбар з назвою екрану і кнопкою «Назад». При натисканні на кпонку «Назад» буде виконана навігація на попередній екран. Нижче розміщено дві view для вибору початкової і шуканої локації. Під ними розміщені дві кнопки: «Start AR Navigation» і «Start Map Navigation». При натисканні на кнопку «Start AR Navigation» буде відкрито екран з AR навігацією. При натисканні на «Start Map Navigation» буде відкрито екран з картою і прокладено маршрут між обраними точками.

Також було розроблено дизайн екрану для роботи з доповненою реальністю: основний AR екран, банери для POI (Point of interest) елементів, та діалог з детальною інформацією про обраний об’єкт (рис. 2.5).

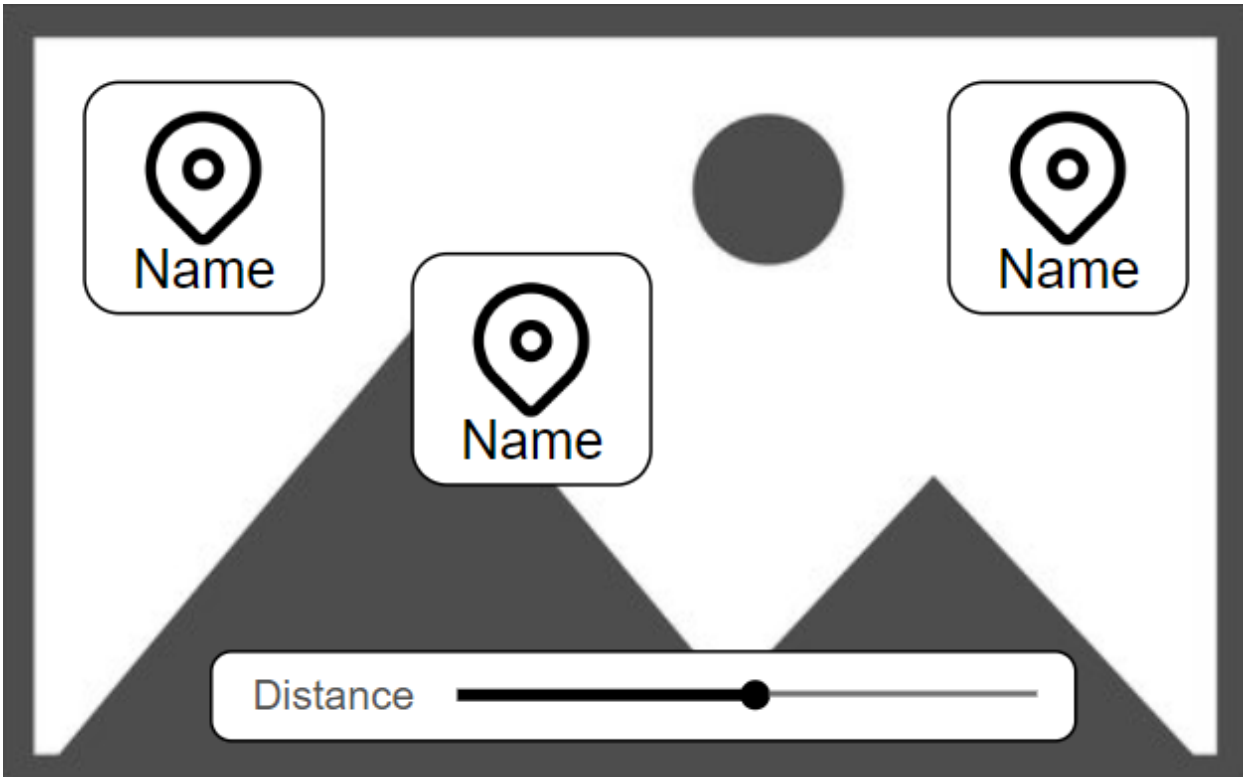

Рисунок 2.5 –Дизайн основного AR екрану з POI елементами

При відкритті даного екрану вмикається камера і основну частину екрану займає зображення з камери. В нижній частині екрану розміщений слайдер з вибором дистанції пошуку локацій. Банери з POI елементами розміщаються відносно їх локації на місцевості. При натисканні на банер відкривається екран з детальною інформації про об’єкт (рис. 2.6).

Рисунок 2.6 – Дизайн екрану з детальною інформацією про обраний об’єкт

Екран містить назву обраного об’єкта, опис, картинку чи фото, і дві кнопки для прокладення маршруту до цієї локації.

Також було розроблено дизайн екрану з AR навігацією (рис. 2.7). Основну частину екрану займає зображення з камери та маршрут який вказує напрямок до обраної локації. Також у правому нижньому куті розміщено покажчики дальності та часу до обраної локації.

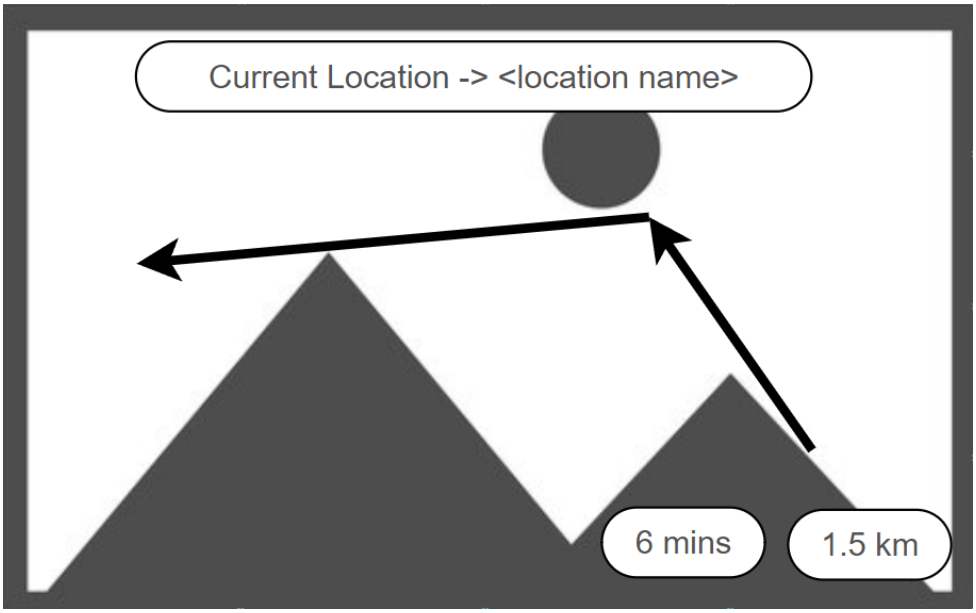

Рисунок 2.7 – Дизайн основного екрану з прокладеним маршрутом

2.4 Розробка алгоритму роботи додатку

Кожне програмне забезпечення містить певні процеси, які можна описати блок-схемою. Алгоритм включає в себе вхідні дані, обробку даних, а також вихідні дані, які користувач, може використовувати.

На діаграмі наведено базовий алгоритм роботи програми (рис. 2.8). Під час запуску додатку спершу відкривається головний екран , який містить мапу та навігаційну панель. Відбувається завантаження даних з API. Далі у користувача є вибір – обрати інший режим програми чи залишись на базовому екрані. При виборі «Main map» або при закритті меню вибору режимів користувач залишиться на головному екрані. При виборі «AR map» додаток переключиться з головної сторінки на сторінку з екраном з доповненою реальністю. При виборі пункту «POI browser» попередньо обраний екран заміниться AR екраном з преселектнутим показом елементів Point of interest (POI). При закритті додатку відбувається очистка введених даних та закриття додатку.

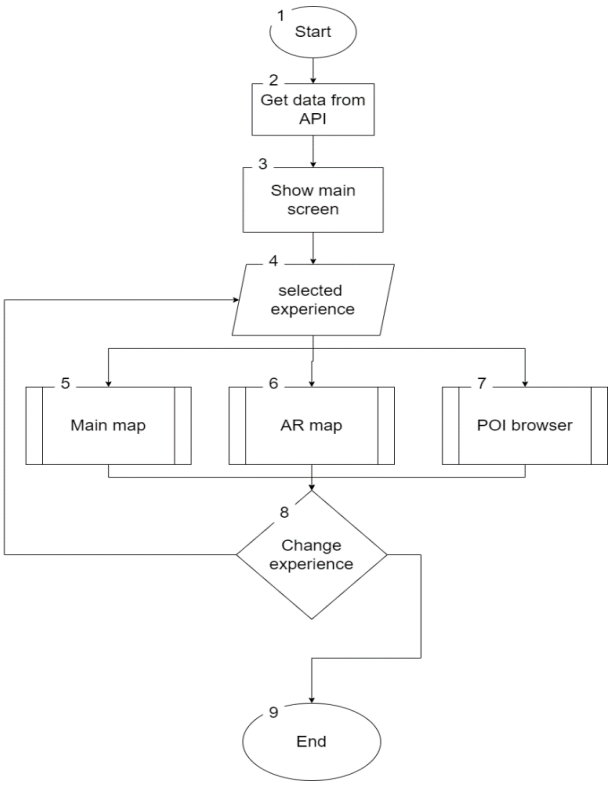

Рисунок 2.8 – Блок-схема базового алгоритма програми

2.5 Висновки

У результаті написання цього розділу, було описано аналіз вхідних та вихідних даних. Було розроблено графічний інтерфейс. А також було розроблено базовий алгоритм роботи додатку, алгоритм відображення карти, та алгоритм відображення доповненої реальності. Розроблено структуру інтерфейсу мобільного навігатора.

3 РОЗРОБКА МОДУЛІВ ПРОГРАМНОГО ПРОДУКТУ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу.

Від використаних технологій та програмних засобів залежить якість програмного продукту, швидкодія та підтримка. Мобільна розробка буває трьох типів: під девайси з операційною системою Android, iOS та кросплатформенна розробка. Мобільний навігатор з доповненою реальністю було вирішено створити для платформи Android, з подальшою можливістю міграції на технологію Kotlin Multiplatform Mobile.

Для розробки мобільного додатку було вирішено використовувати мову програмування Kotlin [10]. Kotlin – статично типізована мова програмування, що працює поверх JVM і розробляється компанією JetBrains. Також компілюється в JavaScript. Фактично являється лаконічнішою та типо-безпечнішою мовою ніж Java. З травня 2019 року є рекомендованою мовою програмування для розробки Android додатків.

Також було вирішено використовувати Android Studio в якості IDE. Android Studio – інтегроване середовище розробки (IDE) для платформи Android, представлене 16 травня 2013 року на конференції Google I/O [11]. Android Studio прийшла на заміну плагіну ADT для середовища Eclipse. Середовище побудоване на основі вихідного коду IntelliJ IDEA Community Edition, що розвивається компанією JetBrains.

Gradle – система автоматичного збирання, яка далі розвиває принципи, закладені в Apache Ant та Apache Maven і використовує предметно-орієнтовану мову (DSL) на основі мови Groovy замість традиційної XML-подібної форми представлення конфігурації проєкту [12]. Являється основною системою збирання проектів для середовища Android Studio.

Буде використано GitHub в якості системи контроля версій, оскільки даний веб-сервіс дозволить зберігати дані, робити зміни, і зберігати на інших

серверах дану інформацію. Навіть при втраті даних на власному комп'ютері, можна буде завантажити дані з репозиторію. Ця технологія є необхідним інструментом кожного розробника, оскільки при розробці ми зберігаємо проміжні етапи програмної розробки.

Дана технологія працює, за змішаним варіантом, в певних умовах зберігає файли, тобто основні версії, а інформація про менші зміни, записується у файл, де вказано, який рядок був змінений, і відповідно на який. Другий варіант, є дуже оптимізованим, і займає мало пам'яті на сервері, чи локально на комп'ютері.

3.2 Розробка мобільного навігатора з доповненою реальністю

Було створено розмітку екранів згідно попередньо розроблених дизайнів. Так як додаток повинен бути адаптивним для смартфона з будь якою діагональю розробці інтерфейсу користувача була приділена особлива увага [13]. Інтерфейс додатку побудовано за принципами Material Design, з використанням рекомендованого набору інструментів таких як : Jetpack Navigation, Material, ConstraintLayout. Для верстки дизайну було використано мову розмітки XML (рис. 3.1). Було виконано розмітку екрану з основною картою (3.2).

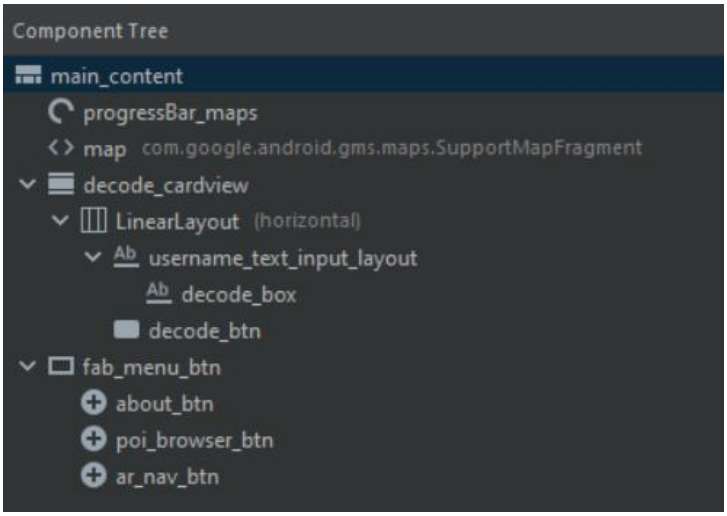

Рисунок 3.1 – Ієрархія UI елементів основного екрану з мапою

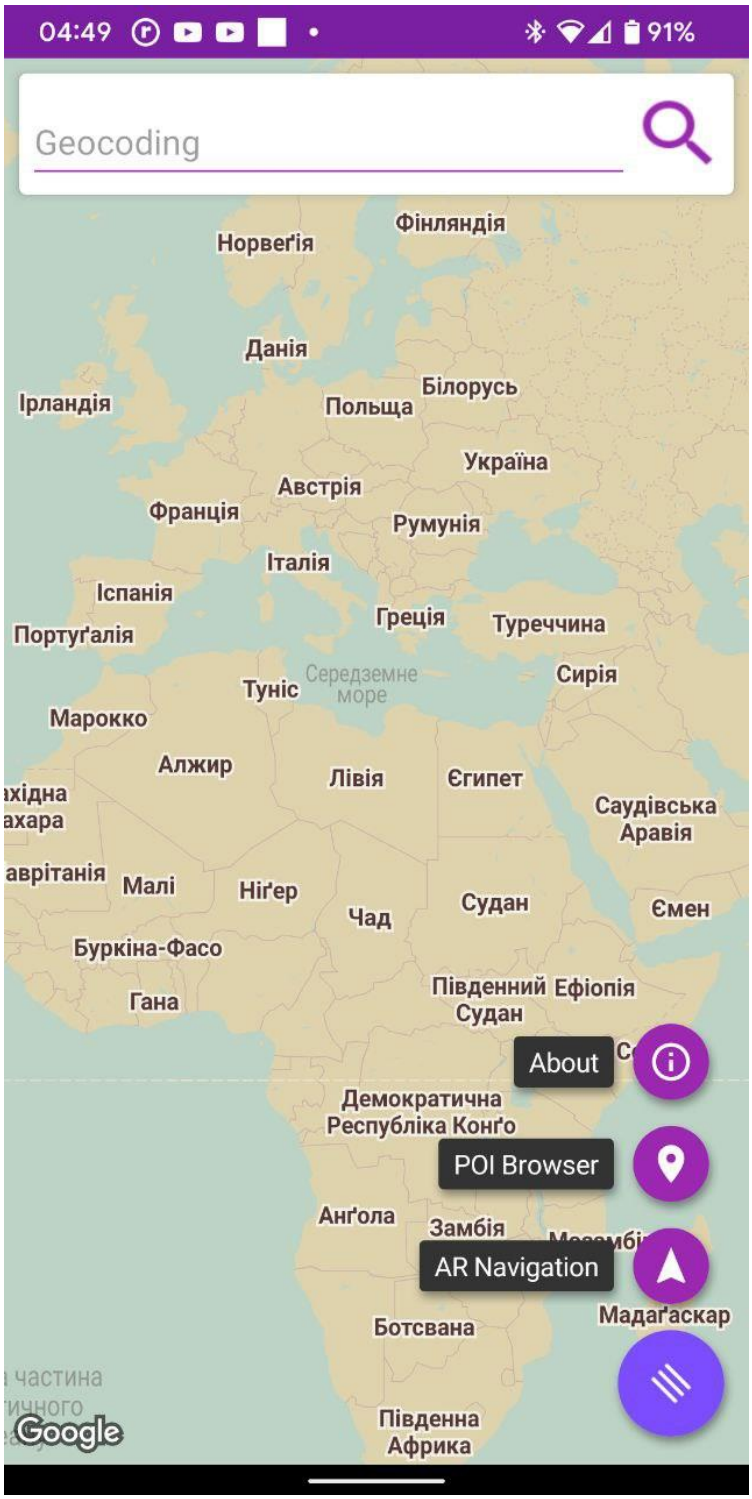

Рисунок 3.2 – Основний екран додатку з мапою та панеллю керування

Для завантаження мапи в додаток використано GoogleMaps API. В програмі реалізований колбек який викликається як тільки мапа буде завантажена (рис. 3.3). Після завантаження мапи об’єкт класу GoogleMap записується у змінну mMap і в подальшому відмальовується на UI шарі.

```java
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    try {
        boolean success = googleMap.setMapStyle(
                MapStyleOptions.loadRawResourceStyle(
                        clientContext: this, R.raw.style_json));
        if (!success) {
            Log.e(TAG,  msg: "Style parsing failed.");
        }
    } catch (Resources.NotFoundException e) {
        Log.e(TAG,  msg: "Can't find style. Error: ", e);
    }

    mMap.setOnMapLongClickListener(this);
}
```

Рисунок 3.3 – Фрагмент коду для завантаження мапи в додаток

Також було створено допоміжний екран для введення початковою та шуканої точки , з варіантами прокладення маршруту між ними (рис. 3.4, рис. 3.5).

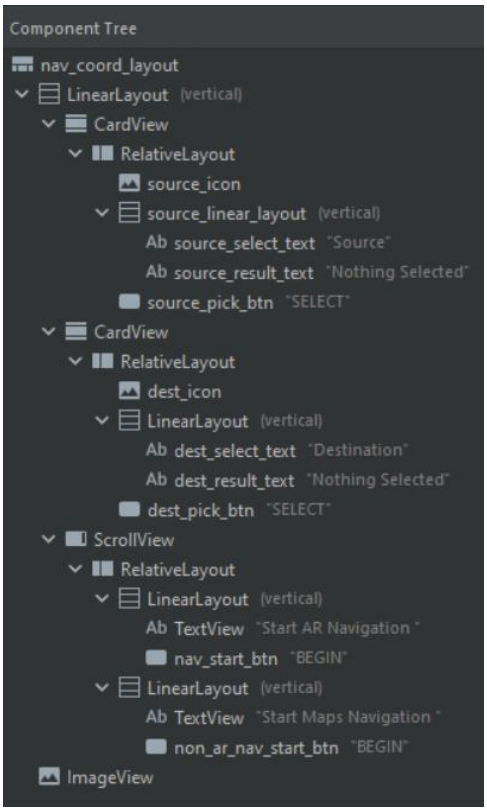

Рисунок 3.4 – Ієрархія UI елементів допоміжного екрану

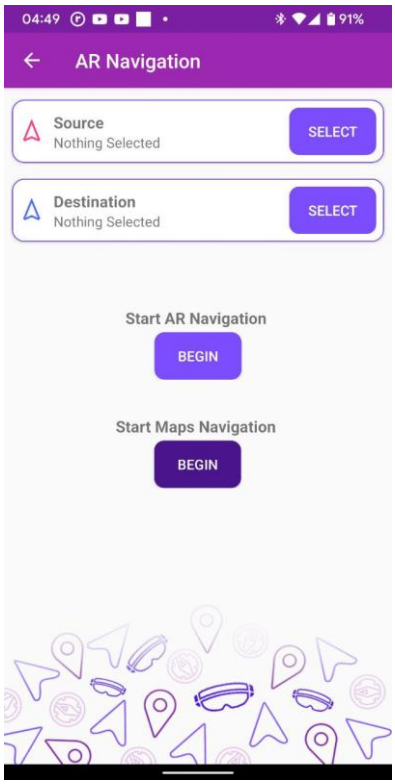

Рисунок 3.5 – Дизайн допоміжного екрану

При натисканні на кнопку «Start Maps Navigation» запуститься екран з навігацією між обраними точками на 2D мапі. Фрагмент коду для запуску навігації між обраними точками наведено на рисунку 3.6.

```
mapNavStartBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent;
        try{
            Uri.Builder builder = new Uri.Builder();
            builder.scheme("http")
                    .authority("maps.google.com")
                    .appendPath("maps")
                    .appendQueryParameter("saddr", srcLatLong.latitude + "," + srcLatLong.longitude)
                    .appendQueryParameter("daddr", destLatLong.latitude + "," + destLatLong.longitude);

            intent = new Intent(android.content.Intent.ACTION_VIEW,
                    Uri.parse( builder.build().toString()));
                            //"http://maps.google.com/maps?saddr=20.344,34.34&daddr=20.5666,45.345"));
            startActivity(intent);
        }catch (Exception e){
            Log.d(TAG,  msg: "onClick: mapNav Exception caught");
            Snackbar mySnackbar = Snackbar.make(findViewById(R.id.nav_coord_layout),
                    text: "Source/Destination Fields are Invalid", Snackbar.LENGTH_SHORT);
            mySnackbar.show();
        }
    }
});
```

Рисунок 3.6 – Фрагмент коду для запуску екрану з навігацією між обраними точками

При натисканні на кнопку «Start AR Navigation» запуститься екран з AR навігацією між обраними точками за допомогою доповненої реальності. Фрагмент коду для запуску навігації між обраними точками наведено на рисунку 3.7.

```
navStartBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent=new Intent( packageContext: NavActivity.this,ArCamActivity.class);

        try {
            intent.putExtra( name: "SRC", sourceResultText.getText());
            intent.putExtra( name: "DEST", destResultText.getText());
            intent.putExtra( name: "SRCLATLNG",  value: srcLatLong.latitude + "," + srcLatLong.longitude);
            intent.putExtra( name: "DESTLATLNG",  value: destLatLong.latitude + "," + destLatLong.longitude);
            startActivity(intent);
        }catch (NullPointerException npe){
            Snackbar mySnackbar = Snackbar.make(findViewById(R.id.nav_coord_layout),
                    text: "Source/Destination Fields are Invalid", Snackbar.LENGTH_SHORT);
            mySnackbar.show();
            Log.d(TAG,  msg: "onClick: The IntentExtras are Empty");
        }
    }
});
```

Рисунок 3.7 - Фрагмент коду для запуску екрану з AR навігацією між обраними точками

Також було створено екран та набір view для роботи з доповненою реальністю: основний AR екран, банери для POI (Point of interest) елементів, та діалог з детальною інформацією про обраний об’єкт [14]. Так як для реалізації доповненої реальності програмі необхідно мати доступ до камери, було реалізовано запит на використання камери (рис. 3.8).

```
if (ContextCompat.checkSelfPermission(context, Manifest.permission.CAMERA) !=
        PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(activity, new String[]{Manifest.permission.CAMERA},  requestCode: 10);
}
```

```
@Override
public void onRequestPermissionsResult(int requestCode,String permissions[], int[] grantResults) {
    switch (requestCode) {
        case 10: {
            if (grantResults.length > 0
                    && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            } else {
            }
            return;
        }
    }
}
```

Рисунок 3.8 – Фрагмент коду для запиту на використання камери

Для інтеграції AR екрану в мобільний додаток було реалізовано кастомний ArFragment. В XML файлі розмітки екрану вводиться назва екрану у тег <fragment> (рис. 3.9).

```
<fragment
    android:id="@+id/ar_cam_fragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:name="com.univ.team12.navar.ar.ArFragmentSupport">
    </fragment>
```

Рисунок 3.9 – Кастомний фрагмент ArFragment

Далі потрібно впевнитись чи додаток має доступ до потрібних сенсорів , таких як компас та акселерометер [15]. Для цього була реалізована функція checkIfSensorsAvailable() (рис.3.10).

```
private void checkIfSensorsAvailable() {
    PackageManager pm = getActivity().getPackageManager();
    boolean compass = pm.hasSystemFeature(PackageManager.FEATURE_SENSOR_COMPASS);
    boolean accelerometer = pm.hasSystemFeature(PackageManager.FEATURE_SENSOR_ACCELEROMETER);
    if (!compass && !accelerometer) {
        throw new IllegalStateException(getClass().getName()
                + " can not run without the compass and the acelerometer sensors.");
    } else if (!compass) {
        throw new IllegalStateException(getClass().getName() + " can not run without the compass sensor.");
    } else if (!accelerometer) {
        throw new IllegalStateException(getClass().getName()
                + " can not run without the acelerometer sensor.");
    }

}
```

Рисунок 3.10 – Метод checkIfSensorsAvailable() для перевірки доступу до сенсорів

Згідно попередніх дизайнів була виконана верстка AR екрану з відображенням POI (Point of interest) елементів. Основну частину екрану займає зображення з камери поверх якого відображаються мітки POI елементів (рис.3.11).

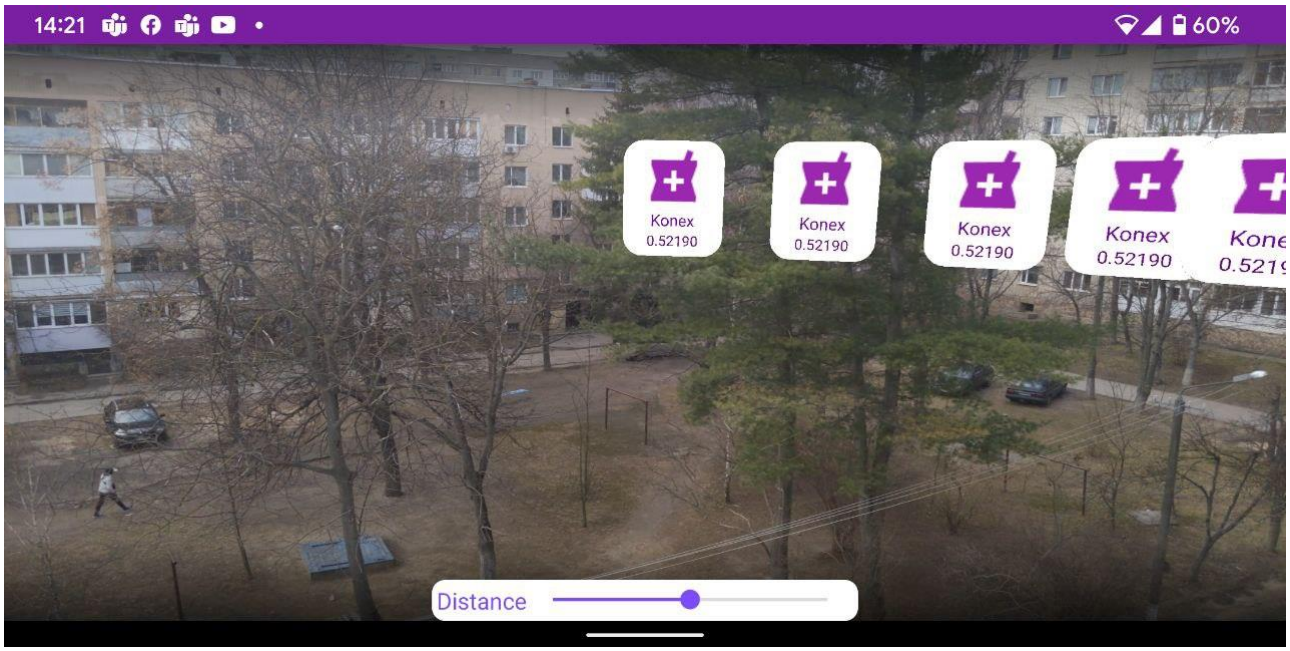

Рисунок 3.11 – Основний AR екран з POI елементами

На рисунку 3.12 зображена ієрархія UI елементів на основному AR екрані.

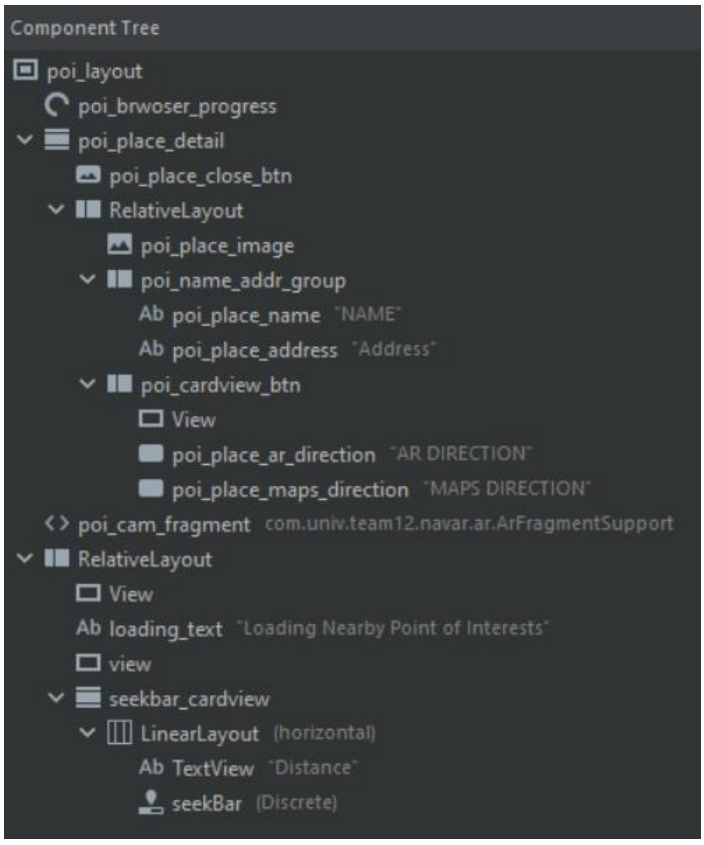

Рисунок 3.12 – Ієрархія UI елементів на основному AR екрані

Для завантаження та відображення view для POI елементів було реалізовано метод Poi_list_call(int radius) , яка на вхід приймає радіус пошуку і виконує запит для завантаження даних і їх подальше відображення (рис.3.13).

```
void Poi_list_call(int radius){
    poi_browser_progress.setVisibility(View.VISIBLE);
    HttpLoggingInterceptor interceptor = new HttpLoggingInterceptor();
    interceptor.setLevel(HttpLoggingInterceptor.Level.BODY);
    OkHttpClient client = new OkHttpClient.Builder().addInterceptor(interceptor).build();

    Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("https://maps.googleapis.com/")
            .client(client)
            .addConverterFactory(GsonConverterFactory.create())
            .build();

    RetrofitInterface apiService =
            retrofit.create(RetrofitInterface.class);

    final Call<PoiResponse> call = apiService.listPOI( location: String.valueOf(mLastLocation.getLatitude())+
            String.valueOf(mLastLocation.getLongitude()),radius,
            getResources().getString(R.string.google_maps_key));

    call.enqueue(new Callback<PoiResponse>() {
        @Override
        public void onResponse(Call<PoiResponse> call, Response<PoiResponse> response) {

            poi_browser_progress.setVisibility(View.GONE);
            seekbar_cardview.setVisibility(View.VISIBLE);

            List<Result> poiResult=response.body().getResults();

            Configure_AR(poiResult);
        }

        @Override
        public void onFailure(Call<PoiResponse> call, Throwable t) {
            poi_browser_progress.setVisibility(View.GONE);
        }
    });

}
```

Рисунок 3.13 – Фрагмент коду для завантаження інформації про POI елементи та їх відображення на екрані

Також було розроблено екран з детальною інформацією про об’єкт (рис. 3.14). Екран відповідає попередньо створеному дизайну. Екран відкривається після кліка по мітці POI елемента.

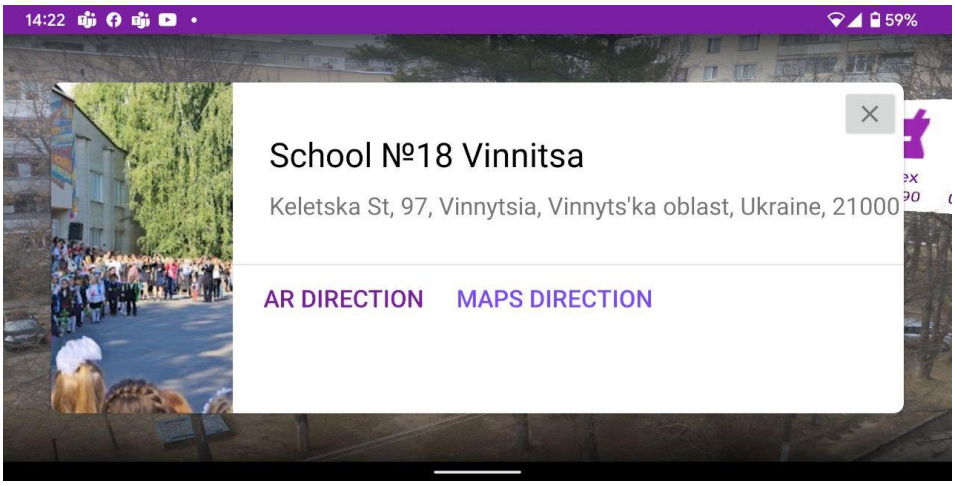

Рисунок 3.14 – Екран з детальною інформацією про обраний об’єкт

Для завантаження інформації на екран з детальною інформацією було реалізовано метод onResponse(Call<PlaceResponse> call, Response <PlaceResponse> response), у якому відбувається завантаження інформації з GoogleMaps API (рис. 3.15).

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

    seekbar_cardview.setVisibility(View.GONE);
    poi_cardview.setVisibility(View.VISIBLE);
    poi_browser_progress.setVisibility(View.GONE);

    final com.univ.team12.navar.network.place.Result result=response.body().getResult();

    poi_place_name.setText(result.getName());
    poi_place_addr.setText(result.getFormattedAddress());

    try {
        HttpUrl url = new HttpUrl.Builder()
                .scheme("https")
                .host("maps.googleapis.com")
                .addPathSegments("maps/api/place/photo")
                .addQueryParameter( name: "maxwidth", value: "400")
                .addQueryParameter( name: "photoreference", result.getPhotos().get(0).getPhotoReference())
                .addQueryParameter( name: "key", getResources().getString(R.string.google_maps_key))
                .build();

        new PoiPhotoAsync().execute(url.toString());

    }catch (Exception e){
        Log.d(TAG, msg: "onResponse: "+e.getMessage());
        Toast.makeText( context: PoiBrowserActivity.this, text: "No image available", Toast.LENGTH_SHORT).show();
    }
```

Рисунок 3.15 – Фрагмент коду методу onResponse

Також було розроблено екран з навігацією за допомогою доповненої реальності. Екран використовує базовий функціонал від екрану з пошуком POI елементів, але замість POI елементів відображається маршрут до обраної точки

Для відображення маршрута на AR екрані було реалізовано метод draw(), у якому відбувається прохід по масиву polylineLatLng з подальшим відображенням розпарсених елементів на AR екрані (рис. 3.16, 3.17).

```
public void draw() {
    List<List<LatLng>> polylineLatLng = new ArrayList<>();
    for (int i = 0; i < steps.length; i++) {
        polylineLatLng.add(i, PolyUtil.decode(steps[i].getPolyline().getPoints()));
        String instructions = steps[i].getHtmlInstructions();
        if (i == 0) {
            GeoObject signObject = new GeoObject( id: 10000 + i);
            signObject.setImageResource(R.drawable.start);
            signObject.setGeoPosition(steps[i].getStartLocation().getLat(),
                    steps[i].getStartLocation().getLng());
            world.addBeyondarObject(signObject);
            Log.d(TAG,  msg: "Configure_AR: START SIGN:" + i);
        }
        if (i == steps.length - 1) {
            GeoObject signObject = new GeoObject( id: 10000 + i);
            signObject.setImageResource(R.drawable.stop);
            LatLng latlng = SphericalUtil.computeOffset(
                    new LatLng(steps[i].getEndLocation().getLat(),
                            steps[i].getEndLocation().getLng()),
                     distance: 4f, SphericalUtil.computeHeading(
                            new LatLng(steps[i].getStartLocation().getLat(),
                                    steps[i].getStartLocation().getLng()),
                            new LatLng(steps[i].getEndLocation().getLat(),
                                    steps[i].getEndLocation().getLng())));
            signObject.setGeoPosition(latlng.latitude, latlng.longitude);
            world.addBeyondarObject(signObject);
            Log.d(TAG,  msg: "Configure_AR: STOP SIGN:" + i);
        }
        if (instructions.contains("right")) {
            Log.d(TAG,  msg: "Configure_AR: " + instructions);
            GeoObject signObject = new GeoObject( id: 10000 + i);
            signObject.setImageResource(R.drawable.turn_right);
            signObject.setGeoPosition(steps[i].getStartLocation().getLat(),
                    steps[i].getStartLocation().getLng());
            world.addBeyondarObject(signObject);
            Log.d(TAG,  msg: "Configure_AR: RIGHT SIGN:" + i);
        } else if (instructions.contains("left")) {
            Log.d(TAG,  msg: "Configure_AR: " + instructions);
            GeoObject signObject = new GeoObject( id: 10000 + i);
            signObject.setImageResource(R.drawable.turn_left);
            signObject.setGeoPosition(steps[i].getStartLocation().getLat(),
                    steps[i].getStartLocation().getLng());
            world.addBeyondarObject(signObject);
            Log.d(TAG,  msg: "Configure_AR: LEFT SIGN:" + i);
        }
    }
```

Рисунок 3.16 – Перша частина методу draw()

```
int temp_polycount = 0;
int temp_inter_polycount = 0;
for (int j = 0; j < polylineLatLng.size(); j++) {
    for (int k = 0; k < polylineLatLng.get(j).size(); k++) {
        GeoObject polyGeoObj = new GeoObject( id: 1000 + temp_polycount++);
        polyGeoObj.setGeoPosition(polylineLatLng.get(j).get(k).latitude, polylineLatLng.get(j).get(k).longitude);
        polyGeoObj.setImageResource(R.drawable.ar_sphere_150x);
        polyGeoObj.setName("arObj" + j + k);
        try { double dist = LocationCalc.haversine(polylineLatLng.get(j).get(k).latitude,
                    polylineLatLng.get(j).get(k).longitude, polylineLatLng.get(j).get(k + 1).latitude,
                    polylineLatLng.get(j).get(k + 1).longitude) * 1000;
            if (dist > 6) {
                int arObj_count = ((int) dist / 3) - 1;
                double bearing = LocationCalc.calcBearing(polylineLatLng.get(j).get(k).latitude,
                        polylineLatLng.get(j).get(k + 1).latitude,polylineLatLng.get(j).get(k).longitude,
                        polylineLatLng.get(j).get(k + 1).longitude);
                double heading = SphericalUtil.computeHeading(new LatLng(polylineLatLng.get(j).get(k).latitude,
                                polylineLatLng.get(j).get(k).longitude),new LatLng(polylineLatLng.get(j).get(k + 1).latitude,
                                polylineLatLng.get(j).get(k + 1).longitude));
                LatLng tempLatLng = SphericalUtil.computeOffset(new LatLng(polylineLatLng.get(j).get(k).latitude,
                                polylineLatLng.get(j).get(k).longitude),  distance: 3f , heading);
                double increment_dist = 3f;
                for (int i = 0; i < arObj_count; i++) {
                    GeoObject inter_polyGeoObj = new GeoObject( id: 5000 + temp_inter_polycount++);
                    if (i > 0 && k < polylineLatLng.get(j).size()) { increment_dist += 3f;
                        tempLatLng = SphericalUtil.computeOffset(new LatLng(polylineLatLng.get(j).get(k).latitude,
                                        polylineLatLng.get(j).get(k).longitude),increment_dist,
                                SphericalUtil.computeHeading(new LatLng(polylineLatLng.get(j).get(k).latitude
                                        , polylineLatLng.get(j).get(k).longitude), new LatLng(polylineLatLng.get(j).get(k + 1).latitude
                                        , polylineLatLng.get(j).get(k + 1).longitude)));
                    }
                    inter_polyGeoObj.setGeoPosition(tempLatLng.latitude, tempLatLng.longitude);
                    inter_polyGeoObj.setImageResource(R.drawable.ar_sphere_default_125x);
                    inter_polyGeoObj.setName("inter_arObj" + j + k + i);
                    world.addBeyondarObject(inter_polyGeoObj);
                }
            }
        } catch (Exception e) {
            Log.d(TAG,  msg: "Configure_AR: EXCEPTION CAUGHT:" + e.getMessage());
        }
        world.addBeyondarObject(polyGeoObj);
    }
```

Рисунок 3.17 – Друга частина методу draw()

В результаті було отримано екран додатку, який відповідає усім поставленим вимогам (рис. 3.18).

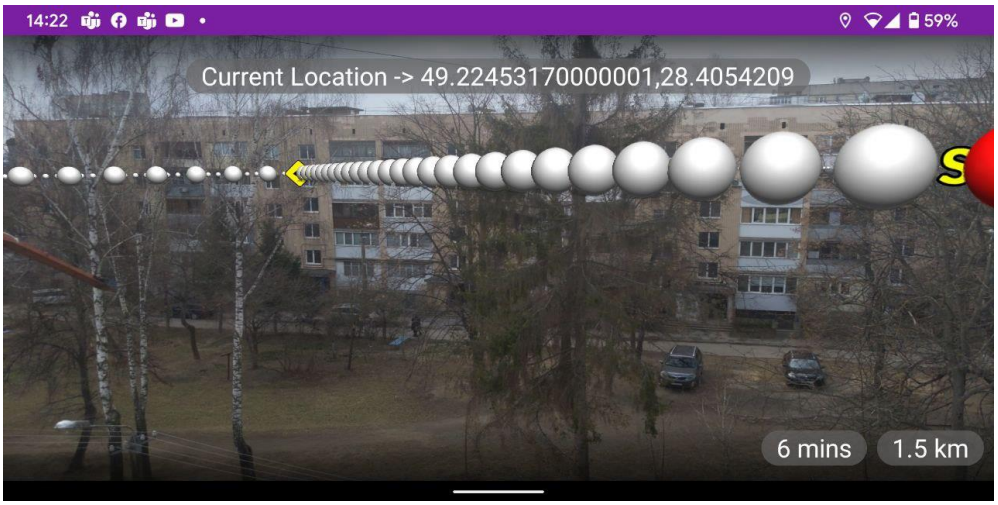

Рисунок 3.18 – Основний екран з прокладеним маршрутом

Насамкінець було створено екран «Про програму» , який містить інформацію про розробника та додаток (рис. 3.19).

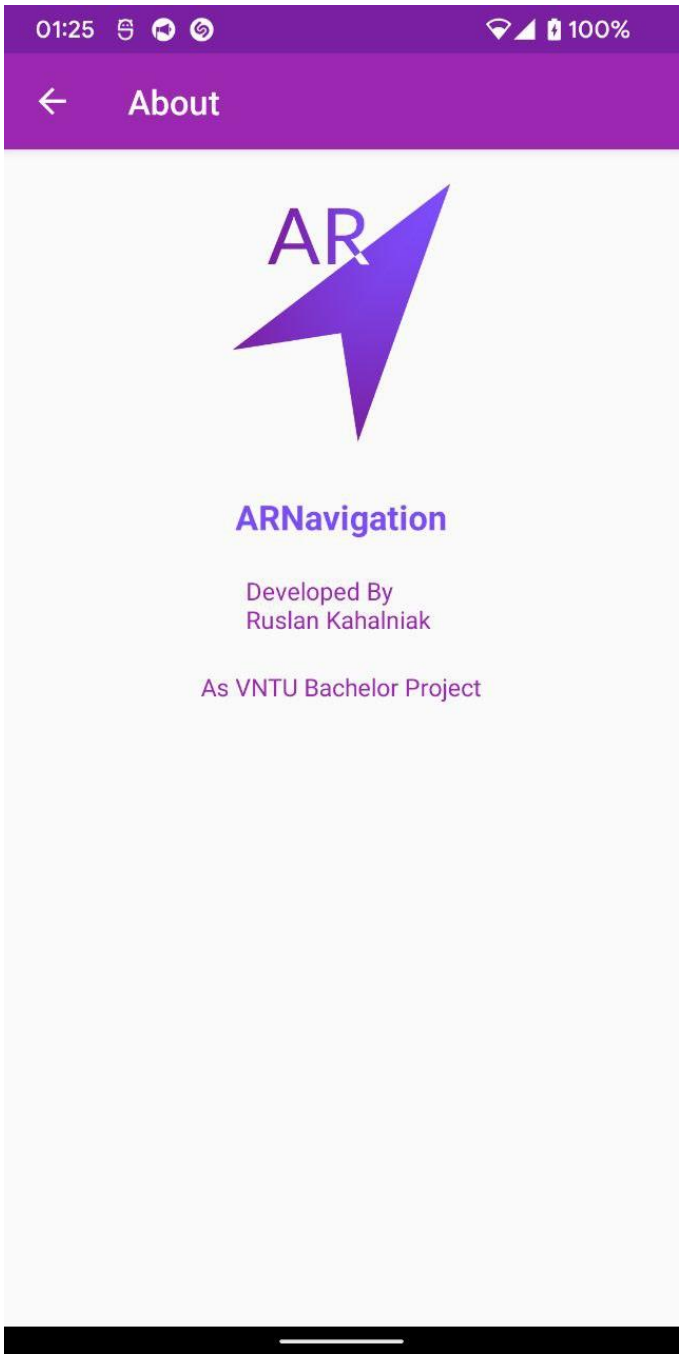

Рисунок 3.19 – Екран «Про програму»

Екран побудовано за допомогою parent елемента <LinearLayout>, який в свою чергу містить вкладений <ScrollView> за рахунок чого при збільшенні інформації на сторінці з’являється можливість прокрутки екрану (рис. 3.20).

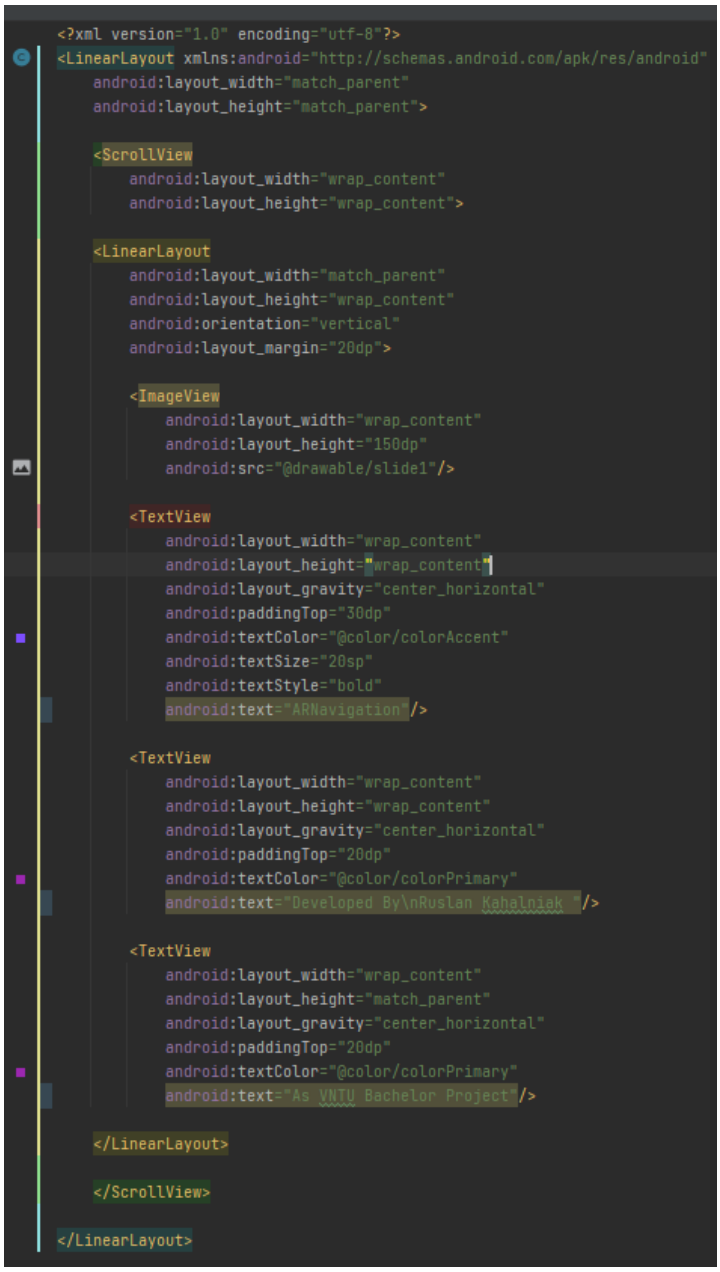

Рисунок 3.20 – Структура екрану «Про програму»

3.3 Висновки

У третьому розділі було обґрунтовано стек технологій, які будуть використані під час розробки мобільного навігатора з доповненою реальністю, а також наведено їхні недоліки та переваги. У результаті було обрано мову програмування Kotlin, IDE Android Studio, систему збирання Gradle, та систему контроля версій Git.

Було описано розробку мобільного навігатора з доповненою реальністю.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

При розробці будь якого програмного продукту завжди проводиться тестування створеного ПЗ . Тестування допомагає виявити недоліки в функціональності продукту та перевірити відповідність додатку до поставлених вимог. Тестування проводять перед тим, як опублікувати додаток, чи впровадити його в роботу [16].

Створення мобільного додатку це доволі складний та комплексний процес, який складається з декількох етапів [17]. Завершаючим етапом є проведення тестування створеного додатку. Дана процедура грає найважливішу роль в створенні ресурсу, так як саме від якості тестування залежить подальше життя проекту. Програмний продукт який містить велику кількість критичних помилок викликає у користувачів розчарування та негатив, в результаті чого довіра до компанії падає, погіршується репутація компанії а головне це несе за собою втрату коштів. Отож тестування грає одну з найважайжливіших ролей при розробці будь якого продукту.

В сучасниому світі для тестування додатків проводять багато варінатів тестування. Це і юніт тестування, інтеграційне тестування, регресійне тестування, та різні види тестування за знанням системи такі як тестування чорного ящика, тестування сірого ящика та тестування білого ящика.

Тестування методом «чорного ящика», також відоме як тестування, засноване на специфікації або тестування поведінки – техніка тестування, заснована на роботі виключно з зовнішніми інтерфейсами тестованої системи.

Тестування методом сірого ящика – метод тестування програмного забезпечення, який передбачає, комбінацію White Box і Black Box підходів [18]. Тобто, внутрішній устрій програми нам відомо лише частково. Передбачається, наприклад, доступ до внутрішньої структури та алгоритмів роботи ПЗ для написання максимально ефективних тест-кейсів, але саме

тестування проводиться за допомогою техніки чорного ящика, тобто, з позиції користувача. Цю техніку тестування також називають методом напівпрозорого ящика: щось ми бачимо, а щось – ні.

Тестування методом білого ящика (також: прозорого, відкритого, скляного ящика; засноване на коді або структурне тестування) – метод тестування програмного забезпечення, який передбачає, що внутрішня структура/пристрій/реалізація системи відомі тестувальнику. Ми вибираємо вхідні значення, грунтуючись на знанні коду, який буде їх обробляти. Точно так само ми знаємо, яким повинен бути результат цієї обробки. Знання всіх особливостей тестованої програми та її реалізації – обов'язкові для цієї техніки. Тестування білого ящика – поглиблення у код системи, за межі її зовнішніх інтерфейсів.

Тестування сумісності найважливіше тоді, коли справа доходить до тестування мобільних додатків. Мета тестування мобільного додатку на сумісність, як правило, полягає в тому, щоб ключові функції програми працювали належним чином на конкретному пристрої. Сама сумісність повинна займати всього кілька хвилин і може бути заздалегідь спланованою. Прийняти рішення, які тести на сумісність мобільних пристроїв слід виконати, є нелегким завданням (оскільки тестування на усіх існуючих пристроях просто неможливе). Тому необхідно підготувати тестову матрицю з кожною можливою комбінацією та розставити пріоритети для клієнта.

Для тестування було обрано метод чорного ящика, оскільки це швидше та дозволяє не вникати у деталі реалізації, а просто зрівнювати результат фактичний, з тим, який має бути після тестування.

4.2 Тестування розробленого програмного продукту

Спочатку було протестовано головну сторінку на відповідність елементів до раніше розроблених дизайнів та було протестовано функціональність розробленого екрану та відображення карти (рис. 4.1).

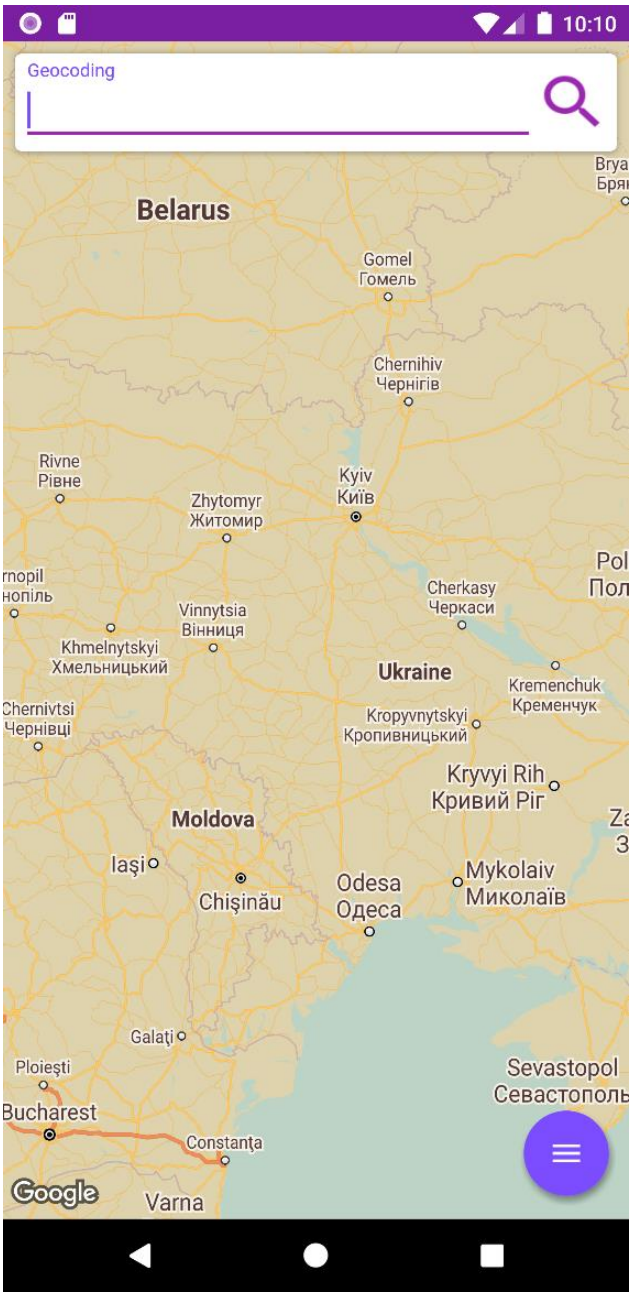

Рисунок 4.1 – Тестування функціональності головної сторінки

Далі було перевірено перехід з головного екрану на екран вибору початкової адреси і шуканого місця, а також перевірено працездатність елементів для вибору користувача (рис. 4.2).

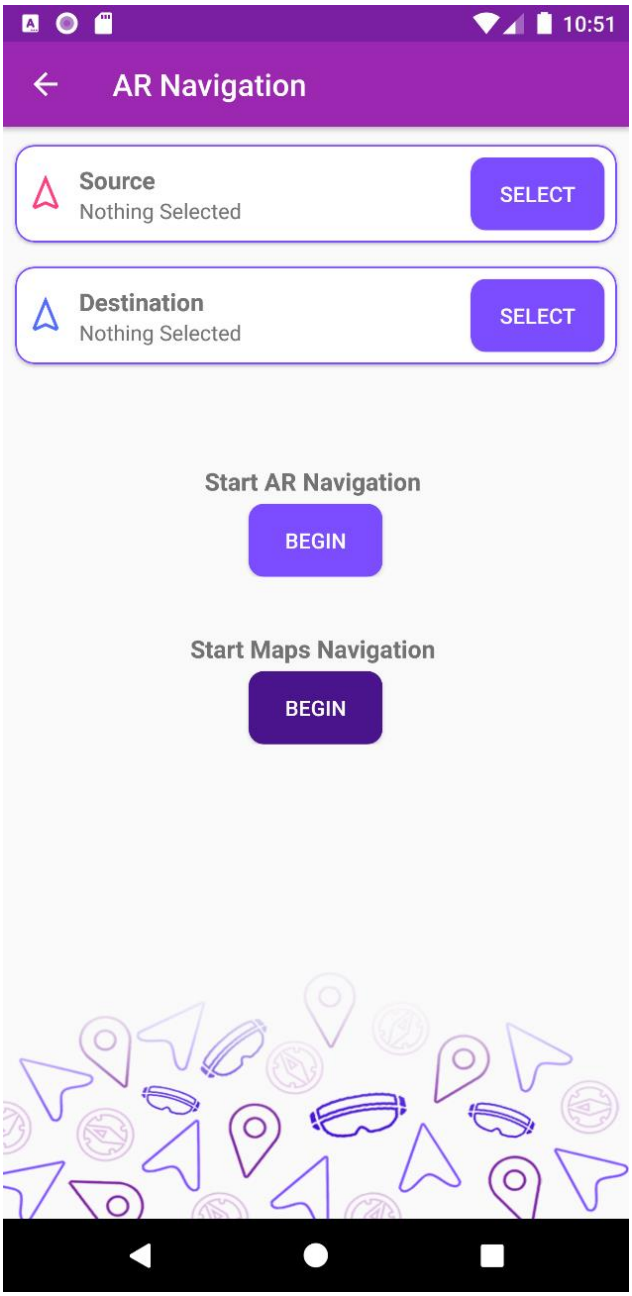

Рисунок 4.3 – Тестування екрану вибору локацій та режимів

При кліку на кнопки «SELECT» повинен відкриватись додатковий екран для вибору локації на мапі. Було перевірено відкривання екрану а також достовірність вибору локації і її відображення (рис. 4.4).

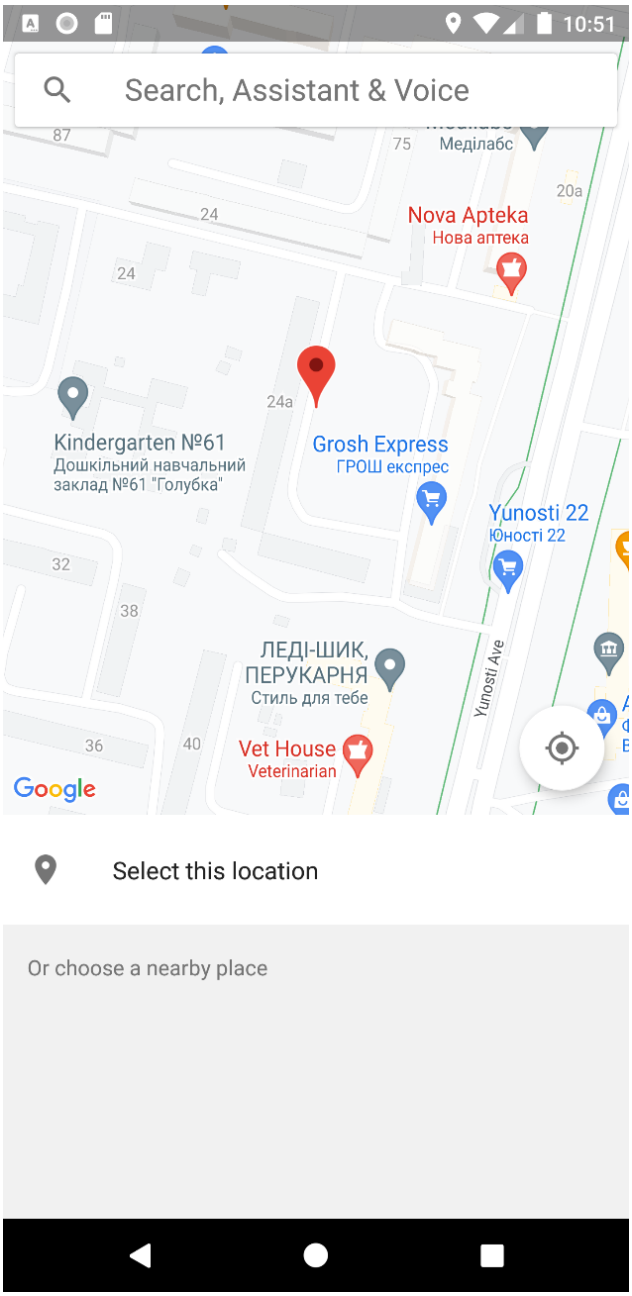

Рисунок 4.4 – Тестування екрану вибору локацій

Також було протестовано роботу функції «Start Map Navigation». При кліку на відповідну кнопку «BEGIN» повинна відкритись Google Map чи інша дефолтна навігаційна програма і прокладено маршрут між попередньо обраними точками (рис. 4.5).

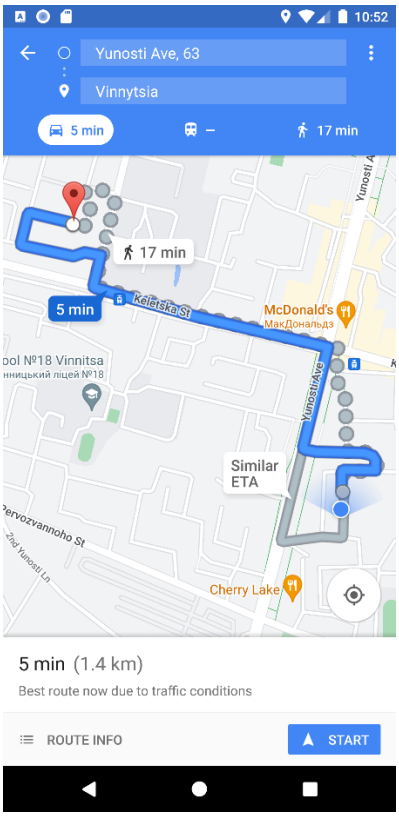

Рисунок 4.5 – Результат виконання функції «Start Map Navigation»

Також було протестовано роботу функції «Start AR Navigation». При кліку на відповідну кнопку «BEGIN» повинен відкритись AR-екран з навігацією між обраними точками. Тестування було проведено на емуляторі , з замоканими функціями камери та GPS щоб виключити ймовірність впливу на тест змінних величин. Для усіх проведених тестів було обрано статичну локацію девайса (рис. 4.6).

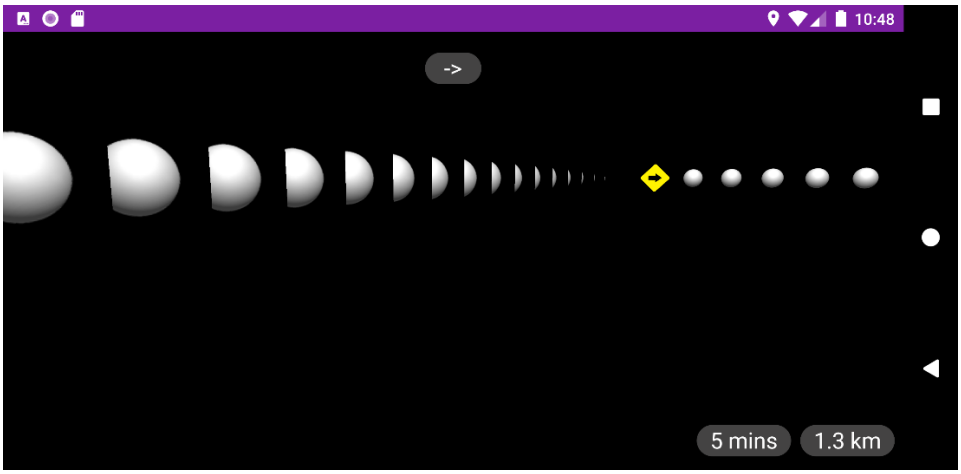

Рисунок 4.6 – Тестування функції «Start AR Navigation» з замоканими даними

Далі було протестовано екран з пошуком POI (Point of interest) об’єктів. Для тестування даного екрану також було використано емулятор , статичну локацію, та віртуальну камеру (рис. 4.7).

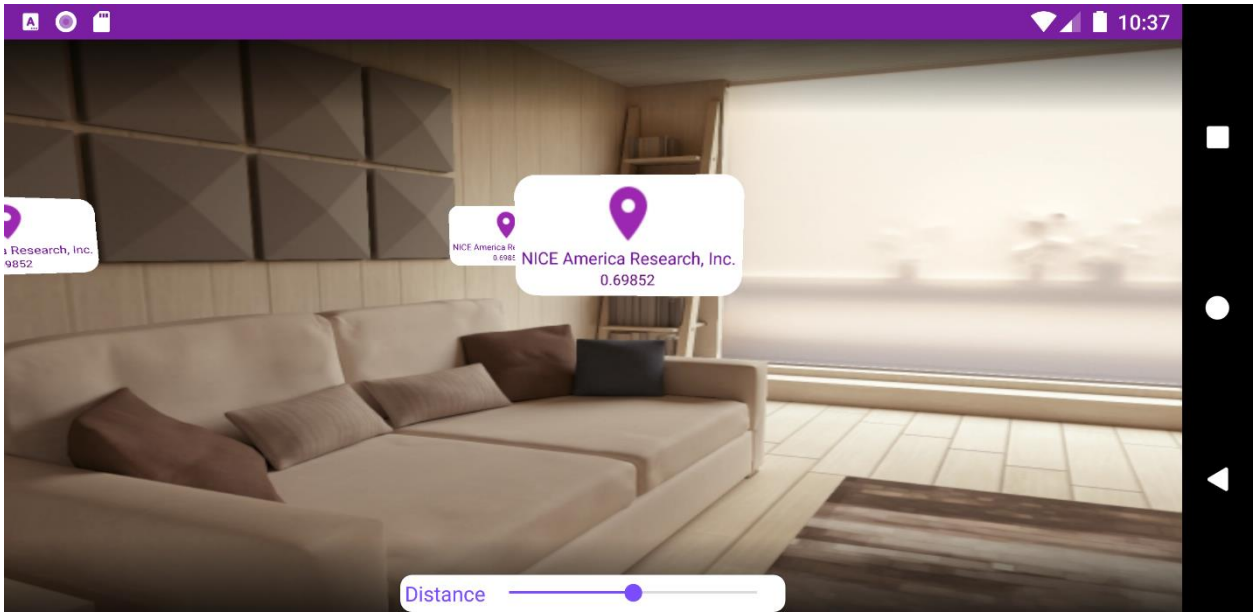

Рисунок 4.7 – Тестування екрану для пошуку POI об’єктів

Було проведено тестування опрацювання кліків користувача на view POI об’єкта. При кліку на відповідний об’єкт повинен відкритись діалог з детальною інформацією про об’єкт , фото та варіантами навігації (рис. 4.8).

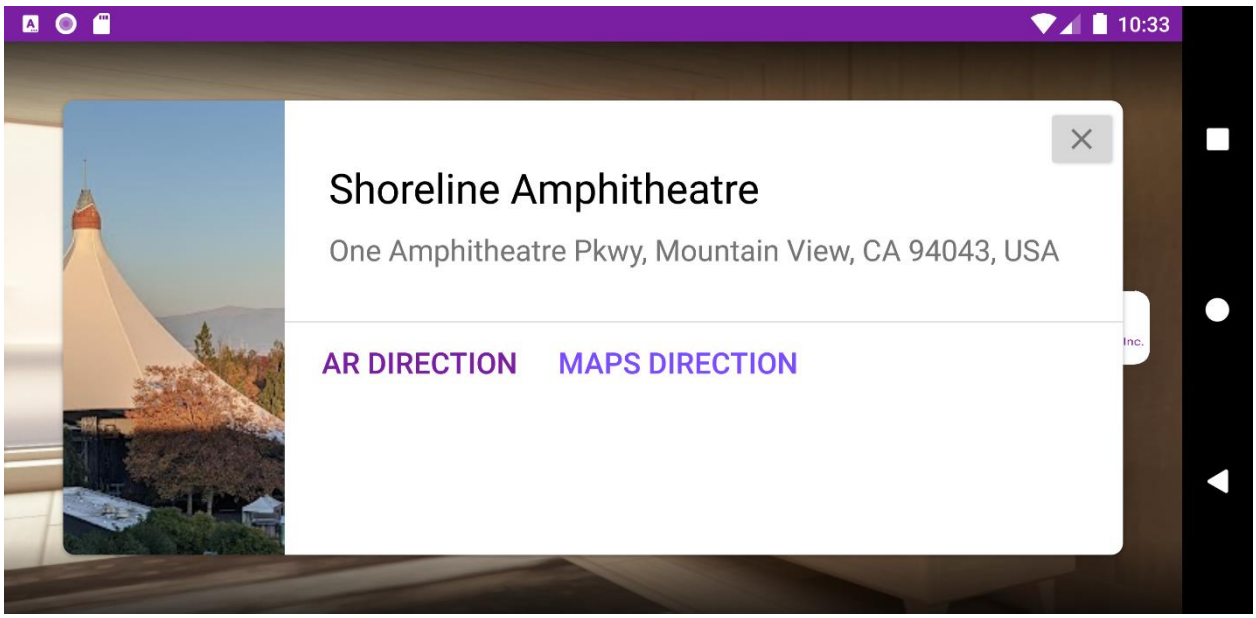

Рисунок 4.8 – Тестування показу діалогу з детальною інформацією

Під час тестування не було виявлено явних недоліків, додаток відповідає поставленим вимогам. Закладена у додаток функціональність працює правильно.

4.3 Розробка інструкцій користувача

Розробка інструкції також є важливим етапом при розробці будь якого програмного продукту, адже саме завдяки інструкції користувач дізнається про функціональність додатку та алгоритм роботи продукту.

Для початку роботи з додатком користувач повинен встановити його на свій мобільний пристрій. Додаток встановлюється шляхом відкриття apk файлу з програмою.

В якості інструкції користувача при першому відкритті програми було створено екрани онбордингу нового користувача, який містить опис та інформацію про основні функції додатку (рис. 4.9).

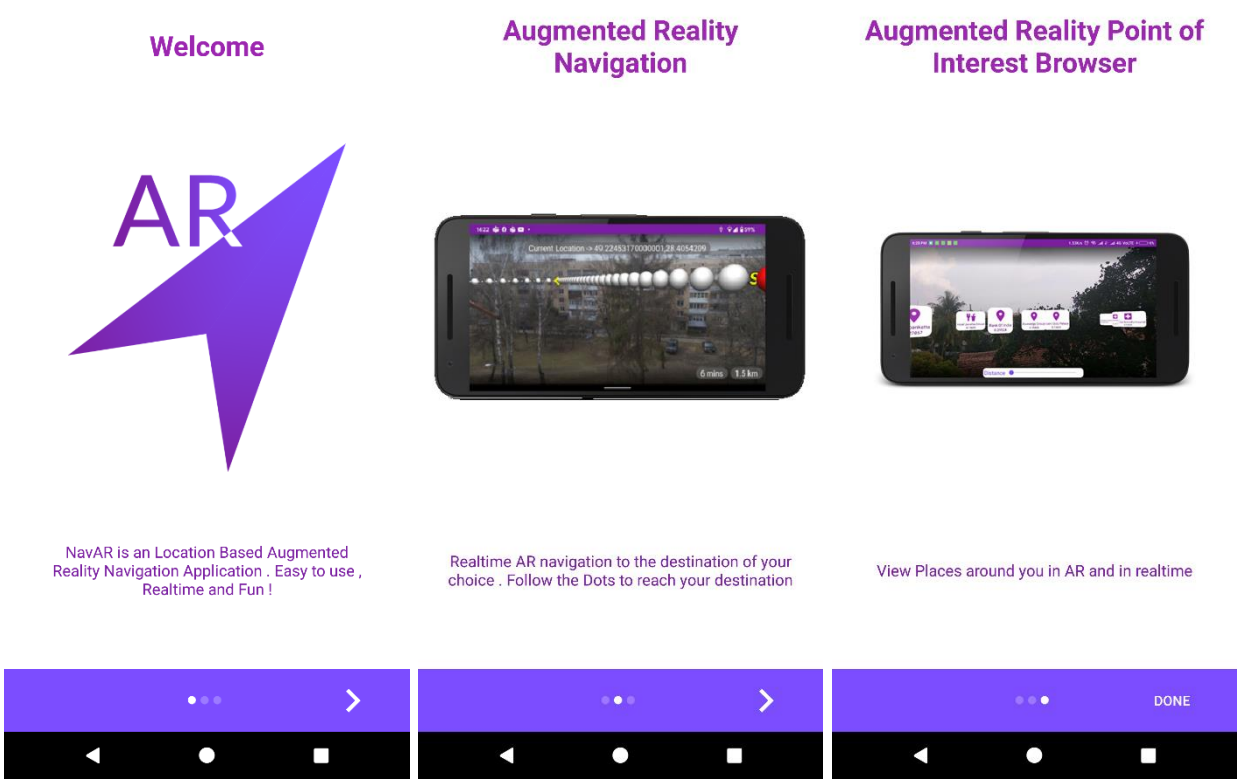

Рисунок 4.9 – Екрани з інструкцією для користувача

4.3 Висновки

Отже, в результаті було проведено тестування додатку методом «чорного ящика» і отримано позитивні результати. Розроблено інструкцію користувача з описом основних функціональних можливостей додатку.

ВИСНОВКИ

У процесі проходження практики було розроблено програмний додаток – мобільний навігатор з доповненою реальністю. Для розробки було використано середовище програмування Android Studio.

Було проаналізовано стан даної проблеми на сьогоднішній день. Розглянуто основні аналоги розроблюваного додатку, визначено їх переваги та недоліки і розроблено порівняння з власним програмним продуктом.

У результаті проходження практики було зроблено наступне:

- інтегровано технологію доповненої реальності у мобільний навігатор;
- розроблено алгоритми завантаження та відображення карт у мобільному додатку;
- розроблено адаптивний графічний інтерфейс користувача;
- розроблено програмний модуль навігатора з доповненою реальністю;
- проведено тестування програмного модуля.

Перед створенням програмного продукту було розроблено схеми загального алгоритму роботи додатку, обробки вхідних та вихідних даних. Розроблено модуль для інтеграції технології доповненої реальності у мобільний додаток.

Тестування програми довело повну працездатність даного програмного продукту та відповідність поставленому завданню.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. GPS [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/GPS

2. Google Maps [Електронний ресурс]. – Режим доступу: https://www.google.com.ua/maps/

3. Apple Maps [Електронний ресурс]. – Режим доступу: https://apps.apple.com/us/app/apple-maps/id915056765

4. ARCity [Електронний ресурс]. – Режим доступу: https://www.blippar.com/blog/2017/11/06/welcome-ar-city-future-maps-and-navigation

5. Кагальняк Р.Ю. Бабюк Н.П. Порівняння мобільних AR навігаторів. / Молодь в науці: дослідження, проблеми, перспективи (МН-2022), Вінниця: ВНТУ, 2022. 2 с.

6. Шилдт Г. Java 8. Керівництво для початківців: пер. з англ. М. Вільямс / Г. Шилдт – Санкт-Петербург: Вільямс, 2015. – 712 с.

7. Mapbox [Електронний ресурс]. – Режим доступу: https://www.mapbox.com/

8. Google Maps API [Електронний ресурс]. – Режим доступу: https://console.cloud.google.com/google/maps-apis/welcome

9. POI (Point of interest) [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/POI

10. Kotlin [Електронний ресурс]. – Режим доступу: https://kotlinlang.org/

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

12. Gradle [Електронний ресурс]. – Режим доступу: https://gradle.org/

13. Романюк О. Н. Веб-дизайн і комп’ютерна графіка / О. Н. Романюк, Д. І. Катєльніков, О. П. Косовець. – Вінниця, 2007. – 142 с.

14. POI [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/POI

15. Permissions on Android [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/topics/permissions/overview

16. Типи тестування [Електронний ресурс] – Режим доступу до ресурсу: https://sqa.lviv.ua/yaki-ye-typy-testuvannya.

17. Тестування мобільних додатків [Електронний ресурс] – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/testuvannya-mobilnih-dodatkiv/

18. White/Black/Grey Box-тестирование [Електронний ресурс] – Режим доступу до ресурсу: https://qalight.ua/ru/baza-znaniy/white-black-grey-box-testirovanie/

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"25" березня 2022 р.

Технічне завдання

на бакалаврську дипломну роботу

«Розробка програмного засобу-навігатора з доповненою реальністю»

за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:

к.т.н., доцент кафедри ПЗ Бабюк Н. П

"____"_____2022 р.

Виконав:

студент гр. 2ПІ-18б Кагальняк Р. Ю.

"____"_____2022 р.

Вінниця – 2022 року

1 Найменування та галузь застосування.

Бакалаврська робота: «Розробка програмного засобу-навігатора з доповненою реальністю».

Сфера застосування – навчальна(інформаційні технології).

2 Підстава для розробки.

Підставою для розробки бакалаврської дипломної роботи є рішення засідання кафедри програмного забезпечення (протокол № 65 від "25" березня 2022 року).

3 Мета та призначення розробки.

Мета виконання бакалаврської дипломної роботи – спрощення процесу навігації, завдяки розробці навігатора з доповненою реальністю.

Призначення роботи – розробка програмного засобу – навігатора з доповненою реальністю

4 Вихідні дані для проведення НДР.

Вихідні дані для розробки є індивідуальне завдання на бакалаврську дипломну роботу на розробку програмного продукту для вивчення мов програмування.

5 Технічні вимоги.

Методи навігації за допомогою мобільного пристрою; архітектура мобільного додатку; відображення карти; формування маршрутів; сформований мобільний додаток.

6 Конструктивні вимоги.

Мобільний додаток має відповідати ергономічним та технічним вимогам. текстова та графічна документація повинна відповідати стандартам України.

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

- пояснювальна записка до бакалаврської дипломної роботи.
- технічне завдання
- лістинг програми.

8 Вимоги до рівня уніфікації та стандартизації.

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9 Стадії та етапи розробки.

№ з/п	Назва етапів дипломної бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору методу розробки та постановка задач дослідження		Вик.
2	Розробка структури та алгоритмів програмного продукту		Вик.
3	Розробка модулів програмного продукту		Вик.
4	Тестування програми		Вик.
5	Оформлення матеріалів до захисту БДР		Вик.

10 Порядок контролю та прийняття.

Всі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту. Дозволяється корегування бакалаврської дипломної роботи.

Додаток Б – Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Розробка програмного додатку для покращення тайм менеджменту користувача

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доцент Бабюк Наталя Петрівна

Оригінальність	
Схожість	

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 _____ Богач К.С.

Керівник роботи _____ Бабюк Н.П.

Додаток В – Лістинг програми

```
package com.kahalniak.bakalavr.arnavigation.network;

import retrofit2.Call;
import retrofit2.http.GET;
import retrofit2.http.Query;

public interface RetrofitInterface {

    @GET("maps/api/directions/json?")
    Call<DirectionsResponse> getDirections(
            @Query("origin") String origin,
            @Query("destination") String destination,
            @Query("key") String key
    );

    @GET("/maps/api/place/nearbysearch/json?")
    Call<PoiResponse> listPOI(
            @Query("location") String location,
            @Query("radius") int radius,
            @Query("key") String key
    );

    @GET("/maps/api/place/details/json?")
    Call<PlaceResponse> getPlaceDetail(
            @Query("placeid") String location,
            @Query("key") String key
    );

    @GET("/maps/api/geocode/json?")
    Call<GeocodeResponse> getGecodeData(
            @Query("address")String address,
            @Query("key")String key
    );

    @GET("/maps/api/geocode/json?")
    Call<GeocodeResponse> getRevGecodeData(
            @Query("latlng")String latlng,
            @Query("key")String key
    );
}



package com.kahalniak.bakalavr.arnavigation;

import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.content.res.Resources;
import android.preference.PreferenceManager;
```

```
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

import com.google.android.material.snackbar.Snackbar;

import androidx.fragment.app.FragmentActivity;

import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.Toast;

import com.github.clans.fab.FloatingActionMenu;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationServices;
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.LatLng;
import com.google.android.gms.maps.model.MapStyleOptions;
import
com.google.android.gms.common.api.GoogleApiClient.OnConnectionFa
iledListener;
import
com.google.android.gms.common.api.GoogleApiClient.ConnectionCall
backs;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import
com.kahalniak.bakalavr.arnavigation.network.GeocodeResponse;
import
com.kahalniak.bakalavr.arnavigation.network.RetrofitInterface;
import
com.kahalniak.bakalavr.arnavigation.network.geocode.Location;
import
com.kahalniak.bakalavr.arnavigation.network.geocode.Result;
import
com.kahalniak.bakalavr.arnavigation.onboarding.DefaultIntro;
import com.kahalniak.bakalavr.arnavigation.utils.UtilsCheck;
import
com.kahalniak.bakalavr.arnavigation.utils.PermissionCheck;

import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;
```

```
import okhttp3.OkHttpClient;
import okhttp3.logging.HttpLoggingInterceptor;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;


public class MapsActivity extends FragmentActivity implements
OnMapReadyCallback, OnConnectionFailedListener
        , ConnectionCallbacks, GoogleMap.OnMapLongClickListener,
GoogleMap.OnMapClickListener {

    private final static String TAG = "MapsActivity";

    SharedPreferences getPrefs;
    boolean isFirstStart;

    private GoogleApiClient googleApiClient;
    private GoogleMap mMap;

    private Location location;

    private Marker RevMarker;

    @BindView(R.id.fab_menu_btn)
    FloatingActionMenu fab_menu;
    @BindView(R.id.ar_nav_btn)
    com.github.clans.fab.FloatingActionButton ar_nav_btn;
    @BindView(R.id.poi_browser_btn)
    com.github.clans.fab.FloatingActionButton poi_browser_btn;
    @BindView(R.id.decode_box)
    EditText decode_editText;
    @BindView(R.id.decode_btn)
    Button decode_button;
    @BindView(R.id.progressBar_maps)
    ProgressBar progressBar;
    @BindView(R.id.about_btn)
    com.github.clans.fab.FloatingActionButton about_btn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        ButterKnife.bind(this);

        Init_intro();
        PermissionCheck.initialPermissionCheckAll(this, this);

        progressBar.setVisibility(View.GONE);

        if (!UtilsCheck.isNetworkConnected(this)) {
```

```
            Snackbar mySnackbar =
Snackbar.make(findViewById(R.id.main_content),
                    "Turn Internet On", Snackbar.LENGTH_SHORT);
            mySnackbar.show();
        }

        if (googleApiClient == null) {
            googleApiClient = new GoogleApiClient.Builder(this)
                    .addApi(LocationServices.API)
                    .addConnectionCallbacks(this)
                    .addOnConnectionFailedListener(this)
                    .build();
        }

        ar_nav_btn.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(MapsActivity.this,
NavActivity.class);
                startActivity(intent);
            }
        });

        poi_browser_btn.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(MapsActivity.this,
PoiBrowserActivity.class);
                startActivity(intent);
            }
        });

        about_btn.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(MapsActivity.this,
AboutActivity.class);
                startActivity(intent);
            }
        });

        decode_button.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                try {
                    if
(TextUtils.isEmpty(decode_editText.getText())) {
                        Snackbar mySnackbar =
Snackbar.make(findViewById(R.id.main_content),
```

```
                                    "Search Field is Empty",
Snackbar.LENGTH_SHORT);
                            mySnackbar.show();
                        } else {

Geocode_Call(decode_editText.getText().toString());
                        }
                    } catch (NullPointerException npe) {
                        Snackbar mySnackbar =
Snackbar.make(findViewById(R.id.main_content),
                                "Search Field is Empty",
Snackbar.LENGTH_SHORT);
                        mySnackbar.show();
                    }
                }
            });

            SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map);
            mapFragment.getMapAsync(this);
        }

        void Geocode_Call(String address) {
            HttpLoggingInterceptor interceptor = new
HttpLoggingInterceptor();
            interceptor.setLevel(HttpLoggingInterceptor.Level.BODY);
            OkHttpClient client = new
OkHttpClient.Builder().addInterceptor(interceptor).build();

            progressBar.setVisibility(View.VISIBLE);

            Retrofit retrofit = new Retrofit.Builder()

.baseUrl(getResources().getString(R.string.directions_base_url))
                    .client(client)

.addConverterFactory(GsonConverterFactory.create())
                    .build();

            RetrofitInterface apiService =
                    retrofit.create(RetrofitInterface.class);

            final Call<GeocodeResponse> call =
apiService.getGecodeData(address,

getResources().getString(R.string.google_maps_key));

            call.enqueue(new Callback<GeocodeResponse>() {
                @Override
                public void onResponse(Call<GeocodeResponse> call,
Response<GeocodeResponse> response) {

                    progressBar.setVisibility(View.GONE);
```

```
                List<Result> results =
response.body().getResults();
                if (!results.isEmpty()) {
                    location =
results.get(0).getGeometry().getLocation();

                    Toast.makeText(MapsActivity.this,
location.getLat() + "," + location.getLng(),
Toast.LENGTH_SHORT).show();

                    try {
                        mMap.clear();
                        LatLng loc = new
LatLng(location.getLat(), location.getLng());
                        mMap.addMarker(new MarkerOptions()
                                .position(loc)

.title(results.get(0).getFormattedAddress())

.snippet(results.get(0).getGeometry().getLocationType()));

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(loc, 14.0f));

mMap.getUiSettings().setMapToolbarEnabled(false);
                    } catch (NullPointerException npe) {
                        Log.d(TAG, "onMapReady: Location is
NULL");
                    }
                }
            }

            @Override
            public void onFailure(Call<GeocodeResponse> call,
Throwable t) {
                progressBar.setVisibility(View.GONE);
                Toast.makeText(MapsActivity.this, "Invalid
Request", Toast.LENGTH_SHORT).show();
            }
        });

    }

    void Rev_Geocode_Call(LatLng latlng) {
        HttpLoggingInterceptor interceptor = new
HttpLoggingInterceptor();
        interceptor.setLevel(HttpLoggingInterceptor.Level.BODY);
        OkHttpClient client = new
OkHttpClient.Builder().addInterceptor(interceptor).build();

        Retrofit retrofit = new Retrofit.Builder()

.baseUrl(getResources().getString(R.string.directions_base_url))
```

```
                .client(client)

.addConverterFactory(GsonConverterFactory.create())
                .build();

        progressBar.setVisibility(View.VISIBLE);

        RetrofitInterface apiService =
                retrofit.create(RetrofitInterface.class);

        final Call<GeocodeResponse> call =
apiService.getRevGecodeData(latlng.latitude + "," +
latlng.longitude,

getResources().getString(R.string.google_maps_key));

        call.enqueue(new Callback<GeocodeResponse>() {
            @Override
            public void onResponse(Call<GeocodeResponse> call,
Response<GeocodeResponse> response) {

                progressBar.setVisibility(View.GONE);

List<com.kahalniak.bakalavr.arnavigation.network.geocode.Result>
results = response.body().getResults();
                String address =
results.get(0).getFormattedAddress();
                Toast.makeText(MapsActivity.this, address,
Toast.LENGTH_SHORT).show();

                RevMarker.setTitle(address);

RevMarker.setSnippet(results.get(0).getGeometry().getLocationTyp
e());
            }

            @Override
            public void onFailure(Call<GeocodeResponse> call,
Throwable t) {
                progressBar.setVisibility(View.GONE);
                Toast.makeText(MapsActivity.this, "Invalid
Request", Toast.LENGTH_SHORT).show();
            }
        });

    }

    void Init_intro() {

        getPrefs = PreferenceManager
                .getDefaultSharedPreferences(getBaseContext());

        isFirstStart = getPrefs.getBoolean("firstStart", true);
```

```
        Thread t = new Thread(new Runnable() {
            @Override
            public void run() {

                Intent i = new Intent(MapsActivity.this,
DefaultIntro.class);
                startActivity(i);

                SharedPreferences.Editor e = getPrefs.edit();

                e.putBoolean("firstStart", false);

                e.apply();
            }
        });

        if (isFirstStart) {
            t.start();
        }
    }

    @Override
    public void onRequestPermissionsResult(int
requestCode,String permissions[], int[] grantResults) {
        switch (requestCode) {
            case 10: {
                if (grantResults.length > 0
                        && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                } else {
                }
                return;
            }
        }
    }


    @Override
    public void onMapLongClick(LatLng latLng) {

        mMap.clear();

        RevMarker = mMap.addMarker(new
MarkerOptions().position(latLng));
        Toast.makeText(this, latLng.latitude + " " +
latLng.longitude, Toast.LENGTH_SHORT).show();
        Rev_Geocode_Call(latLng);
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
```

```
        try {
            boolean success = googleMap.setMapStyle(
                    MapStyleOptions.loadRawResourceStyle(
                            this, R.raw.style_json));
            if (!success) {
                Log.e(TAG, "Style parsing failed.");
            }
        } catch (Resources.NotFoundException e) {
            Log.e(TAG, "Can't find style. Error: ", e);
        }
        mMap.setOnMapLongClickListener(this);
    }

    @Override
    protected void onStart() {
        googleApiClient.connect();
        super.onStart();
    }

    @Override
    protected void onStop() {
        super.onStop();
        googleApiClient.disconnect();
    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult
connectionResult) {

    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        //PermissionCheck.initialPermissionCheck(this,this);

    }

    @Override
    public void onConnectionSuspended(int i) {

    }

    @Override
    public void onMapClick(LatLng latLng) {
        Log.d(TAG, "onMapClick: Short Click " +
latLng.toString());
    }
}



package com.kahalniak.bakalavr.arnavigation.network;
```

```
import com.google.gson.annotations.Expose;
import com.google.gson.annotations.SerializedName;
import
com.kahalniak.bakalavr.arnavigation.network.model.GeocodedWaypoi
nt;
import com.kahalniak.bakalavr.arnavigation.network.model.Route;

import java.util.List;

public class DirectionsResponse {

    @SerializedName("geocoded_waypoints")
    @Expose
    private List<GeocodedWaypoint> geocodedWaypoints = null;
    @SerializedName("routes")
    @Expose
    private List<Route> routes = null;
    @SerializedName("status")
    @Expose
    private String status;

    public List<GeocodedWaypoint> getGeocodedWaypoints() {
        return geocodedWaypoints;
    }

    public void setGeocodedWaypoints(List<GeocodedWaypoint>
geocodedWaypoints) {
        this.geocodedWaypoints = geocodedWaypoints;
    }

    public List<Route> getRoutes() {
        return routes;
    }

    public void setRoutes(List<Route> routes) {
        this.routes = routes;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

}
```

Додаток В – Графічна частина

ГРАФІЧНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ-НАВІГАТОРА З ДОПОВНЕНОЮ РЕАЛЬНІСТЮ