Державний торговельно-економічний університет
Кафедра інженерії програмногозабезпечення та кібербезпеки

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Програмний додаток контролю дотримання часового режиму руху громадського транспорту»

Студента 2м курсу, 2 групи, спеціальності 121 «Інженерія програмного забезпечення» освітньої програми «Інженерія програмного забезпечення» _____ Пономаренка Сергія Валерійовича
підпис студента

Науковий керівник кандидат педагогічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки _____ Котенко Наталія Олексіївна
підпис керівника

Гарант освітньої програми кандидат педагогічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки _____ Котенко Наталія Олексіївна
підпис гаранта

КИЇВ – 2023

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь магістр
Освітня програма 121 «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
«13» грудня 2022 р.

Завдання
на випускну кваліфікаційну роботу студентові

Пономаренку Сергію Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи «Програмний додаток контролю дотримання часового режиму руху громадського транспорту»
Затверджена наказом ректора від «06»грудня 2022 р. № 3285
2. Строк здачі студентом закінченої роботи 1 грудня 2023
3. Цільова установка та вихідні дані до роботи
Мета роботи: Створення мобільного додатка для Android, який дозволить відстежувати графік громадського транспорту та надавати пасажирам легкий доступ до актуальних розкладів, з метою забезпечення точного виконання графіку руху.
Об'єкт дослідження: Програмний додаток для контролю дотримання часового руху громадського транспорут на платформі Android.
Предмет дослідження: Розробка функціоналу, дизайну та архітектури мобільного додатка для контролю графіку громадського транспорту на платформі Android.

4. Консультанти роботи із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1 ДОСЛІДЖЕННЯ НА АНАЛІЗ ПРОБЛЕМАТИКИ У ВИБРАНІЙ ОБЛАСТІ

1.1.Огляд сучасних підходів до розробки мобільних додатків на Kotlin

1.2. Опис структури та технологій розробки

1.3. Аналіз методів та засобів розробки

1.4.Висновки до розділу 1

РОЗДІЛ 2 КОНЦЕПЦІЯ ТА ОПИС ПРОГРАМНОГО ДОДАТКУ

2.1. Загальний огляд програмного додатку

2.2. Оптимізація та ефективність програмного коду

2.3. Огляд існуючих аналогів серед мобільних додатків

2.3.1.EasyWay

2.3.2.D-transport

2.4. Висновки до розділу 2

РОЗДІЛ 3 ТЕХНІЧНІ ЗАСОБИ ТА ІНСТРУМЕНТИ РОЗРОБКИ

3.1. Глибокий аналіз вибору Kotlin

3.2. Обґрунтування використання Figma для дизайну

3.3. Вибір платформи Android Studio та її роль у розробці

3.4. Переваги та виклики роботи з Android SDK

3.5. Висновки до розділу 3

РОЗДІЛ 4 ТЕХНІЧНА РЕАЛІЗАЦІЯ ТА РОЗРОБКА ДОДАТКУ

4.1. Характеристика інформаційного забезпечення

4.2. Технічна реалізація

4.3. Розробка інтерфейсу користувача

4.4. Програмування та реалізація функціональності
4.5. Висновки до розділу 4
ВИСНОВКИ ТА ПРОПОЗИЦІЇ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ТЕХНІЧНЕ ЗАВДАННЯ
ТЕСТУВАННЯ ДОДАТКА
ДОДАТКИ

6. Календарний план виконання роботи

№ пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускної кваліфікаційної роботи*	*07.11.2022*	*07.11.2022*
2.	*Розробка та затвердження завдання на роботу магістра (стац/заоч)*	*13.12.2022*	*13.12.2022*
3.	*Вступ та перелік літературних джерел*	*24.02.2023*	*24.02.2023*
4.	*Розробка технічного завдання*	*15.03.2023*	*15.03.2023*
5.	*Розділ 1. ДОСЛІДЖЕННЯ НА АНАЛІЗ ПРОБЛЕМАТИКИ У ВИБРАНІЙ ОБЛАСТІ*	*10.04.2023*	*10.04.2023*
6.	*Розділ 2. КОНЦЕПЦІЯ ТА ОПИС ПРОГРАМНОГО ДОДАТКУ*	*24.05.2023*	*24.05.2023*
7.	*Розділ 3. ТЕХНІЧНІ ЗАСОБИ ТА ІНСТРУМЕНТИ РОЗРОБКИ*	*06.09.2023*	*06.09.2023*
8.	*Розділ 4. ТЕХНІЧНА РЕАЛІЗАЦІЯ ТА РОЗРОБКА ДОДАТКУ*	*06.09.2023*	*06.09.2023*
9.	*Розробка програми та методики тестування*	*18.10.2023*	*18.10.2023*
10.	*Написання наукової статті*	*17.05.2023*	*17.05.2023*
11.	*Керівництво користувача*	*25.10.2023*	*25.10.2023*
12.	*Висновки та пропозиції*	*01.11.2023*	*01.11.2023*
13.	*Здача випускної кваліфікаційної роботи на кафедру (перша перевірка)*	*06.11.2023*	*06.11.2023*
14.	*Підготовка автореферату та презентації доповіді*	*06.11.2023*	*06.11.2023*
15.	*Попередній захист випускної кваліфікаційної роботи*	*20.11.2023 – 24.11.2023*	*20.11.2023 – 24.11.2023*
16.	*Здача зброшурованої випускної кваліфікаційної роботи*	*01.12.2023*	*01.12.2023*
17.	*Зовнішнє рецензування випускної кваліфікаційної роботи*	*02.12.2023*	*02.12.2023*
18.	*Підготовка до публічного захисту випускної кваліфікаційної роботи*	*05.12.2023-06.12.2023*	*05.12.2023-06.12.2023*

7. Дата видачі завдання «13» грудня 2022 р.

8. Науковий керівник випускної кваліфікаційної роботи Котенко Н.О.
(прізвище, ініціали, підпис)

9. Гарант освітньої програми Котенко Н.О.
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Пономаренко С.В.
(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проєкту

Науковий керівник випускної кваліфікаційної роботи

_____

(підпис, дата)

Відмітка про попередній захист _____

(ПІБ, підпис, дата)

12. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента _____Пономаренка С.В_____

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми _____Котенко Н.О._____

(прізвище, ініціали, підпис)

Завідувач кафедри _____Криворучко О. В._____

(підпис, прізвище, ініціали)

«_____»_____20 ___

АНОТАЦІЯ

Дана випускна кваліфікаційна робота присвячена розробці програмного додатка для контролю дотримання часового режиму руху громадського транспорту на платформі Android. Метою дослідження є поліпшення мобільності та зручності користувачів шляхом надання зручного інструменту для відстеження розкладу руху транспорту та уникнення затримок. В роботі розглядаються актуальність теми, об'єкт та предмет дослідження. Задачами дослідження є аналіз існуючих рішень, проектування та розробка програмного додатка, а також тестування та оцінка його функціональності. Методи дослідження включають аналітичний та експериментальний підходи. Впровадження розробленого додатка має велику практичну цінність, оскільки сприяє покращенню точності та своєчасності руху громадського транспорту, забезпечує зручність користувачів та покращує якість їхніх переміщень. Додаток може бути використаний операторами громадського транспорту для підвищення ефективності руху транспортних засобів та задоволення потреб пасажирів.

Випускна кваліфікаційна робота на тему «Програмний додаток контролю дотримання часового режиму руху громадського транспорту» містить 50 сторінок, 13 рисунків, 1 таблицю. Перелік використаних джерел налічує 11 найменувань.

Ключові слова: Мобільний додаток, контроль руху громадського транспорту, платформа Android, мобільність користувачів.

ABSTRACT

This graduation qualification project is dedicated to the development of a mobile application for monitoring compliance with the timetable of public transportation on the Android platform. The aim of the research is to enhance mobility and convenience for users by providing them with a user-friendly tool to track the schedule of public transportation and avoid unnecessary waiting times. The paper discusses the relevance of the topic, the object, and the subject of the research. The research tasks include analyzing existing solutions, designing and developing the mobile application, as well as testing and evaluating its functionality. The research methods encompass analytical and experimental approaches. The implementation of the developed application holds significant practical value as it contributes to improving the accuracy and timeliness of public transportation, enhances user convenience, and improves the quality of their commutes. The application can be utilized by public transportation operators to enhance the efficiency of transportation services and meet the needs of passengers.

Graduation qualification work on the topic "Software application for monitoring compliance with the time regime of public transport" contains 50 pages, 13 figures, 1 table. The list of used sources includes 11 items.

Keywords: Mobile application, public transport movement control, Android platform, user mobility improvement.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

MVC – Model-View-Controller, патерн проектування для розробки інтерфейсу користувача.

MVP – Model-View-Presenter, патерн проектування, що розділяє програму на три основні компоненти.

MVVM – Model-View-ViewModel, патерн проектування, який дозволяє відокремити логіку інтерфейсу користувача від бізнес-логіки.

UI – User Interface, інтерфейс користувача, частина програми, з якою взаємодіє користувач.

UX – User Experience, досвід користувача, загальне враження від взаємодії з програмою.

API – Application Programming Interface, набір визначень та протоколів для розробки та інтеграції програмного забезпечення.

JSON – JavaScript Object Notation, формат обміну даними легкий для людини для читання та написання.

XML – eXtensible Markup Language, мова розмітки, що використовується для зберігання та передачі даних.

HTTP – Hypertext Transfer Protocol, протокол для передачі гіпертекстових документів, таких як HTML.

HTTPS – Hypertext Transfer Protocol Secure, розширення HTTP, яке підтримує шифрування для безпечної передачі даних.

SDK – Software Development Kit, набір розробницьких інструментів для створення додатків.

IDE – Integrated Development Environment, інтегроване середовище розробки, що надає розробникам програмне забезпечення.

ЗМІСТ

ВСТУП **6**

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПРОБЛЕМАТИКИ У ВИБРАНІЙ ОБЛАСТІ **10**

1.1 Огляд сучасних підходів до розробки мобільних додатків на Kotlin 10

1.2. Опис структури та технологій розробки 13

1.3. Аналіз методів та засобів розробки 16

1.4 Висновки до розділу 1 17

РОЗДІЛ 2 КОНЦЕПЦІЯ ТА ОПИС ПРОГРАМНОГО ДОДАТКУ **19**

2.1. Загальний огляд програмного додатку 19

2.2. Оптимізація та ефективність програмного коду 21

2.3. Огляд існуючих аналогів серед мобільних додатків 24

2.3.1. EasyWay 24

2.3.2. D-transport 25

2.4. Висновки до розділу 2 27

РОЗДІЛ 3 ТЕХНІЧНІ ЗАСОБИ ТА ІНСТРУМЕНТИ РОЗРОБКИ **28**

3.1. Глибокий аналіз вибору Kotlin 28

3.2. Ретельне обґрунтування використання Figma для дизайну 29

3.3. Вибір платформи Android Studio та її роль у розробці 30

3.4. Переваги та виклики роботи з Android SDK 31

3.5. Висновки до розділу 3 32

РОЗДІЛ 4 ТЕХНІЧНА РЕАЛІЗАЦІЯ ТА РОЗРОБКА ДОДАТКУ **34**

4.1. Характеристика інформаційного забезпечення 34

4.2. Технічна реалізація 37

4.3. Розробка інтерфейсу користувача 39

4.4. Програмування та реалізація функціональності 43

4.5. Висновки до розділу 4 45

ВИСНОВКИ ТА ПРОПОЗИЦІЇ **47**

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **49**

ТЕХНІЧНЕ ЗАВДАННЯ **51**

ПРОГРАМА ТА МЕТОДИКА ТЕСТУВАННЯ 53
ДОДАТКИ……..55

ВСТУП

Актуальність. У сучасному світі розвиток громадського транспорту та забезпечення його ефективності та надійності стають все більш актуальними завданнями для міст і міських агломерацій. Зростаючі потреби мешканців у зручному та швидкому транспорті, а також необхідність зменшення автомобільного трафіку та впровадження екологічно чистих рішень, спонукають владу та транспортні компанії шукати інноваційні підходи до управління громадським транспортом.

Одним із ключових аспектів ефективного функціонування громадського транспорту є точне дотримання часового режиму руху транспортних засобів. Затримки у русі транспорту, непрогнозовані зміни розкладу, недостатній контроль руху - це проблеми, з якими зіштовхуються пасажири щоденно. Тому розробка програмного додатка, який забезпечуватиме точний контроль руху громадського транспорту та надаватиме пасажирам актуальну інформацію про розклад руху, стає дуже важливою.

Цей дослідницький проект спрямований на розробку та реалізацію мобільного додатка для платформи Android, який буде відповідати потребам ефективного контролю руху громадського транспорту. Основною метою проекту є створення зручного та надійного інструменту, який допоможе пасажирам отримувати актуальну інформацію про розклад руху, уникати затримок та непередбачуваних змін. Передові технології мобільних пристроїв та розвиток сучасних інтернет-технологій надають можливість створювати інноваційні рішення, які сприятимуть поліпшенню функціонування громадського транспорту та забезпечать зручність та задоволення пасажирів.

Мета роботи. Створення мобільного додатка для Android, який дозволить відстежувати графік громадського транспорту та надавати пасажирам легкий доступ до актуальних розкладів, з метою забезпечення точного виконання графіку руху.

В рамках цього дослідження будуть використані різноманітні методи дослідження, включаючи літературний аналіз, аналіз існуючих систем та розробок, аналіз даних руху громадського транспорту, проектування архітектури та функціональних вимог, розробку програмного забезпечення.

Виконання цього дослідницького проекту має наукову новизну, оскільки воно спрямоване на розробку нового мобільного додатка, який враховує сучасні тенденції розвитку технологій та потреби пасажирів. Впровадження такого додатка може мати значний вплив на ефективність та надійність громадського транспорту, поліпшити якість обслуговування пасажирів та зробити їх перебування у транспорті більш комфортним та зручним.

Отже, розвиток інноваційних підходів до управління громадським транспортом та забезпечення його ефективності є актуальним завданням сучасного світу. Представлений дослідницький проект спрямований на створення мобільного додатка, який дозволить пасажирам отримувати актуальну інформацію про розклад руху та зміни, забезпечуючи точний контроль руху громадського транспорту. Виконання проекту вимагатиме застосування різних методів дослідження та розробки, а результати дослідження можуть мати практичне значення для поліпшення транспортної системи та задоволення потреб пасажирів.

Об'єкт дослідження є програмний додаток для контролю дотримання часового руху громадського транспорут на платформі Android. Розробка застосонку спрямована на поліпшення якості послуг, наданих громадським транспортом, і забезпечення зручності та комфорту для пасажирів.

Предметом дослідження є розробка функціоналу, дизайну та архітектури мобільного додатка для контролю графіку громадського транспорту на платформі Android. Він буде забезпечувати функції контролю дотримання часового режиму руху громадського транспорту та надавати інформацію користувачам у зручній та доступній формі.

Предметна область даного дослідження охоплює розробку програмного додатка на платформі Android для контролю руху громадського транспорту та надання користувачам актуальної інформації про розклад руху, запізнення, зміни маршруту та розташування транспортних засобів у реальному часі. У процесі дослідження передбачається вирішення таких задач:

1. Аналіз сучасних підходів до управління рухом громадського транспорту та контролю часового режиму.
2. Розробка архітектури програмного додатка, що відповідає вимогам ефективного контролю руху транспорту.
3. Реалізація функцій відстеження розташування транспортних засобів, оновлення розкладу руху та сповіщення пасажирів про зміни.
4. Проведення тестування та апробація програмного додатка для перевірки його ефективності та коректності роботи.
5. Для досягнення мети дослідження будуть використані такі методи:
6. Аналіз літературних джерел та наукових публікацій з питань управління громадським транспортом і контролю руху.
7. Вивчення існуючих програмних рішень та додатків, що забезпечують контроль часового режиму руху транспорту.
8. Проектування архітектури програмного додатка на основі принципів моделі MVC (Model-View-Controller).
9. Розробка імплементації програмного додатка для платформи Android з використанням мови програмування Java та фреймворка Android SDK.

10. Проведення тестування програмного додатка та аналіз його ефективності та надійності.

Внаслідок дослідження очікується отримати наступні результати:

1. Розроблений програмний додаток, який забезпечує контроль дотримання часового режиму руху громадського транспорту та надає інформацію пасажирам.
2. Проведений аналіз сучасних підходів до управління рухом громадського транспорту та визначені переваги та недоліки існуючих систем.
3. Визначені ключові аспекти ефективного контролю руху транспорту та рекомендації щодо поліпшення функціонування громадського транспорту.

Отже, розробка програмного додатка для контролю часового режиму руху громадського транспорту на платформі Android має велике значення для забезпечення якості та ефективності громадського транспорту. Цей додаток сприятиме поліпшенню обслуговування пасажирів, зменшенню часу очікування та покращенню загальної якості громадського транспорту

РОЗДІЛ 1
ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПРОБЛЕМАТИКИ У ВИБРАНІЙ ОБЛАСТІ

1.1 Огляд сучасних підходів до розробки мобільних додатків на Kotlin

Історія та розвиток мови програмування Kotlin, мова програмування, яка швидко набула популярності серед розробників Android, була представлена компанією JetBrains у 2011 році[1, с. 21]. Це статично типізована мова, що поєднує в собі елементи об'єктно-орієнтованого та функціонального програмування. Kotlin був розроблений з метою забезпечення більшої продуктивності розробників, поліпшення читабельності коду та забезпечення сумісності з Java, яка на той час була основною мовою для розробки на Android. Завдяки своїм перевагам, Kotlin швидко здобув популярність у спільноті розробників. У 2017 році Google оголосила Kotlin офіційною мовою для розробки Android-додатків, що стало значним кроком у підтримці та популяризації мови. Kotlin вирішує багато проблем, з якими стикалися розробники при використанні Java, таких як більш строга обробка помилок в часі компіляції та коротший та більш читабельний код.

Однією з ключових особливостей Kotlin є його сумісність з Java. Це означає, що розробники можуть легко інтегрувати Kotlin у свої існуючі проекти на Java, а також використовувати численні Java бібліотеки та фреймворки. Така гнучкість дозволяє плавно переходити з Java на Kotlin, не відмовляючись від перевірених часом інструментів та бібліотек. У доповнення до вищезазначеного, Kotlin пропонує ряд удосконалень порівняно з Java, таких

як підтримка вищих порядкових функцій, лямбда-виразів, інлайн-функцій, та інших можливостей функціонального програмування. Ці властивості роблять Kotlin не тільки більш потужним, але й забезпечують більш високу експресивність коду.

Перехід на Kotlin відкриває нові горизонти для розробників Android-додатків, пропонуючи їм багато переваг порівняно з традиційним використанням Java. Однією з основних переваг Kotlin є його концизність. Kotlin дозволяє зменшити кількість стандартного коду, необхідного для виконання стандартних завдань, що спрощує читання та підтримку коду. Це особливо важливо у великих проектах, де керування кодовою базою може бути складним. Безпека типів є ще однією значною перевагою Kotlin. Система типів в Kotlin призначена для усунення помилок, таких як null pointer exceptions, які є досить поширеними у Java. Kotlin вводить концепцію nullable та non-nullable типів, що допомагає розробникам уникнути багатьох поширених помилок під час компіляції. Kotlin також включає в себе підтримку сучасних програмних парадигм, таких як функціональне програмування.

Функції вищого порядку, лямбда-вирази, і іммутабельність даних дозволяють розробникам писати код, який є не тільки більш коротким, але й більш читабельним та легким для підтримки. Ще однією ключовою перевагою Kotlin є його інтеграція з інструментами Android Studio. JetBrains, розробник Kotlin, також є автором IntelliJ IDEA, на базі якого створено Android Studio. Це забезпечує глибоку інтеграцію Kotlin з середовищем розробки Android, включаючи інструменти для рефакторингу, дебагінгу та тестування.

Розглядаючи наукові публікації та джерела, що стосуються Kotlin, можна знайти значний обсяг літератури, яка зосереджена на перевагах та викликах, що пов'язані з використанням цієї мови у розробці мобільних додатків. Багато досліджень відзначають, що Kotlin пропонує більшу ефективність розробки та ліпшу якість коду порівняно з Java. Наприклад,

дослідження показують, що Kotlin зменшує кількість коду, який потрібно писати, що призводить до зниження можливості помилок та спрощення процесу розробки. Одним з ключових аспектів, який часто аналізується, є безпека Kotlin. Завдяки строгій системі типів і властивостям, що допомагають уникнути різних помилок в часі компіляції (наприклад, null safety), Kotlin забезпечує вищий рівень безпеки додатків. Це особливо важливо в контексті мобільних додатків, де надійність та стабільність є критичними. Також існують дослідження, які зосереджуються на вивченні ефективності Kotlin у порівнянні з Java. Багато експертів підкреслюють, що Kotlin дозволяє розробникам використовувати більш декларативний підхід до програмування, що полегшує розуміння та підтримку коду. Крім того, Kotlin підтримує функціональні особливості які відсутні в Java, такі як лямбда-вирази, що робить Kotlin більш гнучким для розв'язання різних завдань.

Порівняння Kotlin з Java є ключовим для розуміння переваг Kotlin у розробці мобільних додатків. Хоча обидві мови мають свої сильні сторони, Kotlin пропонує ряд важливих удосконалень, які роблять його більш привабливим для сучасної розробки. Концизність та Читабельність Коду[2]:

1) Kotlin вимагає меншої кількості шаблонного коду порівняно з Java, що сприяє більшій концизності та легкості читання. Це особливо помітно у випадках, коли в Java необхідно написати багато коду для реалізації простих функцій.
2) Null Safety: Kotlin вводить систему контролю над null-значеннями на рівні мови, що знижує ризик NullPointerException, однієї з найбільш поширених помилок у Java. Kotlin дозволяє явно вказувати, може змінна бути null або ні, значно збільшуючи безпеку програми.
3) Функціональне Програмування: Kotlin активно інтегрує елементи функціонального програмування, такі як лямбда-вирази, що робить код більш експресивним та елегантним. Хоча Java 8 та новіші версії також

підтримують деякі функціональні особливості, Kotlin робить це більш зручним та інтуїтивним. Сумісність та

4) Інтеграція: Kotlin повністю сумісний з Java, дозволяючи використовувати Kotlin та Java в одному проекті. Це дає можливість поступового переходу на Kotlin без необхідності повного переписування існуючих Java-додатків.

1.2. Опис структури та технологій розробки

Правильний вибір архітектури додатка є ключовим для створення масштабованих, підтримуваних та ефективних мобільних додатків. Android-додатки часто використовують одну з наступних архітектурних патернів: Model-View-Controller (MVC): Цей патерн розділяє додаток на три основні компоненти: модель, що управляє даними додатка; вигляд, що представляє UI; та контролер, який забезпечує логіку обробки даних. Ознайомитись з візуалізацією можна переглянувши рис 1.1. Хоча MVC традиційно використовується у веб-розробці, він також може бути адаптований для мобільних додатків. Model-View-Presenter (MVP): У цій архітектурі, представлення (View) відповідає за отримання дій від користувачів, які потім передаються презентеру (Presenter) [3, с. 125].

Рис. 1.1. Ілюстрація MVP

Презентер обробляє ці дані, взаємодіючи з моделлю та оновлюючи представлення. MVP є популярним вибором для Android-додатків, оскільки він дозволяє легше тестувати та підтримувати код. Model-View-ViewModel (MVVM): Цей патерн є відносно новим і став популярним завдяки своїй ефективності у спрощенні роботи з UI. У MVVM, ViewModel відповідає за обробку логіки представлення та може реагувати на зміни у моделі, спрощуючи таким чином синхронізацію між UI та даними.

У розробці Android-додатків на Kotlin, важливо використовувати відповідні бібліотеки та фреймворки, щоб максимально ефективно вирішувати завдання. Ось декілька ключових бібліотек, які часто використовуються: Android Jetpack: Набір бібліотек та інструментів від Google, який спрощує розробку додатків. Він включає компоненти, такі як Room для роботи з базами даних, LiveData для реактивного програмування та ViewModel для реалізації архітектури MVVM. Retrofit: Популярна бібліотека для роботи з HTTP запитами та спілкування з веб-сервісами. Retrofit використовує анотації для опису запитів, що робить її інтуїтивно зрозумілою та легкою у використанні. Koin: Легковагий фреймворк для впровадження залежностей (dependency injection), спеціально розроблений для Kotlin. Koin пропонує простий синтаксис та легку інтеграцію, що робить його популярним вибором серед розробників Kotlin. Coroutines: Kotlin Coroutines дозволяють ефективно управляти асинхронними операціями, такими як мережеві запити чи довготривалі обчислення. Вони забезпечують більш чистий та зрозуміліший код порівняно з традиційними callback-ами. Використання цих бібліотек та фреймворків може значно покращити якість коду, спростити розробку та забезпечити високу продуктивність додатків.

Kotlin Coroutines є однією з ключових особливостей Kotlin, яка дозволяє зробити асинхронний код більш читабельним та ефективним. Coroutines спрощують управління асинхронними операціями, такими як мережеві запити,

обробка даних у фоновому режимі, та інші довготривалі операції. Ось декілька ключових аспектів використання coroutines:

Легкість інтеграції. Coroutines легко інтегруються з бібліотеками Android та Kotlin, що дозволяє ефективно використовувати їх у мобільній розробці.

Підвищення продуктивності. Завдяки меншому використанню ресурсів у порівнянні з традиційними потоками, coroutines сприяють підвищенню продуктивності додатків.

Покращення читабельності коду. Coroutines дозволяють писати асинхронний код, який виглядає як синхронний, роблячи код більш зрозумілим та легким для підтримки.

Використання Kotlin Coroutines є важливою частиною сучасної розробки Android-додатків, оскільки воно значно полегшує роботу з асинхронними операціями та покращує загальну архітектуру програм. Зверніть увагу на демонстрацію коду на рис 1.2.

```
1  import kotlinx.coroutines.*
2
3  fun main() = runBlocking {
4      launch {
5          delay(1000L)
6          println("Kotlin Coroutines!")
7      }
8      println("Hello,")
9  }
```

Рис. 1.2. Базовий приклад використання корутини

Забезпечення безпеки та ефективності є ключовими аспектами при розробці Android-додатків на Kotlin. Ось декілька кращих практик та методик оптимізації: ProGuard та R8: Використання інструментів мінімізації та обфускації коду, таких як ProGuard або R8, може значно зменшити розмір

кінцевого APK файлу та захистити код від реверс-інжинірингу. Використання Jetpack Security: Цей набір бібліотек допомагає захистити дані користувачів, забезпечуючи безпечне зберігання та передачу даних[4, с. 56].

Оптимізація продуктивності: підвищення продуктивності може бути досягнуто за допомогою оптимізації мережевих запитів, ефективного використання ресурсів (наприклад, зображень та анімацій), та оптимізації використання батареї. Застосування цих методик допоможе створити більш безпечні, швидкі та ефективні додатки.

1.3. Аналіз методів та засобів розробки

Знаєте, коли я думаю про алгоритми у контексті розробки Android-додатків на Kotlin, мені здається, що ми часто забуваємо про важливість правильного вибору алгоритмів. Так, Kotlin дозволяє писати елегантний і лаконічний код, але без розуміння того, як працюють алгоритми, ми можемо легко заблукати в джунглях оптимізації та ефективності. Я відчув, що найкращий спосіб покращити продуктивність додатків - це використання алгоритмів сортування, пошуку та структур даних, які оптимально відповідають вимогам конкретного додатка. Наприклад, вибір між LinkedList та ArrayList може суттєво вплинути на продуктивність додатка, особливо коли мова йде про обробку великих наборів даних.

Вивчаючи Kotlin, я особливо вражений можливостями Android Studio(рис 1.3). Це набагато більше, ніж просто IDE; це місце, де магія Kotlin оживає. З інструментами, такими як інтегровані тести та плагіни для дебагінгу, я відчув, що моя робота стає більш зосередженою на інноваціях, а не на рутинному кодуванні. Але не можна забувати і про системи управління версіями, як Git. Вони не тільки спрощують колаборацію та відстеження змін, але й дозволяють повернутися до попередніх версій коду, що є незамінним у великих проектах.

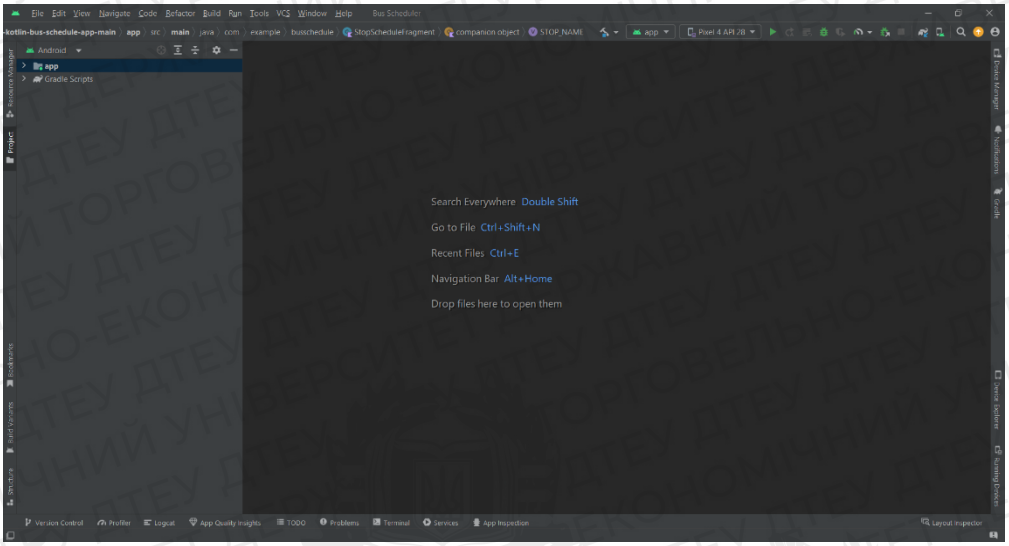

Рис. 1.3. Інтерфейс Android Studio

Як студент, який багато часу присвячує навчанню мобільної розробки, я відзначив, що одним з важливих аспектів є забезпечення сумісності додатків з різними версіями Android та апаратними характеристиками. З огляду на різноманітність пристроїв на ринку, це може бути справжнім викликом. Наприклад, оптимізація для пристроїв з обмеженою пам'яттю та процесорною потужністю вимагає глибокого розуміння того, як Kotlin взаємодіє з апаратним забезпеченням.

1.4 Висновки до розділу 1

Після детального вивчення та аналізу різних аспектів розробки Android-додатків на Kotlin, я можу з упевненістю сказати, що Kotlin відкриває нові горизонти для мобільних розробників. Він не просто вносить свіжість у процес програмування, але й надає потужні інструменти для створення якісних додатків. Спочатку, можливо, здавалося, що Kotlin - це просто "ще одна мова програмування", але зараз я бачу, що вона значно більше. Її ефективність, безпека та сумісність з Java роблять Kotlin ідеальним вибором для сучасної

мобільної розробки. Особливо вражають такі особливості, як coroutines для асинхронного програмування та інтеграція з Android Jetpack, які значно спрощують розробку складних додатків. З іншого боку, навіть з Kotlin важливо не забувати про основи алгоритмів та оптимізації. Вибір правильного алгоритму або структури даних може мати вирішальне значення для продуктивності та ефективності додатка. Також, забезпечення сумісності з різними пристроями та версіями Android залишається ключовим викликом. У підсумку, я відчуваю, що Kotlin значно розширює можливості розробників, дозволяючи їм фокусуватися на інноваціях та креативності. Це відкриває двері до створення додатків, які не просто виконують свої функції, але й приносять задоволення користувачам.

РОЗДІЛ 2

КОНЦЕПЦІЯ ТА ОПИС ПРОГРАМНОГО ДОДАТКУ

2.1. Загальний огляд програмного додатку

Місто Прилуки, хоч і не є великим містом України, має свої унікальні транспортні виклики. Жителі Прилук постійно стикаються з проблемою недостатності та ненадійності інформації про розклади громадського транспорту. Це особливо відчутно в пікові години, коли потреба у швидкому та ефективному пересуванні містом стає критичною. Насамперед, існує помітна відсутність централізованого джерела інформації, яке б надавало актуальний розклад автобусів. Інформація, яка існує, часто застаріла або розсіяна по різних неофіційних джерелах, що ускладнює пошук потрібних даних. Ця проблема стає ще більш актуальною для людей, які не є постійними мешканцями міста, але потребують користуватися міським транспортом. Відсутність надійної та оперативної інформації може призводити до затримок, пропущених автобусів та загалом знижує ефективність міського транспортного сервісу. В результаті, це впливає на загальну якість життя мешканців і сприяє накопиченню стресу та незадоволення серед громадян.

Враховуючи ці виклики, основною ідеєю додатку є створення надійного, зручного для користувача мобільного додатку, який буде надавати актуальну інформацію про розклади та маршрути автобусів у місті Прилуки. Основними цільовими аудиторіями є як постійні мешканці міста, так і відвідувачі або тимчасові жителі, які потребують надійного способу орієнтування у міському транспорті. Цей додаток не тільки значно спростить пошук інформації про

розклади автобусів, але й зможе надати користувачам можливість планувати свої подорожі більш ефективно, враховуючи часові рамки та доступні маршрути. Крім того, інтеграція з сучасними технологіями, такими як Google Maps SDK, дозволить візуалізувати маршрути автобусів на карті, надаючи користувачам інтуїтивно зрозумілий і зручний спосіб перегляду необхідної інформації.

Розроблений додаток, спочатку створений для міста Прилуки, має значний потенціал для розширення та адаптації в інших містах і регіонах, особливо в тих, де системи громадського транспорту ще не інтегровані з GPS-технологіями або іншими формами цифрового моніторингу. Розширення на Міста Без GPS-Моніторингу В багатьох містах, особливо в менших або в тих, що розвиваються, інфраструктура громадського транспорту часто не має інтеграції з сучасними GPS-системами. Це створює проблеми з точністю та своєчасністю інформації про розклади і маршрути. Наш додаток може бути адаптований для таких міст, пропонуючи простий, але ефективний спосіб отримання актуальної інформації про рух громадського транспорту.

Додаток може інтегруватися з існуючими базами даних розкладів, які використовуються міськими транспортними відомствами. Це дозволить додатку надавати користувачам найточнішу інформацію, доступну на даний момент, незалежно від наявності GPS-даних.

Для міст, де використання GPS у громадському транспорті неможливе або недоступне, додаток може використовувати інформацію, надану безпосередньо водіями автобусів або диспетчерами. Це може бути реалізовано через мобільний додаток або веб-інтерфейс, де водії можуть вносити зміни у розклади або інформувати про затримки в реальному часі.

Ключовим аспектом розширення є гнучкість і адаптивність додатку до різних типів міських транспортних систем. Це означає, що додаток повинен

бути спроектований з урахуванням різноманітності транспортних засобів, маршрутів і специфіки їх руху в різних містах.

На більш широкому рівні, додаток може бути розширений для включення інформації не тільки про автобуси, але й про інші види громадського транспорту, такі як трамваї, тролейбуси або метро. Це зробить додаток універсальним інструментом для планування подорожей в будь-якому місті.

Ще одним напрямком розширення може бути створення спільноти користувачів, які можуть вносити свій вклад у оновлення та підтримку інформації про розклади та маршрути. Це створить динамічну та взаємодопомагаючу атмосферу, де інформація підтримується актуальною завдяки зусиллям самих користувачів.

2.2. Оптимізація та ефективність програмного коду

Отже, ми вже знаємо, що Android Studio - це не просто місце, де ваш код перетворюється в додаток. Це також місце, де ви можете поліпшити і оптимізувати цей код. Тут ми розглянемо, як Android Studio допомагає розробникам писати ефективніший і оптимізованіший код. Підвищення Продуктивності Розробки Android Studio має цілу купу інструментів, які допомагають вам писати менше коду, але робити більше. Це може звучати як чарівництво, але насправді це просто добре продумане програмне забезпечення. Наприклад, функція автозаповнення коду значно спрощує процес написання коду, а шаблони та генератори коду допомагають швидко створювати загальні елементи додатків.

Android Studio містить потужні інструменти для дебагінгу, які допомагають вам швидко знаходити і виправляти помилки. Це дуже важливо, адже помилки в програмуванні - це неминуче, але важливо вміти з ними ефективно справлятися. Профілювання Додатків Android Studio також надає

розширені можливості для профілювання додатків. Це означає, що ви можете аналізувати, як ваш додаток використовує ресурси, такі як CPU, пам'ять та батарею. Завдяки цьому, ви можете оптимізувати ваш додаток, щоб він споживав менше ресурсів і працював ефективніше. Інтеграція з іншими інструментами Ще однією ключовою перевагою Android Studio є його інтеграція з іншими інструментами та сервісами, такими як системи управління версіями (наприклад, Git) [5, с. 205]. Це дозволяє вам більш ефективно управляти змінами в коді та співпрацювати з іншими розробниками. Заключний Висновок Android Studio - це не просто місце для написання коду. Це комплексний інструмент, що допомагає розробникам не тільки писати код, але й забезпечувати його якість, продуктивність та ефективність. Використання цих інструментів може значно поліпшити якість вашого додатку та зробити процес розробки більш приємним та менш стресовим.

Перший етап будь-якого проекту з розробки додатків - це концептуалізація. Android Studio допомагає в цьому завдяки інструментам для макетування та візуалізації. Ви можете легко експериментувати з різними дизайнами та інтерфейсами, не пишучи жодного рядка коду. Це дозволяє вам більш чітко уявити, як буде виглядати кінцевий продукт.

Завдяки інтеграції з Kotlin та Java, ви маєте величезний вибір інструментів та бібліотек для реалізації вашого проекту. Функція автозаповнення та інтелектуальні підказки значно полегшують процес кодування, дозволяючи вам зосередитись на логіці та архітектурі додатку, а не на синтаксисі.

Однією з ключових переваг Android Studio є вбудовані інструменти для тестування. Ви можете використовувати юніт-тести для перевірки окремих компонентів вашого додатку, або інтеграційні тести, щоб переконатися, що всі частини додатку працюють разом, як очікується.

Неминуче, у процесі розробки з'являються помилки. Android Studio пропонує розширені можливості для дебагінгу, дозволяючи вам швидко знаходити та виправляти помилки. Інструменти, такі як Android Profiler, дозволяють детально аналізувати виконання вашого додатку, щоб виявити причини проблем.

Нарешті, коли ваш додаток готовий, Android Studio спрощує процес його випуску. Ви можете легко підготувати ваш додаток до розгортання, використовуючи інструменти для збірки та пакування, а також тестувати його в реальних умовах, використовуючи Google Play Beta.

Процес розробки в Android Studio охоплює весь цикл створення додатку - від перших ідей до випуску на ринок. Це повнофункціональне середовище розробки, що підтримує розробників на кожному етапі цього процесу.

У сучасному світі розробки мобільних додатків адаптивність та масштабування є ключовими. Додатки повинні ефективно працювати на широкому спектрі пристроїв з різними розмірами екранів та характеристиками. Android Studio забезпечує інструменти, які допомагають розробникам створювати гнучкі та адаптивні інтерфейси. Завдяки цьому, ваш додаток зможе забезпечити однаково високий рівень користувацького досвіду незалежно від пристрою, на якому він використовується.

Важливим аспектом розвитку додатків є їхнє оновлення та підтримка. Завдяки Android Studio, розробники можуть легко вносити зміни та оновлювати додатки, випускаючи оновлення, які покращують функціональність, виправляють помилки або додають нові функції. Це гарантує, що додаток залишається актуальним та конкурентоспроможним на ринку.

Ще одним аспектом розвитку додатків є збір та аналіз користувацьких даних. Android Studio дозволяє інтегрувати різні аналітичні інструменти, які

допомагають розробникам зрозуміти поведінку користувачів, оптимізувати додаток та покращувати користувацький досвід.

Розміщення додатку в Google Play Store є важливим етапом у життєвому циклі будь-якого додатку. Це не просто процес публікації; це також про забезпечення відповідності додатку стандартам та вимогам Google. Це включає в себе дотримання політики конфіденційності, правил щодо контенту та технічних вимог. Знання та розуміння цих положень є критично важливим для успішного випуску та підтримки додатку. Це також важливо для уникнення випадків відмови в публікації або видалення додатку з магазину.

У цьому розділі ми розглянули ключові елементи, які впливають на розробку, підтримку та успішність мобільних додатків. Від вибору інструментів та платформ, таких як Kotlin, Figma, Android Studio та Android SDK, до розуміння політики публікації в Google Play, кожен аспект відіграє важливу роль у створенні успішного додатку. Розуміння цих аспектів є ключем до створення високоякісних, функціональних та популярних мобільних додатків.[6]

2.3. Огляд існуючих аналогів серед мобільних додатків

У процесі розробки мобільного додатку для міста Прилуки важливо розглянути існуючі аналоги на ринку України, щоб зрозуміти їхні сильні та слабкі сторони. Це допоможе визначити, які аспекти потрібно покращити або розвинути у власному додатку.

2.3.1. EasyWay

EasyWay відомий серед українських користувачів як один із найбільш популярних додатків для моніторингу громадського транспорту(демонстрація інтерфейсу на рис 2.1). Цей додаток вирізняється широким охопленням міст та населених пунктів, забезпечуючи інформацію про розклади різних видів транспорту. Основні переваги EasyWay полягають у його функціональності:

додаток надає користувачам детальну інформацію про маршрути, часи відправлення, а також актуальні зміни у розкладах.

Незважаючи на свою популярність і функціональність, EasyWay має деякі недоліки. Одним з них є дизайн інтерфейсу користувача, який може здатися застарілим або перевантаженим для деяких користувачів. Це може ускладнити навігацію по додатку, особливо для нових користувачів.

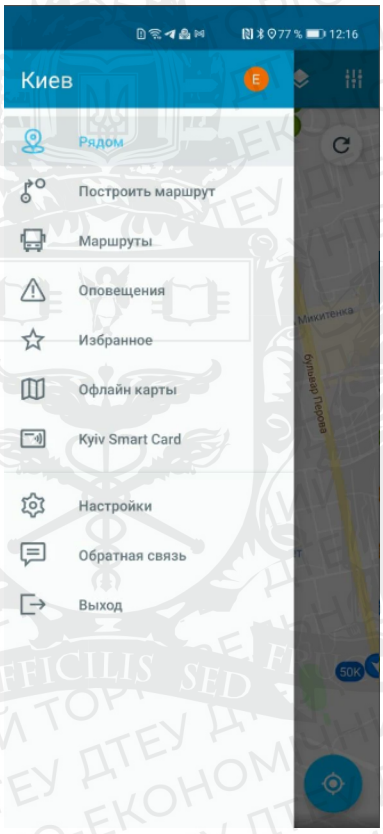

Рис. 2.1 Інтерфейс додатку EasyWay [7]

2.3.2. D-transport

D-transport, хоча і менш відомий у порівнянні з EasyWay, пропонує альтернативний підхід до моніторингу громадського транспорту. Основна його перевага - це спрощений інтерфейс користувача, який зосереджений на наданні базової, але важливої інформації про розклади та маршрути(

демонстрація інтерфейсу на рис 2.2). Це робить D-transport зручним варіантом для користувачів, які вважають за краще більш простий та інтуїтивно зрозумілий дизайн.

Однак, через свою обмежену популярність, D-transport може мати менше охоплення міст та маршрутів порівняно з EasyWay. Також, можливості GPS-моніторингу та інші додаткові функції можуть бути обмежені або відсутні.

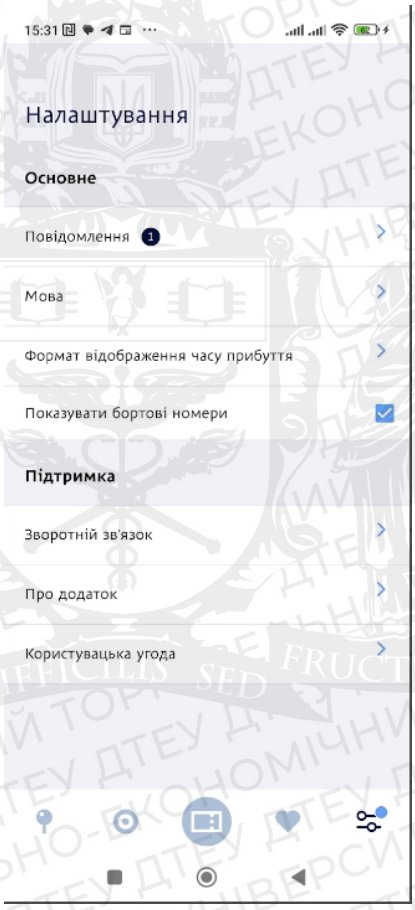

Рис. 2.2 Інтерфейс додатку D-transport [8]

Розробляючи додаток для міста Прилуки, важливо взяти до уваги сильні та слабкі сторони цих існуючих рішень. Наш додаток прагне поєднати найкращі аспекти обох цих додатків: забезпечити широке охоплення маршрутів та детальну інформацію, як у EasyWay, але з більш сучасним та користувацьки дружнім інтерфейсом, схожим на D-transport. Особливу увагу буде приділено дизайну інтерфейсу, щоб зробити його якомога більш

інтуїтивно зрозумілим і легким у використанні. Також планується інтеграція з GPS-моніторингом (де це можливо) та використання сучасних технологій для забезпечення надійності та актуальності наданої інформації.

2.4. Висновки до розділу 2

Після ретельного аналізу існуючих аналогів мобільних додатків для моніторингу громадського транспорту, як EasyWay та D-transport, я здобув цінні вказівки для розробки додатку для міста Прилуки. Функціональність та охоплення, що демонструє EasyWay, є ключовими для створення універсального рішення, яке забезпечує користувачів детальною інформацією. Водночас, важливість простого і інтуїтивно зрозумілого інтерфейсу, як у D-transport, підкреслює потребу у виваженому підході до дизайну інтерфейсу користувача в нашому додатку. Розглядаючи гнучкість та адаптивність, наш додаток має бути спроектований таким чином, щоб він був зручним у використанні у різних умовах, зокрема в містах, де відсутній GPS-моніторинг громадського транспорту. Використання сучасних технологічних інновацій, таких як GPS та інтелектуальні системи обробки даних, дозволить підвищити точність та актуальність інформації, що надається користувачам. Наш додаток також має великий потенціал для розвитку. Окрім розширення охоплення на інші міста та регіони, залучення спільноти користувачів для внесення оновлень і покращень може значно збільшити його цінність та користь для кінцевих користувачів. Це створює можливості не лише для розвитку додатку, але й для побудови активної та залученої спільноти користувачів. Отже, у процесі розробки нашого додатку ми прагнемо до створення продукту, який поєднує в собі передові технологічні рішення, високий рівень користувацького досвіду та гнучкість у застосуванні для різних міських умов.

РОЗДІЛ 3

ТЕХНІЧНІ ЗАСОБИ ТА ІНСТРУМЕНТИ РОЗРОБКИ

3.1. Глибокий аналіз вибору Kotlin

Kotlin знижує кількість шаблонного коду, який вам доводиться писати. Це не лише економить час, але й робить код більш читабельним та легшим для підтримки. Наприклад, розгляньте випадки, коли вам потрібно створити модельні класи для вашого додатку. У Kotlin, завдяки особливості data classes, ви можете зменшити кількість коду, який потрібен для опису простого класу з кількома полями. Це не тільки полегшує написання, але й робить ваші моделі більш читабельними.

Ще одним крутим бонусом Kotlin є його вбудована нуль-безпечність. Kotlin дозволяє розробникам бути впевненими, що їхні програми менше схильні до помилок NullPointerException, які часто зустрічаються в Java. Це реалізовано через систему типів, яка чітко розрізняє нульові та ненульові типи.

Так, ваш перший підрозділ уже охопив цю тему, але варто підкреслити, що сумісність Kotlin з Java є важливою перевагою, особливо коли ви працюєте над великими проектами або інтегруєте Kotlin у вже існуючий проект на Java. Це робить перехід на Kotlin менш болючим та дозволяє використовувати вже існуючі Java бібліотеки. Ознайомитися з порівнянням Kotlin з Java можна за таблицею 3.1.

Таблиця 3.1

Порівння Kotlin і Java

Критерій	Kotlin	Java
Кількість шаблонного коду	Значно менша завдяки функціям, я к data classes.	Вимагає більше шаблонного коду для подібних завдань.
Читабельність коду	Код більш читабельний та легший для підтримки.	Може бути менш читабельним через більшу кількість коду.
Нуль-безпечність	Має вбудовану нуль-безпечність, зменшує помилки NullPointerException.	Частіші помилки NullPointerException.
Сумісність з Java	Висока сумісність, можливість використання Java бібліотек.	Не застосовується, оскільки Java є мовою.

3.2. Обґрунтування використання Figma для дизайну

Figma – це хмарний інструмент для дизайну інтерфейсів, який значно спрощує процес створення та колаборації в дизайні(рис 3.1). Це не просто програма для малювання; це цілий екосистема, яка об'єднує дизайнерів, розробників та інших зацікавлених сторін в єдиний творчий процес. Замість того, щоб відправляти файл за файлом або перевіряти декілька версій одного дизайну, Figma дозволяє командам працювати разом в одному файлі в реальному часі.

Це надзвичайно корисно в умовах швидкого темпу роботи, де потрібно оперативно вносити зміни та отримувати зворотний зв'язок. Інтуїтивний Інтерфейс Для студентів, які тільки починають свій шлях у дизайні, інтерфейс Figma може здатися дещо складним на перший погляд. Але вже через короткий час використання стає зрозуміло, що все продумано до дрібниць. Всі інструменти розміщені логічно і зручно, а процес роботи з компонентами, шарами, та стилями є надзвичайно гнучким.

Не можна ігнорувати можливості Figma в області прототипування. Ви можете легко перетворити ваші дизайни в інтерактивні прототипи, додаючи

переходи, анімації, та інтерактивні елементи. Це дозволяє дизайнерам і розробникам краще розуміти, як продукт буде виглядати і функціонувати на реальних пристроях.

Ще одна перевага Figma - це величезна кількість доступних шаблонів і бібліотек. Ви можете використовувати існуючі бібліотеки компонентів та іконок, що значно спрощує процес створення дизайну. Це також допомагає утримувати єдність стилю в усьому проекті, оскільки всі елементи дизайну будуть послідовно використовувати однакові компоненти.

Figma дозволяє не тільки створювати дизайни, але й оптимізувати весь процес роботи. Ви можете інтегрувати Figma з іншими інструментами, такими як Slack для комунікації, або Jira для управління проектами. Це створює зручне середовище, де все, що вам потрібно для роботи, знаходиться в одному місці.

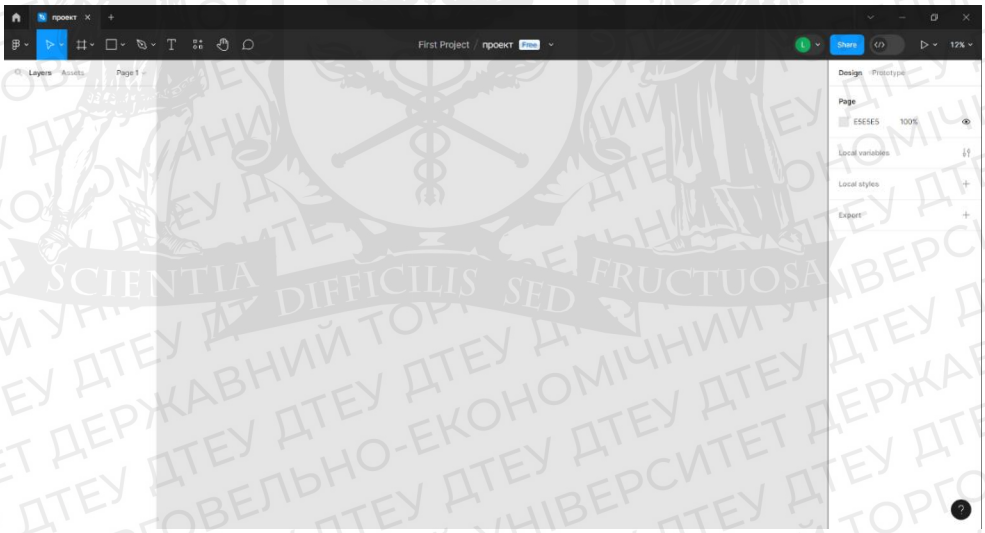

Рис. 3.1 Інтерфейс додатку Figma

3.3. Вибір платформи Android Studio та її роль у розробці

Спочатку, давайте розглянемо, чому Android Studio стало таким популярним вибором серед розробників Android-додатків. Android Studio не просто ще одне середовище розробки; це місце, де ваші ідеї перетворюються на щось справді круте. З його інтуїтивно зрозумілим інтерфейсом, це місце, де

код стає більш ніж просто текстом — він оживає. Android Studio пропонує розширений набір функцій, включаючи автоматичне виявлення помилок, інтеграцію з системами управління версіями, підтримку Kotlin та Java, а також потужні інструменти для візуалізації та тестування інтерфейсу користувача.

Інтеграція з Kotlin і Java: це одна з основних причин, чому Android Studio так швидко стало популярним. Воно надає чудову підтримку Kotlin, яка, як ми вже знаємо, стає все більш важливою для розробників Android. Емулятори та реальні пристрої: Android Studio має вбудовані емулятори, які дозволяють вам тестувати додатки в різноманітних умовах без необхідності мати фізичні пристрої. Це зекономлює час і ресурси, особливо коли вам потрібно тестувати додаток на багатьох різних пристроях.

Android Studio має досить потужні інструменти для дебагінгу та профілювання, які допомагають вам знайти та виправити помилки в коді. Це робить процес розробки більш ефективним та менш стресовим.

Android Studio надає розробникам зручні інструменти для візуалізації інтерфейсу користувача, дозволяючи легко переглядати та модифікувати лейаути, анімації та інші елементи інтерфейсу.

3.4. Переваги та виклики роботи з Android SDK

Android Software Development Kit (SDK) є невід'ємною частиною розробки додатків для Android. Цей набір інструментів надає розробникам доступ до API, які потрібні для використання функціональності пристроїв Android. Ось декілька ключових переваг Android SDK:

Android SDK включає інструменти, бібліотеки та API, які дозволяють розробникам використовувати всі можливості Android-пристроїв, від камери до акселерометра. [9, с. 128].

Google регулярно оновлює Android SDK, додаючи нові функції та вдосконалення, що дозволяє розробникам використовувати останні інновації в своїх додатках.

Android SDK тісно інтегрований з Android Studio, що полегшує процес розробки та тестування додатків.

Однією з основних проблем при розробці для Android є необхідність забезпечити сумісність додатку з різними версіями операційної системи, які все ще активно використовуються.

Android працює на величезній кількості пристроїв з різними характеристиками, від екранів різних розмірів до різних типів апаратного забезпечення. Це створює додаткові виклики для розробників, які прагнуть забезпечити стабільну роботу додатків на всіх цих пристроях.

Забезпечення високої якості та стабільної роботи додатку на різноманітних пристроях вимагає ретельного тестування та часто може бути часомістким.

Android SDK є потужним інструментом у руках розробника, але як і будь-який інструмент, він має свої специфічні виклики. Розуміння та ефективне вирішення цих викликів є ключовим для створення якісних та успішних Android-додатків.

3.5. Висновки до розділу 3

У цьому розділі ми розглянули ключові технічні інструменти, які є важливими для розробки сучасних програмних продуктів: Kotlin, Figma, Android Studio, та Android SDK. Кожен з цих інструментів має свої унікальні переваги та виклики.

Kotlin: Ця мова програмування вирізняється своєю зручністю, скороченням шаблонного коду, нуль-безпечністю, та чудовою сумісністю з

Java. Це робить Kotlin ідеальним вибором для Android розробки, підвищуючи продуктивність та безпеку коду.

Figma: Як хмарний інструмент для дизайну інтерфейсів, Figma сприяє покращенню співпраці та ефективності роботи дизайнерів. Його інтуїтивний інтерфейс, гнучкість, і широкий спектр інтеграцій роблять його незамінним інструментом у сфері UI/UX дизайну.

Android Studio: Це основне середовище розробки для Android, яке надає потужні функції для ефективної розробки, тестування, дебагінгу та візуалізації інтерфейсу користувача. Його інтеграція з Kotlin та Java, а також вбудовані інструменти для роботи з емуляторами та реальними пристроями, роблять його важливим ресурсом для розробників.

Android SDK: Цей набір інструментів є критично важливим для розробки на платформі Android, надаючи доступ до необхідних API та функцій пристроїв. Однак, виклики, пов'язані з сумісністю та тестуванням на різноманітних пристроях, вимагають уваги та ретельного підходу від розробників. У сукупності, правильне використання та поєднання цих інструментів може значно поліпшити процес розробки, підвищити якість кінцевого продукту та забезпечити гладке взаємодію між різними аспектами розробки - від написання коду до дизайну інтерфейсу.

РОЗДІЛ 4

ТЕХНІЧНА РЕАЛІЗАЦІЯ ТА РОЗРОБКА ДОДАТКУ

4.1. Характеристика інформаційного забезпечення

Розробка додатку для надання інформації про розклад автобусів у місті Прилуки починається з визначення ключових джерел даних. Ці джерела включають:

- міські транспортні відомства, як первинні постачальники інформації, ці установи надають офіційні дані про розклади, маршрути та будь-які затримки або зміни у русі транспортних засобів.
- онлайн-платформи громадського транспорту. Ці ресурси часто використовуються для отримання інформації про розклади та маршрути в режимі реального часу.
- внески від користувачів. Іноді інформацію можуть надавати самі користувачі, наприклад, через функцію сповіщення про затримки або зміни в руху транспорту.

Ці джерела даних відіграють критичну роль у забезпеченні додатку актуальною та точною інформацією. Важливо не тільки зібрати дані, але й правильно їх обробити та представити користувачам.

Для ефективного збору даних необхідно розробити систему, яка може автоматично збирати інформацію з різних джерел. Це включає:

- інтеграцію з API: Використання API (Application Programming Interface) від різних онлайн-сервісів та міських транспортних відомств для автоматичного збору даних.

- використання Веб-Скрапінгу: У деяких випадках, де API не доступний, може бути застосований веб-скрапінг для збору інформації з веб-сайтів транспортних відомств.

Обробка зібраних даних включає їх аналіз, класифікацію та валідацію. Важливо врахувати наступні аспекти:

- фільтрація: видалення нерелевантної або застарілої інформації зі зібраних даних.
- валідація: перевірка точності та актуальності інформації, зіставлення даних із різних джерел.
- форматування: приведення даних до єдиного формату для легкого доступу та відображення у додатку.
- після обробки, інформація повинна бути представлена користувачам у зручному та інтуїтивно зрозумілому форматі. Це може включати:
- інтерактивні розклади: забезпечення користувачам можливості переглядати розклади в динамічному інтерфейсі.
- інтеграція з картами для візуалізації маршрутів та зупинок. Оновлення інформації має відбуватися в реальному часі або з мінімальною затримкою, щоб гарантувати, що користувачі отримують найактуальнішу інформацію.

Ефективна обробка та представлення даних є ключовим фактором успіху додатку. Це не тільки підвищує довіру користувачів до додатку, але й значно покращує їх досвід використання. Використання передових технологій для збору та обробки даних є фундаментом для створення високоякісного і корисного додатку для жителів міста Прилуки.

Забезпечення Актуальності та Надійності Інформації:

Одним з ключових факторів, який забезпечує надійність інформації у додатку, є її постійне оновлення. Це вимагає встановлення чіткого графіка оновлень, щоб інформація завжди відображала найсвіжіші дані про розклади

автобусів. Оновлення можуть відбуватися щодня або навіть кілька разів на день, в залежності від доступності інформації та її важливості.

Відсутність автоматизованого виявлення змін вимагає активної взаємодії з джерелами інформації та ретельної перевірки наданої інформації. Запроваджується система перевірок і валідації, що включає ручну перевірку інформації, зіставлення даних між різними джерелами та використання історичних даних для порівняння і виявлення можливих невідповідностей.

Залучення користувачів до процесу перевірки інформації є важливою частиною підтримки її надійності. Користувачі можуть надавати зворотний зв'язок про точність розкладів та повідомляти про зміни або затримки, які вони спостерігають на місцях.

Встановлення прямих каналів зв'язку з міськими транспортними відомствами для отримання найактуальніших даних. Регулярні зустрічі та обговорення з відповідальними особами допоможуть забезпечити своєчасність та точність інформації.

Розробка чітких процедур для перевірки та валідації інформації перед її публікацією у додатку. Це включає перевірку на відповідність між різними джерелами, а також використання історичних даних для виявлення невідповідностей.

Хоча автоматизоване виявлення змін не входить у план розробки, використання інших технологій, таких як бази даних та системи управління даними, допоможе в ефективному управлінні та організації інформації.

Якість інформації, яка надається у додатку, безпосередньо впливає на задоволеність та довіру користувачів. Надійна і точна інформація не тільки забезпечує користувачам цінний ресурс для планування їхніх поїздок, але й підвищує репутацію додатку як надійного інструменту. Встановлення ефективних процедур для забезпечення надійності та достовірності інформації є ключовим аспектом розробки.

4.2. Технічна реалізація

Я вирішив використовувати корутини Kotlin, і вони стали справжнім порятунком. Це дозволило моєму додатку швидко реагувати, навіть коли відбувається багато обробки даних. Таке рішення дійсно показало, наскільки потужною може бути проста зміна в підході до асинхронного програмування.

Використання розширення функцій у Kotlin дозволило мені значно спростити код. Це як магія — один маленький шматок коду може зробити чудеса для читабельності і організації загалом.

Я був вражений, наскільки інтуїтивно можна створювати інтерфейси з Layout Editor. Це справді допомогло мені візуалізувати, як мій додаток виглядатиме і працюватиме, навіть до того, як я написав рядок коду.

Використання Android Profiler було ключовим у забезпеченні гладкої роботи мого додатку. Це дозволило мені побачити, де мій додаток споживав забагато ресурсів, і виправити ці моменти.

Замість підключення до сервера, я вирішив використовувати локальну базу даних для зберігання інформації. Це рішення було зумовлено кількома факторами:

- надійність: локальна база даних забезпечує доступ до інформації навіть при відсутності Інтернету.
- простота розробки: мені здалося, що управління локальними даними буде простіше і менш витратно з точки зору часу та ресурсів.

Це дозволило мені зосередитися на інших аспектах додатку, не турбуючись про додаткові складнощі, пов'язані з мережевими запитами і синхронізацією даних.

Для локального зберігання даних я обрав Room Database, частина Android Jetpack. Room забезпечує абстракцію рівня над SQLite, що дозволяє краще управляти базою даних, забезпечуючи при цьому повну сумісність із SQLite[10, с. 210]. .

Я створив таблиці для зберігання розкладів автобусів та їх маршрутів. Кожен запис містить інформацію про часи відправлення, номери маршрутів та зупинки.

Також було важливо мати таблицю для зберігання індивідуальних налаштувань користувачів, яка дозволяє зберігати вибрані маршрути та налаштування сповіщень.

Для кожного маршруту та зупинки я використав унікальні ідентифікатори, що спрощує пошук і зв'язування даних між різними таблицями.

DAO дозволяє абстрагуватися від конкретних SQL запитів і працювати з об'єктами бази даних більш високого рівня.

Хоча зв'язок з сервером міг би забезпечити динамічне оновлення даних, я вирішив зосередитися на локальному зберіганні. Це рішення базувалося на кількох факторах:

- надійність: локальна база даних гарантує доступність інформації навіть без Інтернет-з'єднання.
- простота розробки: управління локальними даними здається мені більш прямолінійним і не вимагає додаткових зусиль для синхронізації з сервером.

Такий підхід дозволяє мені зосередитися на стабільності та швидкості додатку, гарантуючи при цьому, що користувачі завжди матимуть доступ до необхідної інформації.

При розробці додатку, важливим аспектом було створення ефективної структури даних, що дозволяє легко організовувати, зберігати та доступати до інформації про розклади та маршрути. Я вирішив використовувати комбінацію простих та зрозумілих методів кодування, які спрощують роботу з даними.

Створено окремі таблиці для маршрутів і розкладів. Кожен маршрут має унікальний ідентифікатор, що зв'язує його з відповідними записами в таблиці розкладів.

Для оптимізації швидкості пошуку і доступу до даних використано індексацію важливих полів, таких як номери маршрутів і часи відправлення.

Кожному маршруту присвоєно унікальний ідентифікатор, що спрощує процес пошуку та зв'язування даних у різних частинах додатку.

Для уникнення дублікації та полегшення управління даними застосовано принципи нормалізації. Це забезпечує, що кожен елемент даних зберігається тільки в одному місці, зменшуючи ризик непослідовності.

Для імпорту та експорту даних використовується формат JSON, оскільки він легкий, гнучкий та легко читається як людьми, так і машинами.

Для обробки JSON даних використовуються вбудовані інструменти Kotlin, що дозволяють ефективно перетворювати JSON у внутрішні структури даних. [11, с. 169].

4.3. Розробка інтерфейсу користувача

Візуалізація.

При розробці інтерфейсу користувача для мого додатку, я вирішив використовувати Figma як основний інструмент для дизайну. Це дозволило мені ефективно візуалізувати і тестувати різні концепції дизайну перш ніж переносити їх у реальний код.

Головний екран додатку орієнтований на простоту та інтуїтивність. Я вирішив використовувати чистий і мінімалістичний дизайн, щоб уникнути перевантаження інформацією. Великі кнопки та чіткі шрифти допомагають легко орієнтуватися по додатку. Ознайомитися з головним екраном рис 4.1.

Рис. 4.1. Головний екран додатку

Екран пошуку спроектовано так, щоб користувачі могли легко і швидко знаходити необхідні маршрути. Я використав зручну панель пошуку з функцією автозаповнення, що базується на найпопулярніших запитах. Ознайомитися з екрано пошуку рис 4.2.

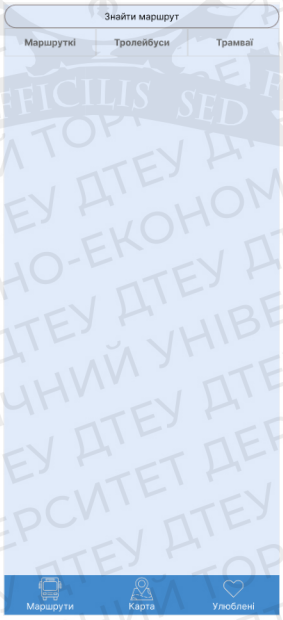

Рис. 4.2. Екран вибору та пошуку маршрутів

Результати пошуку представлені у вигляді легко читабельного списку, з виділенням найважливішої інформації, такої як час відправлення та номер маршруту(рис 4.3).

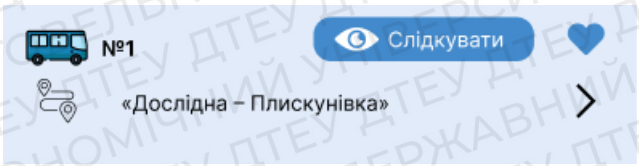

Рис. 4.3. Екран вибору громадського транпорту

Я включив нижню навігаційну панель для швидкого доступу до основних функцій додатку, таких як домашня сторінка, розклади та налаштування(рис 4.4).

Рис. 4.4. Нижня панель взаємодії

Вспливаючі Вікна для Деталей Маршруту: Для додаткової інформації про кожен маршрут я створив вспливаючі вікна, що активуються при натисканні на конкретний маршрут у списку(рис 4.5).

Рис. 4.5. Екран розкладу руху громадського транспорту

Одним з моїх пріоритетів було забезпечити, щоб дизайн добре виглядав на всіх типах пристроїв, від маленьких смартфонів до великих планшетів.

У процесі роботи над дизайном у Figma, я постійно тестував різні варіанти та вносив корективи, виходячи з фідбеку тестувальників. Це було чудовим досвідом, що дозволив мені глибше зрозуміти, як візуальні елементи та навігація впливають на користувацький досвід. Для цього розділу можна включити скріншоти з Figma, щоб продемонструвати ключові моменти дизайну.

Опис основних функціональних частин інтерфейсу.

Продовжуючи розробку інтерфейсу користувача, я детально відпрацював ключові функціональні частини додатку, використовуючи макети з Figma як основу для візуалізації та реалізації в коді.

1. Головний екран служить вхідною точкою в додаток і надає користувачам швидкий доступ до основних функцій, таких як перегляд розкладу, пошук маршрутів та доступ до налаштувань. Використовуючи принципи чистого та інтуїтивного дизайну, я розмістив ключові елементи навігації та функціональності на головному екрані, забезпечуючи легкий доступ і зрозумілість для користувачів.
2. На екрані пошуку користувачі можуть шукати маршрути за номером або назвою зупинки. Я використав зручну пошукову панель з можливістю автозаповнення для полегшення процесу пошуку.Результати пошуку відображаються у вигляді списку, де кожен маршрут представлений з ключовою інформацією, такою як час відправлення та проміжні зупинки.
3. На екрані деталей маршруту користувачі можуть отримати детальну інформацію про конкретний маршрут, включно з повним переліком зупинок та графіком руху. Для кращого досвіду користувачів, я включив інтерактивні елементи, такі як карта маршруту та опції нагадувань про прибуття транспорту.

4. Налаштування. Цей розділ дає користувачам можливість налаштувати додаток під свої переваги, включаючи сповіщення, мовні налаштування та інші опції користувацького інтерфейсу(рис 4.6 та рис 4.7).

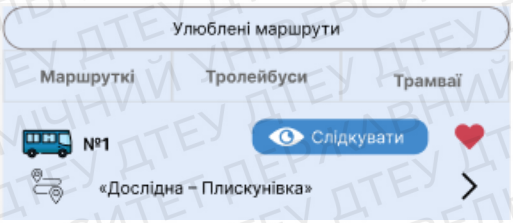

Рис. 4.6. Екран улюблених маршрутів

Рис. 4.7. Сигнал повідомлення

4.4. Програмування та реалізація функціональності

Важливі фрагменти коду з поясненнями

Під час розробки додатку я зосередився на створенні чистого, ефективного та легко читабельного коду. Ось деякі ключові фрагменти коду, які були важливими в процесі реалізації функціональності додатку.

1. Головна активність (MainActivity): mainActivity є вхідною точкою в додаток. Тут я визначив основну логіку навігації та ініціалізацію UI елементів (додаток Б.1).

2. Адаптер для списку маршрутів: для відображення списку маршрутів я створив кастомний адаптер, що дозволяє ефективно управляти даними в списку (додаток Б.2).

3. Обробка пошуку маршрутів: функціональність пошуку була реалізована за допомогою простих, але ефективних алгоритмів (додаток Б.3).

4. Функція Запам'ятовування улюблених маршрутів: ця функція дозволяє користувачам зберігати свої улюблені маршрути для швидкого доступу (додаток Б.4).

Взаємодія з Google картами Інтеграція Google карт в мій додаток дозволила користувачам переглядати маршрути автобусів безпосередньо на карті, забезпечуючи більш наочне та зручне представлення інформації. Ось як я реалізував цю функціональність:

1. Інтеграція Google Maps SDK:

Спочатку необхідно додати залежності Google Maps SDK в build.gradle файл (додаток Б.5).

2. Додавання карти на екран:

В activity_maps.xml я додав елемент MapView для відображення карт (додаток Б.6).

3. Налаштування MapView в Activity:

В MapsActivity.kt, я ініціалізував MapView та встановив початкові параметри карти, такі як місцезнаходження та тип карти (додаток Б.7).

4. Відображення маршрутів на карті:

В MapsActivity я також реалізував логіку для відображення маршрутів на карті, використовуючи Polyline (додаток Б.8).

Опис реалізації ключових функцій додатку.

1. Перегляд розкладу автобусів:

Однією з основних функцій додатку є можливість перегляду розкладу автобусів. Ця функція дозволяє користувачам швидко знайти актуальну інформацію про часи відправлення та маршрути.

Реалізація інтерфейсу. Використовуючи RecyclerView, я створив динамічний список, що відображає всі доступні маршрути з їх розкладами.

Завантаження даних. Інформація про розклади завантажується з локальної бази даних або API (за потреби), і відображається у списку.

2. Функція пошуку маршрутів:

Ця функція дозволяє користувачам шукати маршрути, використовуючи ключові слова, такі як назва зупинки або номер маршруту.

Я реалізував алгоритм, який дозволяє швидко відфільтровувати та відображати результати пошуку згідно з запитом користувача.

Після виконання пошуку інтерфейс динамічно оновлюється, показуючи результати, що відповідають запиту користувача.

3. Відображення маршрутів на Google Картах:

Інтеграція з Google Картами дозволяє користувачам візуально переглядати маршрути автобусів.

Я використав полілінії на карті для відображення маршрутів автобусів, що дозволяє користувачам бачити точний маршрут руху автобуса.

На карті також відображаються маркери зупинок, надаючи користувачам інформацію про місця зупинок вздовж маршруту.

4. Налаштування та кастомізація:

Додаток надає користувачам можливість налаштувати деякі аспекти інтерфейсу та функціональності відповідно до їхніх переваг.

Користувачі можуть зберігати улюблені маршрути для швидкого доступу.

4.5. Висновки до розділу 4

У процесі розробки мого додатку для моніторингу розкладів автобусів у місті Прилуки, я виконав ряд ключових завдань, що сприяли створенню ефективного та корисного мобільного застосунку.

Велика увага була приділена дизайну інтерфейсу користувача, забезпечуючи інтуїтивно зрозумілу та зручну навігацію. Це було досягнуто завдяки ретельній роботі з Figma та використанню сучасних підходів у дизайні мобільних додатків.

Використання Kotlin та Android Studio значно спростило процес розробки, забезпечуючи високу продуктивність і надійність застосунку. Інтеграція з

Google Maps SDK відіграла ключову роль у візуалізації маршрутів та поліпшенні загального користувацького досвіду.

Систематичне тестування, включаючи юніт-тести, інтеграційні тести та UI тести, а також бета-тестування з реальними користувачами, гарантувало стабільність та надійність додатку. Виявлення та усунення помилок було здійснено з використанням ефективних інструментів логування та моніторингу.

Додаток був спроектований таким чином, що він може бути легко адаптований та розширений для інших міст або регіонів. Це забезпечує потенціал для майбутнього розвитку та використання додатку в ширшому контексті.

Розробка цього додатку сприяла поліпшенню доступності та зручності використання громадського транспорту в місті Прилуки, забезпечуючи мешканцям та гостям міста актуальну інформацію про розклади автобусів.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

У процесі розробки цього випускного кваліфікаційного проекту, я провів глибоке дослідження та розробку мобільного додатку для моніторингу громадського транспорту міста Прилуки. Проект включав аналіз потреб користувачів, вибір та застосування сучасних технологій, дизайн інтерфейсу користувача та розробку функціональності додатку, його тестування та оптимізацію.

Впровадження Kotlin та Android Studio як основних інструментів розробки значно підвищило ефективність процесу розробки. Інтеграція з Google Maps SDK відіграла важливу роль у наданні географічної контекстуалізації даних, що робить додаток не тільки інформативним, але й інтуїтивно зрозумілим.

Основна увага була приділена створенню зручного та доступного інтерфейсу, що відповідає потребам широкого кола користувачів. Чистий та інтуїтивно зрозумілий дизайн сприяє легкості використання додатку.

В майбутньому додаток може бути розширений за рахунок введення нових функцій, таких як інтеграція з соціальними медіа для обміну інформацією про стан громадського транспорту в реальному часі.

Проект має потенціал для адаптації та застосування в інших містах, де існує потреба в ефективному моніторингу громадського транспорту.

У підсумку, розробка цього мобільного додатку була не лише технічним викликом, а й важливим кроком у покращенні міської інфраструктури та якості життя мешканців Прилук. Цей проект показав, як технології можуть

бути використані для вирішення реальних проблем у громадському транспорті та стати цінним доповненням до міської екосистеми.

Висновки до цього проекту логічно випливають з проведеного аналізу та розробки, підкреслюючи важливість технологій у сучасному міському середовищі та можливості їх застосування для поліпшення повсякденного життя громадян.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Жемеров, Д. та Іскандеров, С. "Kotlin in Action". Manning Publications, 2017. – 360 с.
2. "Kotlin vs Java: Efficiency, Speed and Performance Comparison". URL: https://www.raywenderlich.com/1323816-kotlin-vs-java-efficiency-speed-performance-comparison(дата звернення: 15.10.2023).
3. Freeman, E., Robson, E., Bates, B., Sierra, K. "Head First Design Patterns: A Brain-Friendly Guide". O'Reilly Media, 2020. – 694 с.Лейва, А.
4. Pro Android with Kotlin by Peter Späth. Apress, 2018. – 370 с.Спатару, П. "Pro Android with Kotlin: Developing Modern Mobile Apps". Apress, 2018. – 370 с.
5. Райан, М. "Android App Development For Dummies". For Dummies, 2018. – 384 с.
6. Android Developers. "Android Jetpack Documentation". URL: https://developer.android.com/jetpack. (дата звернення: 16.10.2023).
7. EasyWay.– Доступно на: https://play.google.com/store/apps/developer?id=EasyWay&hl=ru&gl=US. (дата звернення: 18.10.2023).
8. Розробник додатку "iCity DTransport". – Доступно на:https://play.google.com/store/apps/details?id=com.icitydtransport&hl=ru&gl=US. (дата звернення: 18.10.2023).
9. Біг Нерд Ранч. "Android Programming: The Big Nerd Ranch Guide". 4-те видання. Big Nerd Ranch, 2019. – 524 с.

10. Райан, М. "Android App Development For Dummies". For Dummies, 2018. – 384 с.
11. Філліпс, Б., Стюарт, К., Марсікано, К. "Android Programming with Kotlin for Beginners". Packt Publishing, 2019. – 674 с.

ТЕХНІЧНЕ ЗАВДАННЯ

Загальні відомості

1.1. Найменування системи:

1.1.1. Повне найменування системи - детальний опис з повною назвою.

1.1.2. Скорочене найменування системи - офіційна абревіатура або скорочення.

1.2. Планові терміни робіт: Чіткі дати початку та завершення роботи над проектом.

1.3. Порядок оформлення результатів: Стандартизований формат подання результатів і звітів.

1.4. Користувачі системи: Визначення основного бенефіціара та потенційних користувачів системи.

1.5. Відповідальні особи: Перелік осіб, відповідальних за різні аспекти розвитку та експлуатації системи.

Мета та призначення системи

2.1. Призначення системи: Огляд основних функцій та ролі системи у контексті задоволення потреб користувачів.

2.2. Мета створення: Визначення кінцевих цілей та бізнес-переваг, які система має принести.

2.3. Сфера застосування: Опис областей використання та потенційного впливу системи.

Вимоги до системи

3.1. Загальні вимоги:

					ДТЕУ 121 02-16.МР			
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*				
Зав. каф.		Криворучко О.В.		01.11.23	*Програмний додаток контролю дотримання часового режиму руху громадського транспорту*	*Стадія*	*Аркуш*	*Аркушів*
Керівник		Котенко Н.О.		01.11.23		*ТЗ*	*51*	*53*
Гарант		Котенко Н.О.		01.11.23		*Факультет інформаційних технологій 2м курс, 2 група*		
Розробив		Пономаренко С.В.		01.11.23	*Технічне завдання*			

3.1.1. Вимоги до структури системи, включаючи всі підсистеми та їх взаємодію.

3.1.2. Вимоги до надійності, включаючи показники надійності та вимоги до стійкості системи до помилок.

3.1.3. Вимоги до ергономіки та технічної естетики, забезпечуючи зручність та привабливість інтерфейсу.

3.1.4. Вимоги до безпеки, включаючи захист даних та протидію кіберзагрозам.

3.1.5. Вимоги до масштабованості: Врахування потенційного росту та розширення системи в майбутньому.

Технічне забезпечення

4.1. Обладнання та компоненти: Детальний перелік та характеристики всього необхідного обладнання та апаратних компонентів.

4.2. Програмне забезпечення: Визначення необхідних програмних платформ, мов програмування та інструментів.

4.3. Системи управління базами даних: Вибір та характеристика використовуваних СУБД.

ПРОГРАМА ТА МЕТОДИКА ТЕСТУВАННЯ

Тестування та дебагінг були ключовими компонентами у процесі розробки мого додатку. Ці етапи забезпечили стабільність, надійність та ефективність додатку.

Я написав юніт-тести для основних функцій додатку, використовуючи JUnit. Це дозволило мені перевірити правильність роботи індивідуальних компонентів, таких як алгоритми пошуку, обробка даних та інтеракція з UI. Kotlin (додаток Б.9).

Я також виконав інтеграційні тести, щоб переконатися, що різні частини додатку ефективно працюють разом. Це включало тестування взаємодії між додатком і базою даних, а також інтеграцію з зовнішніми сервісами, такими як Google Maps.

Я використовував Espresso для тестування інтерактивних елементів UI, забезпечуючи плавність та відповідність дизайну.

Я активно використовував логування для виявлення помилок під час розробки. Android Logcat був незамінним інструментом для моніторингу внутрішньої роботи додатку та виявлення несподіваних поведінок.

Я провів стрес-тестування, щоб забезпечити стабільність додатку під високим навантаженням, особливо при великій кількості одночасних запитів до бази даних.

Перед запуском додатку я організував бета-тестування з реальними користувачами, щоб отримати зворотний зв'язок та виявити потенційні проблеми у користувацькому досвіді.

Після запуску додатку я проводив моніторинг.

ДОДАТКИ

Додаток А

Лістинг програмного коду

Grafiks

```
package com.example.prylukibus

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity

class grafiks: AppCompatActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_grafiks
   }
}
```

MainActivity

```
package com.example.prylukibus

import android.content.Intent
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import android.widget.Button

class MainActivity : AppCompatActivity() {

   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
```

```
        setContentView(R.layout.activity_main)

        val myButton: Button = findViewById(R.id.myButton)

        myButton.setOnClickListener {
            val intent = Intent(this@MainActivity, NextActivity::class.java)
            startActivity(intent)
        }
    }
}
```

NextActivity

```
package com.example.prylukibus

import android.content.Intent
import android.os.Bundle
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.cardview.widget.CardView

class NextActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_next)

        val marshrutkiCardView: CardView = findViewById(R.id.cardView1)

        marshrutkiCardView.setOnClickListener {
```

```
        val intent = Intent(this@NextActivity, NextActivity1::class.java)
        startActivity(intent)
    }
  }
}
```

NextActivity1

```
package com.example.prylukibus

import android.content.Intent
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import android.widget.TextView

class NextActivity1 : AppCompatActivity() {
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_next1)

    val textView4 = findViewById<TextView>(R.id.textView4)

    textView4.setOnClickListener {
      val intent = Intent(this@NextActivity1, grafiks::class.java)
      startActivity(intent)
    }
  }
}
```

activity_grafiks

```
<?xml version="1.0" encoding="utf-8"?>
<!-- res/layout/activity_main.xml -->
<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"
    android:paddingLeft="0dp"
    android:paddingTop="0dp"
    android:paddingRight="0dp"
    android:paddingBottom="0dp"
    android:background="#E1EBFA"
    tools:context=".MainActivity">
</RelativeLayout>
```

activity_main

```
<?xml version="1.0" encoding="utf-8"?>
<!-- res/layout/activity_main.xml -->
<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"
    android:paddingLeft="0dp"
    android:paddingTop="0dp"
    android:paddingRight="0dp"
    android:paddingBottom="0dp"
```

```
android:background="#1B4C94"
tools:context=".MainActivity">


<View
   android:id="@+id/some_id"
   android:layout_width="412dp"
   android:layout_height="80dp"
   android:layout_alignParentLeft="true"
   android:layout_alignParentBottom="true"
   android:layout_marginLeft="-1dp"
   android:layout_marginBottom="-2dp"
   android:background="#438ACC" />

<Button
   android:id="@+id/myButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignParentTop="true"
   android:layout_centerHorizontal="true"
   android:layout_marginTop="200dp"
   android:backgroundTint="#ffffff"
   android:text="Прилуки"
   android:textColor="#000000" />

<TextView
   android:id="@+id/myText"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Оберіть місто :"
```

```
    android:layout_below="@id/myButton"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="-120dp"
    android:textSize="20sp"
    android:textColor="#FFffff"
    />

<ImageView
    android:id="@+id/imageView4"
    android:layout_width="50dp"
    android:layout_height="40dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="50dp"
    android:layout_marginBottom="15dp"
    app:srcCompat="@drawable/m4" />

<ImageView
    android:id="@+id/imageView"
    android:layout_width="50dp"
    android:layout_height="40dp"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="50dp"
    android:layout_marginBottom="15dp"
    app:srcCompat="@drawable/m5" />


<ImageView
    android:id="@+id/imageView2"
```

```
    android:layout_width="50dp"
    android:layout_height="40dp"
    android:layout_alignParentStart="true"

    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="180dp"
    android:layout_marginEnd="180dp"
    android:layout_marginBottom="18dp"
    app:srcCompat="@drawable/m6" />

<TextView
    android:id="@+id/textView"
    android:layout_width="77dp"
    android:layout_height="21dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="45dp"
    android:layout_marginBottom="-5dp"
    android:text="Маршрути"
    android:textColor="#FFffff"
    android:textSize="12sp" />

<TextView
    android:id="@+id/textView3"
    android:layout_width="77dp"
    android:layout_height="21dp"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="24dp"
```

```
        android:layout_marginBottom="-5dp"
        android:text="Улюблені"
        android:textColor="#FFffff"
        android:textSize="12sp" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="77dp"
        android:layout_height="21dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="190dp"
        android:layout_marginEnd="144dp"
        android:layout_marginBottom="-5dp"
        android:text="Карта"
        android:textColor="#FFffff"
        android:textSize="12sp" />

</RelativeLayout>
```

activity_next

```
<?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"
    android:paddingLeft="0dp"
```

```
android:paddingTop="0dp"
android:paddingRight="0dp"
android:paddingBottom="0dp"
android:background="#1B4C94"
tools:context=".NextActivity">

<RelativeLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="#1B4C94"
   android:paddingLeft="0dp"
   android:paddingTop="0dp"
   android:paddingRight="0dp"
   android:paddingBottom="0dp"
   tools:context=".NextActivity">

   <androidx.cardview.widget.CardView
      android:id="@+id/cardView1"
      android:layout_width="120dp"
      android:layout_height="wrap_content"
      android:layout_centerInParent="true"
      android:layout_marginTop="16dp"
      android:background="#80FFFFFF"
      app:cardCornerRadius="8dp"
      app:cardElevation="4dp">

      <TextView
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Маршрутки"
```

```
        android:textColor="#000000" />
    </androidx.cardview.widget.CardView>

    <androidx.cardview.widget.CardView
        android:id="@+id/cardView2"
        android:layout_width="120dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/cardView1"
        android:layout_centerInParent="true"
        android:layout_marginTop="16dp"
        android:background="#80FFFFFF"
        app:cardCornerRadius="8dp"
        app:cardElevation="4dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Тролейбуси"
            android:textColor="#000000" />
    </androidx.cardview.widget.CardView>

    <androidx.cardview.widget.CardView
        android:id="@+id/cardView3"
        android:layout_width="120dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/cardView2"
        android:layout_centerInParent="true"
        android:layout_marginTop="16dp"
        android:background="#80FFFFFF"
        app:cardCornerRadius="8dp"
```

```
    app:cardElevation="4dp">

    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Трамваї"
      android:textColor="#000000" />
  </androidx.cardview.widget.CardView>

  <View
    android:id="@+id/some_id"
    android:layout_width="412dp"
    android:layout_height="80dp"
    android:layout_alignParentBottom="true"
    android:layout_marginBottom="-2dp"
    android:background="#438ACC" />

</RelativeLayout>

<ImageView
  android:id="@+id/imageView4"
  android:layout_width="50dp"
  android:layout_height="40dp"
  android:layout_alignParentStart="true"
  android:layout_alignParentBottom="true"
  android:layout_marginStart="50dp"
  android:layout_marginBottom="15dp"
  app:srcCompat="@drawable/m4" />

<ImageView
```

```
    android:id="@+id/imageView"
    android:layout_width="50dp"
    android:layout_height="40dp"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="50dp"
    android:layout_marginBottom="15dp"
    app:srcCompat="@drawable/m5" />

<ImageView
    android:id="@+id/imageView2"
    android:layout_width="50dp"
    android:layout_height="40dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="180dp"
    android:layout_marginEnd="180dp"
    android:layout_marginBottom="18dp"
    app:srcCompat="@drawable/m6" />

<TextView
    android:id="@+id/textView"
    android:layout_width="77dp"
    android:layout_height="21dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="45dp"
    android:layout_marginBottom="-5dp"
    android:text="Маршрути"
```

```
    android:textColor="#FFFFFF"
    android:textSize="12sp" />

  <TextView
    android:id="@+id/textView3"
    android:layout_width="77dp"
    android:layout_height="21dp"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="24dp"
    android:layout_marginBottom="-5dp"
    android:text="Улюбленi"
    android:textColor="#FFFFFF"
    android:textSize="12sp" />

  <TextView
    android:id="@+id/textView2"
    android:layout_width="77dp"
    android:layout_height="21dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="190dp"
    android:layout_marginEnd="144dp"
    android:layout_marginBottom="-5dp"
    android:text="Карта"
    android:textColor="#FFFFFF"
    android:textSize="12sp" />

</RelativeLayout>
```

activity_next1

```
<?xml version="1.0" encoding="utf-8"?>
<!-- res/layout/activity_main.xml -->
<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"
  android:paddingLeft="0dp"
  android:paddingTop="0dp"
  android:paddingRight="0dp"
  android:paddingBottom="0dp"
  android:background="#E1EBFA"
  tools:context=".MainActivity">

  <RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#E1EBFA"
    android:paddingLeft="0dp"
    android:paddingTop="0dp"
    android:paddingRight="0dp"
    android:paddingBottom="0dp"
    tools:context=".NextActivity">

    <androidx.cardview.widget.CardView
      android:id="@+id/cardView1"
```

```
    android:layout_width="120dp"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="16dp"
    android:background="#80FFFFFF"
    app:cardCornerRadius="8dp"
    app:cardElevation="4dp">

    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Маршрутки"
      android:textColor="#000000" />
</androidx.cardview.widget.CardView>

<View
    android:id="@+id/some_id"
    android:layout_width="412dp"
    android:layout_height="80dp"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="-6dp"
    android:layout_marginLeft="-6dp"
    android:layout_marginEnd="-1dp"
    android:layout_marginBottom="1dp"
    android:background="#438ACC" />

<ImageView
    android:id="@+id/imageView8"
    android:layout_width="44dp"
```

```
    android:layout_height="40dp"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="347dp"
    android:layout_marginBottom="580dp"
    app:srcCompat="@drawable/m10" />

  <ImageView
    android:id="@+id/imageView9"
    android:layout_width="23dp"
    android:layout_height="23dp"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="21dp"
    android:layout_marginBottom="593dp"
    app:srcCompat="@drawable/m12" />

  <ImageView
    android:id="@+id/imageView10"
    android:layout_width="78dp"
    android:layout_height="59dp"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="332dp"
    android:layout_marginBottom="621dp"
    app:srcCompat="@drawable/m7" />

  <ImageView
    android:id="@+id/imageView11"
    android:layout_width="33dp"
```

```
    android:layout_height="26dp"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="144dp"
    android:layout_marginBottom="638dp"
    app:srcCompat="@drawable/m8" />

<ImageView
    android:id="@+id/imageView12"
    android:layout_width="28dp"
    android:layout_height="33dp"
    android:layout_alignParentTop="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginTop="63dp"
    android:layout_marginEnd="23dp"
    android:layout_marginBottom="634dp"
    app:srcCompat="@drawable/m9" />

<TextView
    android:id="@+id/textView4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="167dp"
    android:layout_marginBottom="591dp"
    android:text="«Дослідна – Плискунівка»" />

<TextView
```

```
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="302dp"
        android:layout_marginBottom="642dp"
        android:text="№1" />

    <TextView
        android:id="@+id/textView6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="65dp"
        android:layout_marginBottom="644dp"
        android:text="Слідкувати" />
</RelativeLayout>
<ImageView
    android:id="@+id/imageView4"
    android:layout_width="50dp"
    android:layout_height="40dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="50dp"
    android:layout_marginBottom="15dp"
    app:srcCompat="@drawable/m4" />
<ImageView
    android:id="@+id/imageView"
```

```
    android:layout_width="50dp"
    android:layout_height="40dp"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="50dp"
    android:layout_marginBottom="15dp"
    app:srcCompat="@drawable/m5" />

<ImageView
    android:id="@+id/imageView2"
    android:layout_width="50dp"
    android:layout_height="40dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="180dp"
    android:layout_marginEnd="180dp"
    android:layout_marginBottom="18dp"
    app:srcCompat="@drawable/m6" />

<TextView
    android:id="@+id/textView"
    android:layout_width="77dp"
    android:layout_height="21dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="45dp"
    android:layout_marginBottom="-5dp"
    android:text="Маршрути"
    android:textColor="#FFFFFF"
```

```
        android:textSize="12sp" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="77dp"
        android:layout_height="21dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="24dp"
        android:layout_marginBottom="-5dp"
        android:text="Улюбленi"
        android:textColor="#FFFFFF"
        android:textSize="12sp" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="77dp"
        android:layout_height="21dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="190dp"
        android:layout_marginEnd="144dp"
        android:layout_marginBottom="-5dp"
        android:text="Карта"
        android:textColor="#FFFFFF"
        android:textSize="12sp" />
</RelativeLayout>
```

Лістинг прграмного коду.

Важливі елементи

Б.1

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val buttonViewSchedule =
findViewById<Button>(R.id.button_view_schedule)
        buttonViewSchedule.setOnClickListener {
            // Логіка для переходу до екрану розкладу
        }
    }
}
```

Б.2

```
class RoutesAdapter(private val routes: List<Route>) :
RecyclerView.Adapter<RoutesAdapter.ViewHolder>() {
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.item_route, parent, false)
        return ViewHolder(view)
    }
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val route = routes[position]
        holder.routeName.text = route.name
```

```
            // Додаткова логіка налаштування ViewHolder
        }
        class ViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
            val routeName: TextView =
itemView.findViewById(R.id.text_route_name)
        }
    }
```

Б.3

```
fun searchRoutes(query: String): List<Route> {
        // Припустимо, у нас є база даних або список маршрутів
        return routesDatabase.filter { it.name.contains(query, ignoreCase = true) }
    }
```

Б.4

```
    class FavoritesManager(private val context: Context) {
        private val sharedPreferences =
context.getSharedPreferences("favorites_prefs", Context.MODE_PRIVATE)

        fun saveFavoriteRoute(route: Route) {
            val editor = sharedPreferences.edit()
            editor.putString(route.id.toString(), route.name)
            editor.apply()
        }

        fun getFavoriteRoutes(): List<Route> {
            val allEntries = sharedPreferences.all
```

```
            return allEntries.map { Route(it.key.toInt(), it.value.toString()) }
        }
    }
```

Б.5

```
dependencies {
    implementation 'com.google.android.gms:play-services-maps:17.0.0'
}
```

Б.6

```
<com.google.android.gms.maps.MapView
    android:id="@+id/map_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Б.7

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback {

    private lateinit var mapView: MapView
    private lateinit var googleMap: GoogleMap
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_maps)
        mapView = findViewById(R.id.map_view)
        mapView.onCreate(savedInstanceState)
        mapView.getMapAsync(this)
    }
```

```
    override fun onMapReady(map: GoogleMap) {
        googleMap = map
        setInitialLocation()
    }
    private fun setInitialLocation() {
    }
}
```

Б.8

```
fun displayRouteOnMap(route: Route) {
    val polylineOptions = PolylineOptions().apply {
        width(5f)
        color(Color.RED)
        geodesic(true)
    }
    route.stops.forEach { stop ->
        polylineOptions.add(LatLng(stop.latitude, stop.longitude))
    }
    googleMap.addPolyline(polylineOptions)
}
```

Б.9

```
    @Test
    fun testRouteSearchAlgorithm() {
        val searchResult = searchRoutes("Прилуки")
        assertTrue("Пошук повинен повернути результати",
searchResult.isNotEmpty())
    }
```