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

# БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

**" Програмне забезпечення для прогнозування замовлень таксі на основі статистичної обробки отриманих даних"**

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: студент 4 курсу, групи
2КІ-18б
напряму підготовки (спеціальності)
123 – «Комп'ютерна інженерія»
(шифр і назва напряму підготовки, спеціальності)
Черняховський І.Ю.
(прізвище та ініціали)
Керівник Ткаченко О.М.
(прізвище та ініціали)
Рецензент Карпінець В.В.
(прізвище та ініціали)

**Допущено до захисту**
д.т.н., проф. Азаров О.Д.

" 20 " червня 2022 р.

Вінниця ВНТУ — 2022 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність 123 — «Комп'ютерна інженерія»

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри**
обчислювальної техніки
проф. Азарову О.Д.
«06» 02 2022 р.

**З А В Д А Н Н Я**
**НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ**
Черняховському Ігорю Юрійовичу

1 Тема роботи: «Програмне забезпечення для прогнозування замовлень таксі на основі статистичної обробки отриманих даних», керівник роботи к.т.н., доц. каф. ОТ Ткаченко Олександ Миколайович, затверджені наказом вищого навчального закладу від «24» березня 2022 року № 66

2 Строк подання роботи студентом 23.06.2022

3 Вихідні дані до роботи — призначення — додаток для таксі; основні підтримувані функції — зчитування вхідних та вихідних даних, пошук водія.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз предметної області. Розробка програмного забезпеченя для андроід.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): технічне завдання, головна сторінка програмного забезпечення, меню та налаштування.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| Спеціальна частина | Ткаченко О. М, доцент кафедри ОТ | | |

7 Дата видачі завдання «24» березня 2022 р.

8 Календарний план виконання БДР приведений в таблиці 2.

Таблиця 2 — Календарний план

| № з/п | Назва етапів виконання комплексної бакалаврської роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Постановка задачі роботи | 08.03.22 | вик. |
| 2 | Аналіз предметної області | 09.03-19.03.22 | вик. |
| 3 | Аналіз сучасних підходів до побудови програмного забезпечення | 21.03-30.03.22 | вик. |
| 4 | Аналіз поняття програмного забезпечення для таксі | 05.04-16.04.22 | вик |
| 5 | Аналіз принципів програмного забезпечення | 17.04-23.04.22 | вик |
| 6 | Проектування програмного забезпечення | 25.04-.06.05.22 | вик. |
| 7 | Розробка макету програмного забезпечення | 09.05-13.05.22 | вик. |
| 8 | Розробка функціоналу програмного забезпечення | 16.05-21.05.22 | вик. |
| 9 | Підготовка матеріалів та опис розробки інформаційної системи | 22.05-26.05.22 | вик. |
| 10 | Аналіз виконання роботи, висновки, додатки | 28.05-31.05.22 | вик. |
| 11 | Перевірка якості виконання бакалаврського проекту та усунення недоліків | 01.06 -.08.06.22 | вик. |

Студент _____ Черняховський І. Ю.

Керівник роботи _____ Ткаченко О. М.

# АНОТАЦІЯ

Черняховський І.Ю. Програмне забезпечення для прогнозування замовлень таксі на основі статистичної обробки отриманих даних. Бакалаврська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна Інженерія, Вінниця: ВНТУ, 2022. Пояснювальна записка містить 95 сторінки, 30 рисунків та 24 посилань.

В даній бакалаврській дипломній роботі було розроблено програмне забезпеченяна андроід для замовлення таксі. На основі здійсненого аналізу предметної області було проаналізовано сучасні підходи до систем таксі та проаналізовано додатки та сервіси конкурентів. Відповідно до поставленої задачі був розроблений інтерфейс, з використанням Kotlin. Таким чином було отримано зручне прогрмне забезпеченя для замовлень таксі.

Ключові слова: програмне забезпечення, додаток, інтерфейс, статистична обробка.

# ABSTRACT

Chernyakhovsky I.Yu. Software for forecasting taxi orders based on statistical processing of the received data. Bachelor's degree in specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2022. Explanatory note contains 95 pages, 30 figures and 24 references.

In this bachelor's thesis was developed android software for ordering a taxi. Based on the analysis of the subject area, modern approaches to taxi systems were analyzed and applications and services of competitors were analyzed. In accordance with the task, an interface was developed using Kotlin. Thus, a convenient software for taxi orders was obtained.

Keywords: software, application, interface, statistical processing.

**ЗМІСТ**

**ВСТУП**.....8

**1 АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ ЗАБЕЗПЕЧЕННЬ, ІНСТРУМЕНТІВ ТА СТРУКТУР МОБІЛЬНОГО ЗАСТОСУНКУ НА ОСНОВІ СТАТИСТИЧНОЇ ОБРОБКИ ОТРИМАНИХ ДАНИХ**.....10

1.1 Концепція програмного забезпечення для замовлень таксі .....10

1.2 Аналіз аналогів програмного забезпечення .....11

1.2.1 Uber.....11

1.2.2 Lyft.....12

1.2.3 Bolt.....13

1.2.4 Порівняння аналогів .....14

1.3 Вибір мови програмування.....16

1.3.1 Мова програмування C#.....16

1.3.2 Мова програмування Java.....17

1.3.3 Мова програмування Kotlin.....18

1.4 Вибір середовища розробки .....19

1.4.1 Середовище Eclipse IDE .....20

1.4.2 Середовище Android Studio .....22

**2 ПРОГРАМНИЙ ЗАСТОСУНОК ДЛЯ ЗАМОВЛЕНЬ ТАКСІ НА ОСНОВІ СТАСТИЧНОЇ ОБРОБКИ ОТРИМАНИХ ДАННИХ** .....25

2.1 Ціль та задачі розроблення додатка.....25

2.2.1 Початкові дані розробки додатка.....26

2.2.2 Побудова діаграм використання додатка .....26

2.2.3 Функціональні вимоги до додатка .....31

2.2.4 Нефункціональні вимоги до додатка .....32

2.2.5 Специфікації варіантів використання додатка.....32

2.2.6 Розробка макетів екранних форм додатка ..... 34
2.3.7 Опис БД ..... 42
2.3.8 Опис структури БД ..... 43
**3. ДОСЛІДЖЕННЯ ТА АНАЛІЗ РОЗРОБЛЕНОГО МОБІЛЬНОГО ДОДАТКА** ..... 45
3.1 Аналіз інтерфейсу користувача ..... 45
3.1.1 Тестування інтерфейсу мобільного додатка ..... 46
3.1.2 Обробка статистичних данних ..... 54
**ВИСНОВКИ** ..... 58
**ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ** ..... 60
**ДОДАТОК А** Технічне завдання ..... 63
**ДОДАТОК Б** Лістинг основного блоку програмного забезпечення ..... 67
**ДОДАТОК В** Лістинг основного блоку програмного забезпечення ..... 88
**ДОДАТОК Г** ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ ..... 95

# ВСТУП

На сьогоднішній день немає людей, які хоча б раз не скористалися послугами таксі. Робота таксі знаходиться під пильним оком ринку. Компанії, які надають неякісні послуги та виконують недобросовісну роботу, караються ринком за ігнорування таких таксі. І водії, і компанії, які організовують цю послугу, дуже добре розуміють важливість заробітку та збереження іміджу організації.

Тому виникає потреба в автоматизації служби таксі з використанням статистичної обробки даних, що дозволяє повністю виключити рутинні операції та досягти максимального рівня автоматизації служби. Зокрема, для прийому заяв, розподілу водіїв по чергах та обліку роботи з фізичними та юридичними особами необхідно виконати мінімум дій, а в деяких випадках ці дії виконуються автоматично. Наявність безлічі різних регістрів, які оновлюються в процесі, зводить всі дані до результату статистичної обробки

Тому було вирішено створити програмне забезпечення для замовлення таксі на основі статистичної обробки даних.

**Актуальність дослідження** обумовлена необхідністю створення програмного забезпечення для замовлень таксі на основі стастичної обробки отриманих данних, з метою зменшити частку ручної праці та кількості паперових документів, автоматизувати підприємство і наочно демонструвати зайнятість водіїв.

**Об'єкт дослідження** процеси, які відбуваються в програмному забезпеченні для замовлень таксі на основі стастичної обробки отриманих данних.

**Предмет дослідження:** є методи та засоби статистичной обробки отриманих даних в програмному забезпечені для виклику таксі

**Метою дослідження** є розробка програмного забезпечення для замовлень таксі на основі стастичної обробки отриманих данних.

Для досягнення поставленої мети необхідно вирішити наступні

задачі:

— описати концепцію програмного забезпечення;

— проаналізувати аналогів програмного забезпечення;

— порівняти аналоги програмного забезпечення;

— обрати мову програмування;

— обрати середовище розробки;

— описати структуру програмного забезпечення;

— розробити програмне забезпечення для замовлень таксі на основі стастичної обробки отриманих данних.

**Методи дослідження** дипломної роботи: використовувався та досліджувався метод статистичной обробки на освнові отриманих даних

# 1 АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ ЗАБЕЗПЕЧЕННЬ, ІНСТРУМЕНТІВ ТА СТРУКТУР МОБІЛЬНОГО ЗАСТОСУНКУ НА ОСНОВІ СТАТИСТИЧНОЇ ОБРОБКИ ОТРИМАНИХ ДАНИХ

## 1.1 Концепція програмного забезпечення для замовлень таксі

В останні роки набув поширення наступний алгоритм обробки даних. Спочатку диспетчер отримав дзвінок, а потім оператор мав зв’язатися з водієм і надіслати інформацію про замовлення, наприклад, по радіо. Однак з розвитком Інтернету з’явилося багато способів використання цієї мережі в такому сервісі. Тепер досить легко замовити таксі в інтернеті або за допомогою спеціальних додатків для смартфонів. Таким чином, ефективним завданням зібраної інформації є підвищення ефективності обслуговування. Наприклад, ви можете передбачити кількість необхідних диспетчерів і водіїв, роблячи точні прогнози замовлень і відстежуючи виконання замовлень. Це можна зробити за допомогою статистична обробка отриманих даних.

Статистика — це наука, яка в основному включає збір даних, інтерпретацію даних і, нарешті, перевірку даних. Аналіз статистичних даних є методом виконання різноманітних статистичних операцій. Це тип кількісного дослідження, який фокусується на кількісній оцінці даних і зазвичай використовує певну форму статистичного аналізу. Кількісні дані в основному включають описові дані, такі як дані опитування та дані спостережень. Компанії з бронювання таксі хочуть кращої підтримки для програми; Тепер це стало глобальною метою. Готове рішення для бізнесу таксі використовує природні можливості для розширення меж бізнесу та управління на єдиній платформі.

Програмне забезпечення для замовлення таксі — це мобільна система, яка дозволяє керувати завданням замовлення таксі на онлайн-платформі. Клієнти можуть замовити таксі онлайн, що забезпечує безперервний процес для водіїв і споживачів. Програмне забезпечення для управління таксі приносить більше

доходів, розширюючи спектр послуг у багатьох країнах за рахунок повного обміну валют та оперативного управління.

## 1.2 Аналіз аналогів програмного забезпечення

Згідно зі статистикою, 97.4% замовлень таксі здійснюються через мобільні телефони, а 78.8% від загальної кількості замовлень належать власникам Android, у цьому дослідженні було прийнято рішення про розробку програмного забезпечення для замовлення таксі для мобільного додатка на базі Android.

Мобільні пристрої вже давно стали невід'ємною частиною нашого життя. Але тепер це не просто телефони для дзвінків і смс. Тепер у смартфонах має бути все необхідне для спілкування, навчання, роботи та розваг. У зв'язку з цим зараз існує безліч додатків, ігор та інших програм, розроблених спеціально для цієї мети. Популярність мобільних додатків постійно зростає, оскільки вони зручні та швидкі. Одним з основних напрямків у розробці мобільних додатків є сфера замовлення послуг. Сюди входять усі види квиткових послуг, таксі, автомийки тощо. Основною перевагою таких програм є більш швидке бронювання в порівнянні з дзвінком оператора або замовленням на сайті. У даному дослідженні розглядається процес розробки Android-додатку, що дозволяє замовити таксі на основі статистичної обробки даних, а також надати оновлену інформацію про всі розділи, новини та акції даного сервісу. Отже, давайте розглянемо і розберемо аналоги таких мобільних додатків.

### 1.2.1 Uber

Uber — Американський постачальник послуг мобільності. Ця компанія розташований у Сан-Франциско і працює приблизно в 72 країнах і 10 600 містах. Його послуги включають виклики таксі, доставку їжі, посилок, кур’єрів, доставку вантажів, оренду електровелосипедів і скутерів, а також перевезення поромів у партнерстві з місцевими операторами. Uber не володіє транспортними засобами;

Замість цього він стягує комісію за кожне бронювання. Тарифи встановлюються клієнтом заздалегідь, але можуть бути змінені за допомогою динамічної моделі ціноутворення на основі місцевого попиту та пропозиції на момент бронювання. (Рис. 1.1 «Uber»).


Рисунок 1.1 — «Uber»

### 1.2.2 Lyft

Lyft — Американський постачальник транспортних послуг, який розробляє, продає та керує мобільним додатком, який пропонує послуги таксі, оренду автомобілів, скутерів, велосипедів, автомобілів та доставку їжі. Він розташований у Сан-Франциско, Каліфорнія, і працює в 647 містах США. У цієї компанії немає автомобілів; Замість цього він стягує комісію за кожне бронювання. Тарифи встановлюються клієнтом заздалегідь, але можуть бути змінені за допомогою

динамічної моделі ціноутворення на основі місцевого попиту та пропозиції на момент бронювання. (Рис. 1.2 «Lyft»).


Рисунок 1.2 — «Lyft»

### 1.2.3 Bolt

Bolt це естонська мобільна компанія, що надає послуги з використання автомобілів, мікромобілів, каршерингу та доставки їжі зі штаб-квартирою в Таллінні та працює в більш ніж 400 містах у більш ніж 45 країнах Європи, Африки, Західної Азії та Латинської Америки. У компанії 100 мільйонів клієнтів по всьому світу і понад 2,5 мільйона водіїв, які використовують платформу Bolt

Популярність сервісу дозволяє забезпечити велику кількість замовлень для водіїв. При визначенні тарифу агрегатори використовують "принцип голландського аукціону": встановлюють максимальну ціну, яка залежно від попиту зменшується. Якщо попит слабкий — ціна низька, якщо значний — висока.

В реаліях українських сервісів таксі кінцева вартість замовлення залежить від базового тарифу, відстані і тривалості поїздки. Driver для подорожей. (Рис 1.3. «Bolt»).

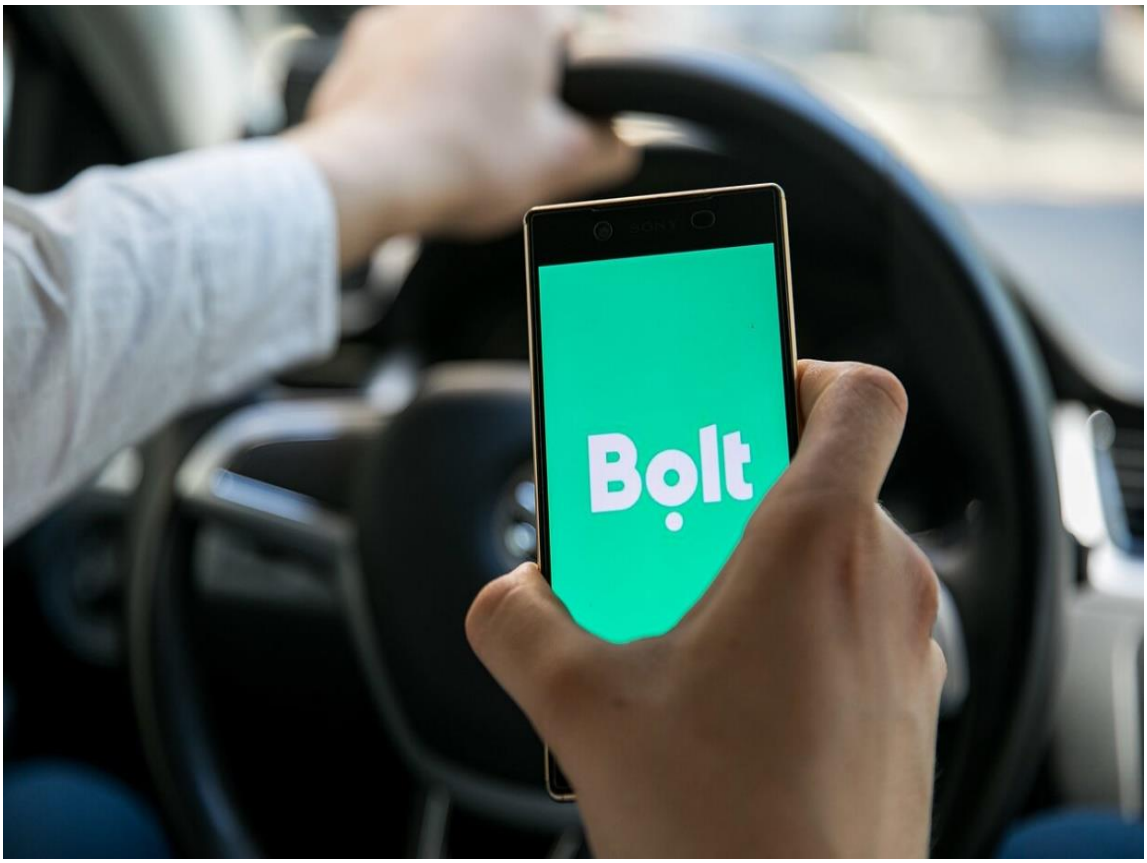


Рисунок 1.3 — «Bolt»

1.2.4 Порівняння аналогів

Проаналізувавши 3 аналоги мобільних додатків для замовлення таксі, наведемо порівняльну таблицю переваг та недоліків цих програмних продуктів B lyft клієнти можуть "конкурувати" за водія, збільшуючи або зменшуючи вартість поїздки. Це особливо актуально в години пік. Uber та Bolt списують кошти після поїздки, а початкова ціна може відрізнятися від кінцевої. Тобто спочатку клієнт бачить орієнтовну вартість поїздки. Якщо водій приїде швидше, то ціна не знизиться, а дорівнюватиме початковій оферті. Це простий варіант для споживача, але якщо в якийсь момент водій відмовиться від поїздки, то списану

суму агрегатор поверне не одразу, а через певний час. (Табл. 1.1 «Переваги та недоліки мобільних додатків «Uber», «Lyft», «Bolt»).

Табл. 1.1— Переваги та недоліки мобільних додатків «Uber», «Lyft», «Bolt»

| Назва додатка | Переваги | Недоліки |
|---|---|---|
| «Uber» | Компанія наявна в багатьох країнах світу;<br>Є додатки як для IOs так і для Android;<br>Потужний автопарк. | Незаконні дії компанії, через які відкрито багато судових процесів в різних країнах і тому, надійність для клієнта значно зменшується. |
| «Lyft» | Друга за розміром компанія з замовлення послуг таксі;<br>Є онлайн-підримка клієнтів;<br>Додаток є багатомовним. | Незаконні дії компанії, через які відкрито багато судових процесів в різних країнах і тому, надійність для клієнта значно зменшується. |
| «Bolt» | Мережа всередині країни;<br>Є гарантії як для водія так і для замовника;<br>Швидкий сервіс. | Незаконні дії компанії, через які відкрито багато судових процесів в різних країнах і тому, надійність для клієнта значно зменшується. |

## 1.3 Вибір мови програмування

Мова програмування — це будь-який набір певних правил, який у випадку візуальних мов програмування перетворює рядки програми або графічні елементи в різні типи виводу машинного коду. Мови програмування є різновидом комп’ютерної мови і використовуються в комп’ютерному програмуванні для реалізації алгоритмів. Більшість мов програмування містять інструкції для комп’ютерів. Існують запрограмовані машини, які використовують спеціальний список інструкцій замість суцільних мов програмування. Програми для цих машин не приводили до різних дій у відповідь на різні доходи чи умови.

### 1.3.1 Мова програмування C#

C# — сучасна, безпечна мова програмування, об’єктно-орієнтована мова, яка дозволяє програмістам швидко й легко створювати рішення для платформи Microsoft .NET.

C# — це легка сучасна об'єктно-орієнтована мова, похідна від C++ та Java. Дана мова була розроблена для поєднання потужного Visual Basic і надійного Power C++. Класи та типи даних є загальними для всіх мов .NET. Ми можемо розробити консольну програму, програму для Windows і веб-додаток за допомогою C#. У C# Microsoft подбала про проблеми C++, як-от керування пам’яттю, покажчики тощо. Він підтримує збір сміття, автоматичне керування пам’яттю тощо. (Рис. 1.4 Мова програмуваннння «C#»). Мова програмування C# переважно використовується для створення корпоративного програмного забезпечення, фінансових проектів, наприклад для банків і бірж, зокрема мобільних додатків, хмарних сервісів.C# у порівнянні з Java легше взаємодіє, з кодом програм, написаних на інших мовах. І саме на C# часто пишуться розширення для інших мов програмування, які використовуються у якості прошарку між бібліотекою C# і мовою, можливості якої під конкретні цілі планується розширювати. C# широко використовується в розробці ігор на Unity.


Рисунок 1.4 — Мова програмуванння «C#»

### 1.3.2 Мова програмування Java

Java — це об'єктно-орієнтована мова програмування. Вся Java є об’єктом. Об’єктно-орієнтований означає, що ми організовуємо наше програмне забезпечення як комбінацію різних типів об’єктів, які містять як дані, так і поведінку.

Список важливих особливостей Java:

— простий у вивчені;

— об'єктно-орієнтований;

— портативний;

— незалежний до платформ;

— динамічний;

— багатопотоковий;

— має нейтральну архітектуру;

— розподілений.


Рисунок 1.5 — «Мова програмування «Java»

При створенні мови Java було п'ять основних цілей, вона повинна бути:

— простою, об'єктно-орієнтованою і знайомо;

— вона повинна бути надійною та захищеною;

— вона повинна бути нейтральною з точки зору архітектури;

— вона повинна виконуватися з високою продуктивністю;

— вона повинна бути інтерпретованою і динамічною.

### 1.3.3 Мова програмування Kotlin

Kotlin — це статична об'єктно-орієнтована мова програмування, сумісна з Java. Kotlin економить час розробника, оскільки менш детальна мова забезпечує коротший і менш надлишковий код. Kotlin вважається заміною Java, хоча і несумісний із синтаксисом, він сумісний з кодом Java та бібліотеками

Kotlin також має власні бібліотеки, створені для додатків Android. Kotlin більш сучасний і спрощений, що полегшує навчання. Kotlin зосереджується на спрощеному коді функцій і уникає дублювання коду. Крапка з комою не потрібна

в кінці кожного рядка, проте нічого не станеться, якщо її поставити (Рис. 1.6 Мова програмування «Kotlin»).

```kotlin
import photomover.cli.web.StartWebOptions

fun main(arguments: Array<String>) {
  val commands = mapOf(
      "organize" to wrapOperation(::organize, OrganizeOptions()),
      "upload" to wrapOperation(::upload, UploadOptions()),
      "startWeb" to wrapOperation(::start, StartWebOptions()))
  val args = if (arguments.size == 0) {
    array("startWeb")
  } else {
    arguments
  }
  val command = commands[args[0]]
  if (command == null) {
    println("Command '${args[0]}' is not supported.")
    println("Please specify one of following: ${commands.keySet()}")
  } else {
    println("Starting ${args[0]}")
    command(args.drop(1))
  }
}

fun wrapOperation<T>(
    operation: (options: T) -> Unit, defaultOptions: T): (args: List<String>) -> Unit {
  return { args ->
    val parser = CmdLineParser(defaultOptions)
    try {
      parser.parseArgument(args)
    } catch (ex: CmdLineException) {
      println(ex.getMessage())
      println(parser.printUsage(System.err))
    }
    println("args: $defaultOptions")
    operation(defaultOptions)
  }
}
```

Рисунок 1.6 — Мова програмування «Kotlin»

Змінні в Kotlin можуть бути доступні лише для читання, оголошені за допомогою ключового слова val або змінені, оголошені за допомогою ключового слова var. Члени стандартного класу є загальнодоступними, а самі стандартні класи є остаточними, що означає, що створення похідного класу вимкнено, якщо базовий клас не оголошено за допомогою ключового слова open. На додаток до класів і функцій-членів (які відповідають методам) об'єктно-орієнтованого програмування, Kotlin також підтримує процедурне програмування з функціями. Функції та конструктори Kotlin підтримують стандартні аргументи, списки аргументів змінної довжини, іменовані аргументи та унікальні перевантаження сигнатур. Функції-члени класу є віртуальними, тобто вони відправляються на основі типу часу виконання об’єкта, для якого вони викликані. Код, написаний на Kotlin простіше, а значить писати його швидше. Це дає можливість заощадити час, і завершити проект швидше без втрати якості.

## 1.4 Вибір середовища розробки

Середовище розробки програмного забезпечення (IDE) — це певне середовище, яке автоматизує або збільшує робочий процес, залучений до процесу розробки програмного забезпечення. Це включає програмування для багатьох завдань, таких як керування групами та проектами, а також великих завдань програмування, таких як керування конфігурацією. IDE також підтримує обширне та довгострокове обслуговування програмного забезпечення.

У міру розвитку технологій і зростання очікувань користувачів, збільшується і функціональність середовища. Наступні чотири категорії представляють тенденції, які мають значний вплив на середовища, тобто їхні інтерфейси, інструменти та архітектуру:

Мови, орієнтовані на мову: ці типи середовища розробляються навколо однієї мови і, таким чином, надають набір інструментів, відповідних цій мові. Вони дуже інтерактивні та пропонують обмежену підтримку для складного програмування.

Структуроване середовище: ці типи середовищ включають методи, які дозволяють користувачам таємно керувати структурами. Ці методи не усувають уявлення, що є концепцією генератора середнього діапазону.

Середовища з набором інструментів: ці типи середовищ надають набір інструментів, які включають незалежну від мови підтримку програмування для важливих завдань, таких як керування версіями та керування конфігурацією.

На основі методів: ці типи середовищ включають підтримку широкого спектру методів, задіяних у процесі розробки програмного забезпечення. Сюди входять такі завдання, як управління командою та проектом. Вони також включають інструменти для конкретних специфікацій та методів проектування.

### 1.4.1 Середовище Eclipse IDE

Eclipse — безкоштовне інтегроване середовище для розробки модульних

кросплатформних додатків. Розроблено та підтримується Eclipse Foundation.

Найвідоміші програми засновані на платформі Eclipse — різноманітні «Eclipse IDE» для розробки програмного забезпечення на кількох мовах (наприклад, найпопулярніша «Java IDE», яка спочатку підтримувалася, не покладається на закриті розширення, використовує відкритий стандартний API для доступу до платформи Eclipse).

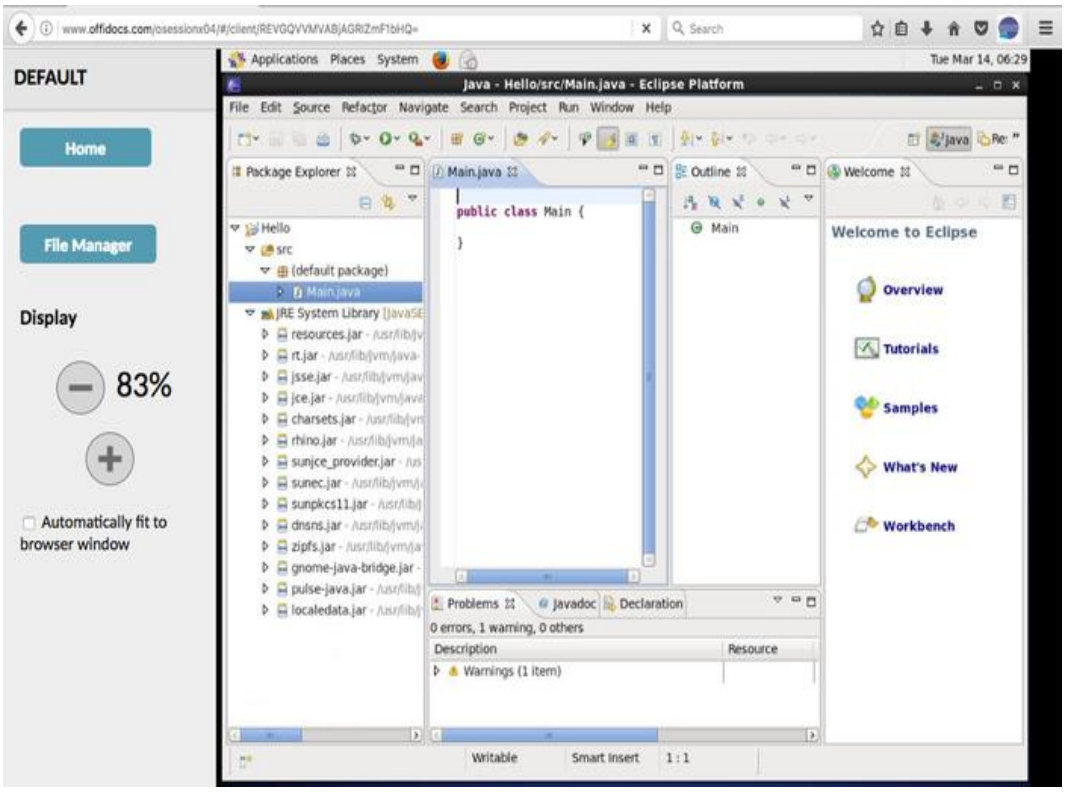


Рисунок 1.7 — «Середовище Eclipse IDE»

Eclipse служить більшою платформою для розробки розширень, оскільки він набув популярності: будь-який розробник може розширити Eclipse за допомогою своїх модулів. Виділіть інструменти Java Development Tools (JDT), C/C++ Development Tools (CDT), розроблені в Bunx qnx спільно з IBM, і m1110 Tools (Hibachi, Hibachi) різними розробниками. Багато розширених середовищ Eclipse доповнюються менеджерами баз даних, серверами додатків тощо.

Eclipse JDT (Java Development Tools) — найвідоміший модуль для групової

розробки: середовище інтегровано з системами контролю версій CVS, GIT в основній поставці, для інших систем (наприклад, Subversion, MS SourceSafe) є плагіни. Також забезпечується підтримка зв'язку між IDE та системою керування завданнями (помилками). Основна поставка включає підтримку відстеження помилок Bugzilla, а також є багато розширень для підтримки інших трекерів (Trac, Jira тощо). Завдяки своїй свободі та високій якості Eclipse є корпоративним стандартом для розробки додатків у багатьох організаціях. Eclipse написаний на Java, тому це продукт, незалежний від платформи, за винятком бібліотеки SWT, яка розробляється для всіх основних платформ (див. нижче). Бібліотека SWT використовується замість стандартної бібліотеки Java Swing. Він повністю покладається на базову платформу (операційну систему), яка забезпечує швидкість і природний вигляд користувальницького інтерфейсу, але іноді викликає проблеми із сумісністю та стабільністю програм на різних платформах.

### 1.4.2 Середовище Android Studio

Android Studio — це інтегроване середовище розробки (IDE) для платформи Android, яке було представлено менеджером з продуктів Google Еллі Пауерс 2013 року на конференції Google I/O. 8 2014 року компанія Google випустила першу стабільну версію Android Studio 1.0

Особливості Android Studio:

— складання додатків, засноване на Gradle;

— різні види збірок та генерація декількох .apk файлів;

— рефакторинг коду;

— шаблони основних макетів та компонентів android;

— підтримка розробки додатків для телевізорів;

— вбудована утиліта для підписування додатків;

— розширений редактор макетів;

— візуалізація графіків;

— швидка інтеграція у хмарі;

— підтримка всіх платформ android.

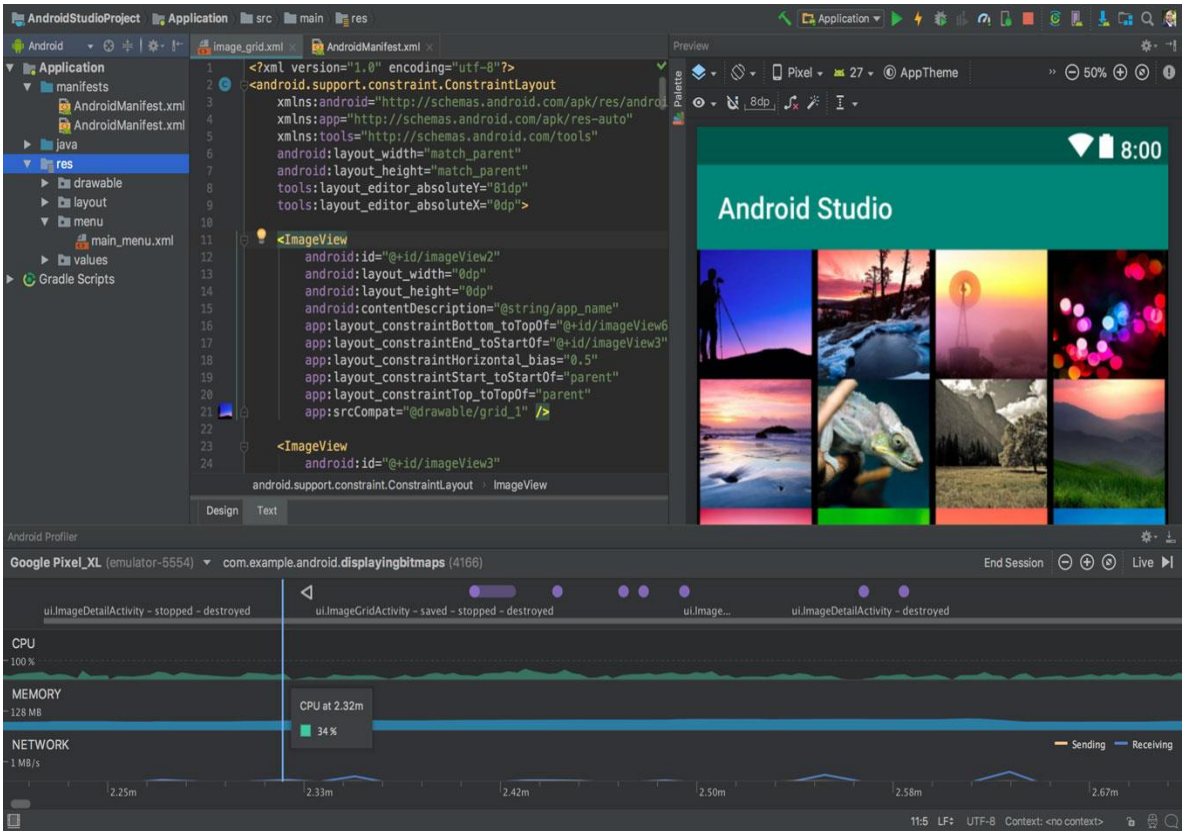


Рисунок 1.8 — «Середовище Android Studio»

Android Studio прийшов на змінний плагін ADT для платформи Eclipse. Середовище створено на основі вихідного коду продукту IntelliJ IDEA Community Edition, що розвивається компанією JetBrains. Android Studio розвивається в рамках відкритої моделі розробки та розширюється під ліцензію Apache 2.0.

Бінарні збірки готуються для Linux (для тестування використовується Ubuntu), macOS та Windows. Середовище надає інструменти для розробки додатків не тільки для смартфонів і планшетів, а й для носимих пристроїв на базі Wear OS, телевізорів, окулярів Google Glass та автомобільних інформаційно-розважальних систем.

Середовище розробки адаптовано для виконання типових завдань, які вирішуються в процесі розробки додатків для платформи Android. Сюди входять інструменти для спрощення тестування програмного забезпечення на сумісність

з різними версіями платформи та інструменти для розробки програм, які працюють на пристроях з різною роздільною здатністю екрана (планшети, смартфони, ноутбуки, годинники, окуляри тощо). На додаток до функцій IntelliJ IDEA, Android Studio має кілька додаткових функцій, таких як: В. Нова уніфікована підсистема для створення, тестування та розгортання додатків, заснована на інструментах складання Gradle і підтримує використання інструментів безперервної інтеграції.

Для прискорення розробки додатків існує набір стандартних елементів інтерфейсу та візуальний редактор для їх компонування, який забезпечує зручний попередній перегляд різних станів інтерфейсу програми (наприклад, можна побачити, як виглядає інтерфейс для різних версій Android та різних розмірів екрана). Для створення нестандартних інтерфейсів існує майстер створення власних елементів дизайну, які підтримують використання шаблонів. Середовище має вбудовані можливості для завантаження типових зразків коду з GitHub.

Платформа Android також включає розширені інструменти рефакторингу, перевірку сумісності з попередніми версіями, проблеми з продуктивністю, моніторинг використання пам’яті та оцінку зручності використання. До редактора додано режим швидкого редагування. Система підсвічування, статичного аналізу та виявлення помилок розширена для підтримки Android API. Інтегрована підтримка оптимізатора коду ProGuard. Інтегровані інструменти для створення цифрових підписів. Надано інтерфейс для керування перекладами на інші мови. В основі робочого процесу Android Studio закладено концепт безперервної інтеграції, що дозволяє відразу виявляти наявні проблеми. Тривала перевірка коду забезпечує можливість ефективного зворотного зв'язку з розробниками. Така опція дозволяє швидше опублікувати версію мобільного додатка в Google Play App Store. Для цього є також підтримка інструментів LINT, Pro-Guard і App Signing. За допомогою засобів оцінки продуктивності визначається стан файлу з пакетом прикладних програм.

# 2 ПРОГРАМНИЙ ЗАСТОСУНОК ДЛЯ ЗАМОВЛЕНЬ ТАКСІ НА ОСНОВІ СТАСТИЧНОЇ ОБРОБКИ ОТРИМАНИХ ДАННИХ

## 2.1 Ціль та задачі розроблення додатка

Мобільний додаток Smart Taxi App вирішує проблему конкурентів застосунків, які не дають користувачам широкої кастомізації своїх подорожей за допомогою таксі. Вирішення даної проблеми, дозволить користувачам налаштовувати свої подорожі так, як вони захочуть, щоб отримати максимальний комфорт, при переміщенні з точки А до точки Б.

Першим кроком у проектуванні мобільного додатка є визначення мети та цілей, яких він має досягнути. Мобільний застосунок, створений в рамках даного проекту, має надавати користувачеві найкращі методи кастомізації та взаємодії з системою підбору поїздок, за допомогою сервісу таксі.

Щоб досягти поставлених цілей, потрібно розробити мобільний додаток, який буде підбирати та налаштовувати процес подорожі сервісом таксі, за допомогою даних, які користувач введе в меню налаштувань

Далі, потрібно розділити процес розробки застосунку на декілька компонентів та декілька етапів.

Серед компонентів можна виділити:

— клієнт (мобільний додаток), який дозволить кінцевому користувачеві отримати доступ до організації подорожей;

— сервер це компонент, який є відповідальним за організацію роботи нашого додатка на стороні сервера;

— база даних. Компонент, який відповідає за зберігання та взаємодію з даними, які необхідні нашому додаткові;

Реалізацію цих компонентів, ми можемо розділити на декілька етапів:

— розробка варіантів використання;

— прототипування та розробка інтерфейсу користувача;

— розробка серверної частни;

— розробка клієнтської частини застосунку;

— прототипування структури бази даних;

— налагодження взаємодії між компонентами.

Між всіма створеними компонентами нашого застосунку, потрібно налаштувати чітку взаємодію, яка дозволить без затримок отримувати потрібні дані та робити потрібні розрахунки. Усі варіанти взаємодій, ми можемо розділити наступним чином:

— клієнт – користувач;

— клієнт – сервер;

— клієнт – БД.

Для кожного варіанта взаємодії, ми повинні створити прозорий інтерфейс з обох боків взаємодії.

### 2.1.1 Початкові дані розробки додатка

Кожен компонент нашої системи, має свій особливий набір обов’язків, також нічого не знає про реалізацію інших розроблених компонентів, і забезпечує прозорий інтерфейс для взаємодії.

Клієнт та сервер, будуть реалізовані в середовищі Android Studio.

Робочий процес клієнта, можна розділити на наступні елементи:

— робота з БД;

— робота з сервером.

Роботу сервера нашого мобільного додатка, можна описати так:

— отримання та опрацювання введених користувачем даних;

Останнім компонентом нашого застосунку, є БД.

### 2.2.2 Побудова діаграм використання додатка

Діаграма варіантів використання — це динамічна діаграма або діаграма поведінки в UML. Діаграми варіантів використання моделюють функціональність

системи за допомогою дійових осіб і варіантів використання. Варіанти використання — це набір дій, послуг і функцій, які повинна виконувати система. У цьому контексті "система" — це те, що розробляється або експлуатується, наприклад, веб-сайт. Дійові особи " — це люди або організації, що виконують певні ролі в системі.

Діаграми варіантів використання цінні для візуалізації функціональних вимог до системи, які будуть втілені у виборі дизайну і пріоритетів розробки.

Вони також допомагають визначити будь-які внутрішні і зовнішні фактори, які можуть вплинути на систему і повинні бути прийняті до уваги.

Вони забезпечують хороший високорівневий аналіз з боку системи. Діаграми варіантів використання визначають, як система взаємодіє з учасниками, не піклуючись про деталі того, як ця функціональність реалізується.

Діаграма варіантів використання системи, включає в себе кількох акторів, серед яких: клієнт, сервер, база даних, користувач. Кожен актор має свої функції.

Користувач мобільного застосунку має наступні функції (рис 2.1):

— введення даних про власне місцезнаходження;

— введення даних про кінцевий маршрут подорожі;

— додавання улюблених місць на мапі;

— налаштування додатку;

— реєстрація;

— вхід.

Клієнт має такі функції (рис 2.2):

— відображення інтерфейсу;

— отримання параметрів;

— комунікація з сервером;

— комунікація з БД.

Сервер має наступні функції (рис 2.3):

— комунікація з клієнтом;

— розрахунок оптимальних подорожей;

— підбір водія та автомобіля згідно заданих параметрів;

База даних має наступні функції:

— зберігання даних.

(рис. 2.1 Діаграма варіантів використання користувача )

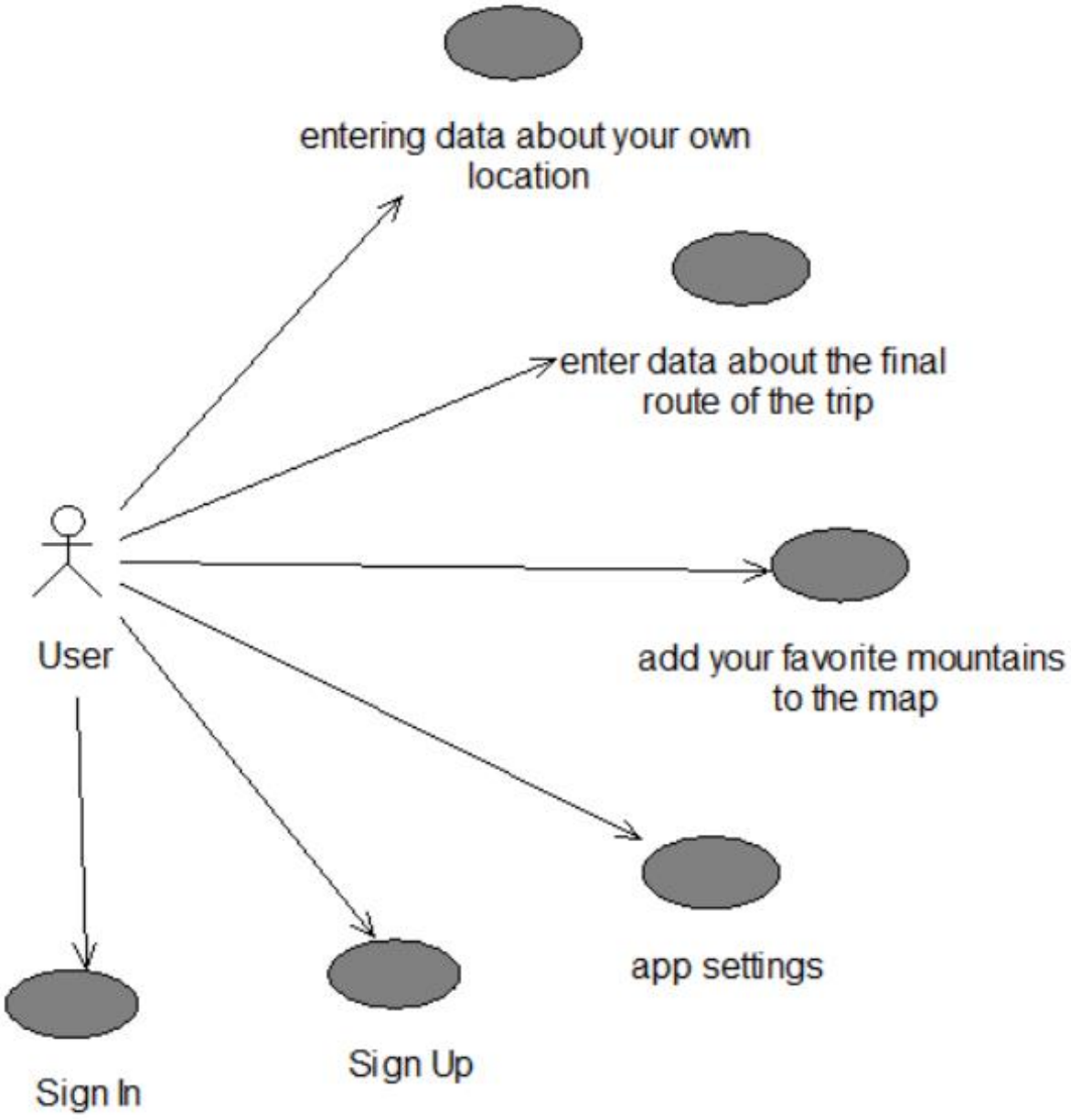


Рисунок 2.1 — Діаграма варіантів використання користувача

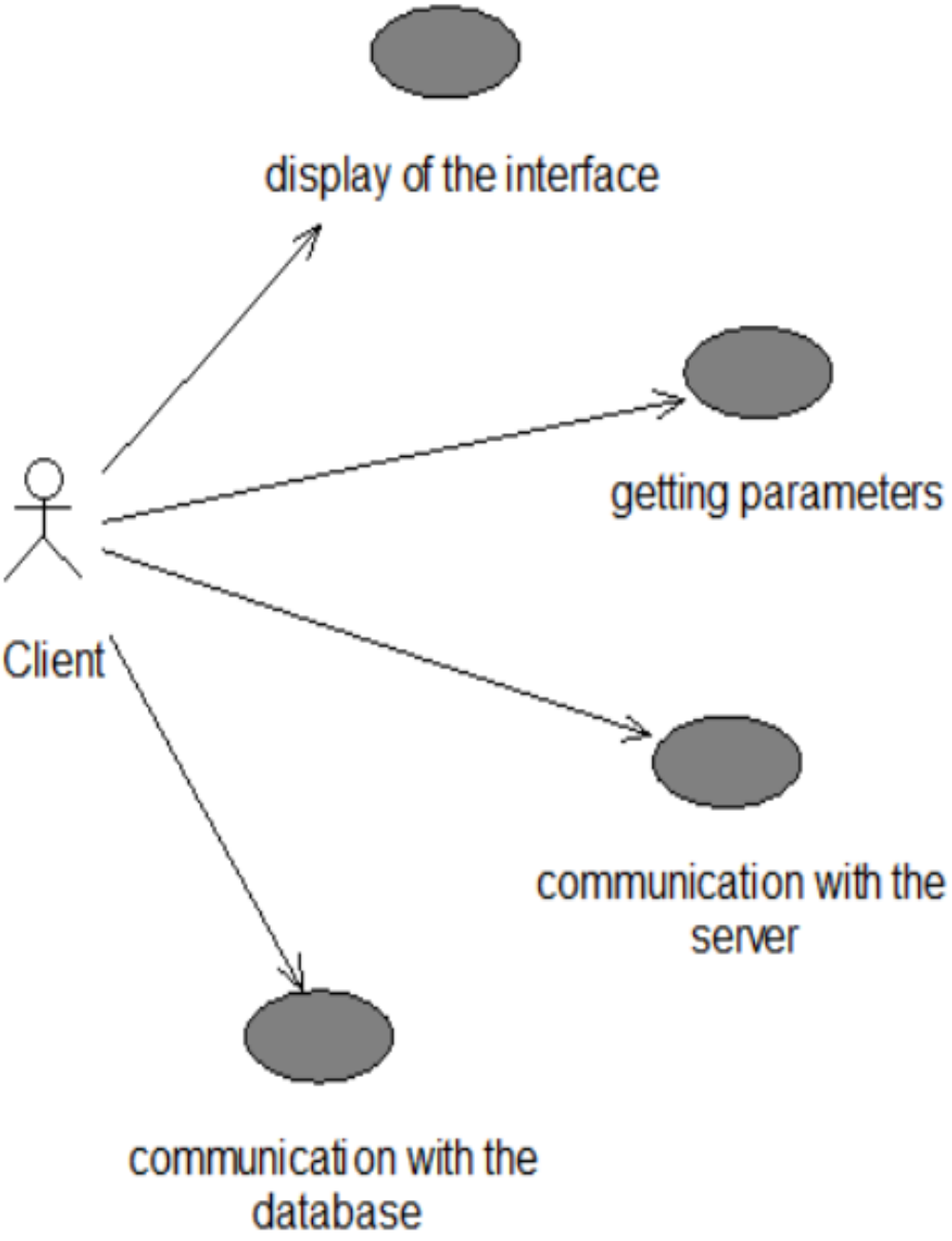


Рисунок 2.2 — Діаграма варіантів використання клієнта

Клієнт взаємодіє з:

— інтерфейсом;

— наданими параметрами;

— сервером;

— базою даних.

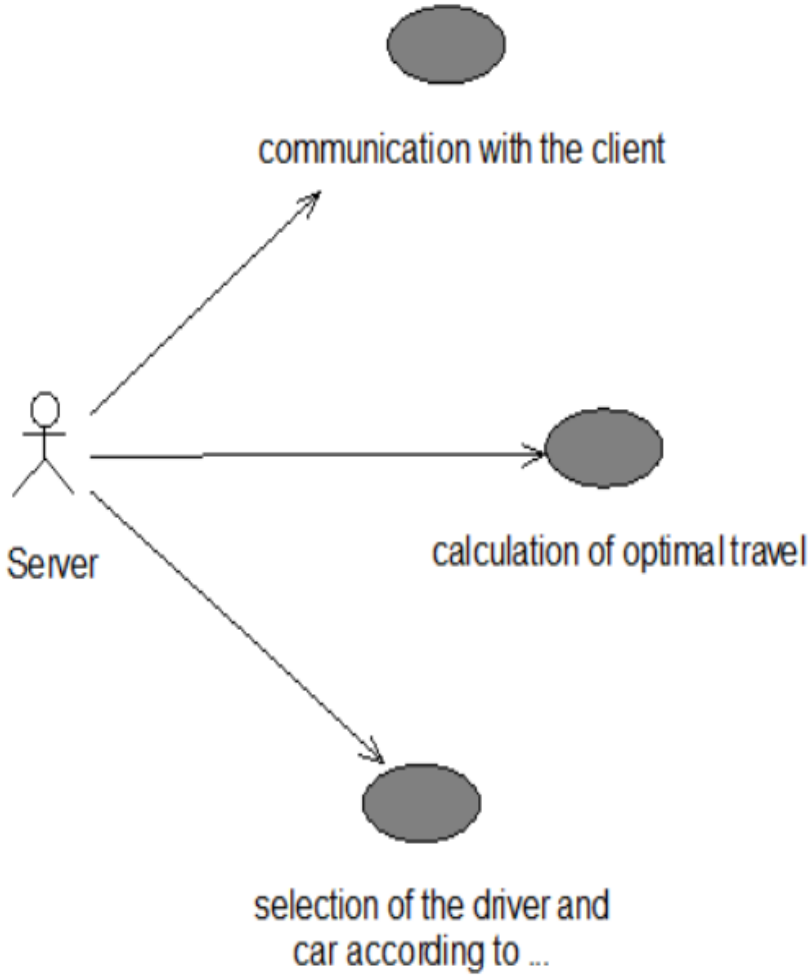


Рисунок 2.3 — Діаграма варіантів використання сервера

Роль сервера — це набір програм, які при правильному встановленні та налаштуванні дозволяють комп'ютеру виконувати певну функцію для декількох користувачів або інших комп'ютерів у мережі. У загальних випадках все ролі мають такі характеристики. Вони визначають основну функцію, призначення або мета використання комп'ютера. Можна призначити комп'ютер для виконання однієї ролі, яка інтенсивно використовується на підприємстві, або для виконання декількох ролей, якщо кожна з них застосовується лише зрідка. Ролі надають користувачам доступ до ресурсів, які управляються іншими комп'ютерами, таким як веб-сайти, принтери або файли.

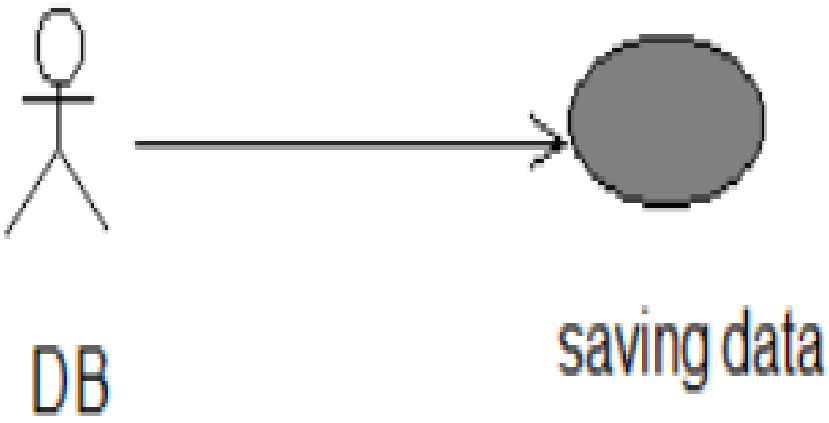


Рисунок 2.4 — Діаграма варіантів використання БД

2.2.3 Функціональні вимоги до додатка

Функціональні вимоги до програмного забезпечення наступні:

1) користувач застосунку повинен мати наступні можливості;

— введення даних про власне місцезнаходження;

— введення даних про кінцевий маршрут;

— додавання улюблених місць на мапі;

— налаштування додатка;

— зареєструвати обліковий запис;

— ввійти до облікового запису.

2) клієнт повинен мати наступні можливості:

— відображати інтерфейс;

— отримувати параметри;

— комунікувати з сервером;

— комунікувати з БД.

3) сервер повинен мати наступні функції:

— комунікувати з клієнтом;

— розраховувати оптимальні подорожі;

— підбирати водія та автомобіль згідно заданих параметрів.

### 2.2.4 Нефункціональні вимоги до додатка

До нефункціональних вимог відносяться мінімальні системні вимоги до роботи програми на стороні клієнта(табл. 2.1) та на стороні сервера (табл. 2.2).

Таблиця 2.1 —Конфігурація ПК для роботи програми на стороні клієнта

| | |
|---|---|
| Операційна система | Android 9 (SDK 29) |
| Процесор | Qualcomm Snapdragon 450 або краще |
| Об'єм оперативної пам'яті | 3 ГБ |

Таблиця 2.2 —Конфігурація ПК для роботи програми на стороні сервера

| | |
|---|---|
| Операційна система | Debian Buster, Fedora, Ubuntu |
| Інтернет канал | Не нижче 300 мегабіт |
| Процесор | Intel Xeon E3-4220 або краще |
| Об'єм оперативної пам'яті | 16 ГБ |

### 2.2.5 Специфікації варіантів використання додатка

Для опису функціональності та поведінки системи використовуються скрипти. Скрипти є важливим механізмом специфікації вимог і можуть бути використані для створення цілей тесту на рівні аналізу вимог. Використовуючи скрипти в системному тестуванні, ми можемо почати тестування на дуже ранній стадії розробки програмного забезпечення. Усунення помилок моделі перед кодуванням і створення тестових випадків призводить до значної економії коштів і вищої якості коду, оскільки помилки, виявлені після факту, дорожчі з точки зору зусиль і часу.

Існуючі підходи до тестування системи використовують сценарій

використання, який включає функціональні деталі, які здаються складними на початку розробки програмного забезпечення. Це дослідження пропонує підхід до тестування системи безпосередньо зі специфікації без урахування функціональних деталей. Попередні та післяумови для сценаріїв використання використовуються як захист, що дозволяє створювати формалізовані тестові випадки; Також додаються контракти на кожному рівні, що дозволяє створювати тестові випадки для будь-якого системного потоку.

Специфікації варіантів використання у вигляді таблиць з описом прецедентів і сценаріїв наведені в таблицях 2.3 - 2.5.

Таблиця 2.3 — Сценарій введення місцезнаходження користувача

| | |
|---|---|
| Ім'я | user_location_input |
| Назва | Введення місцезнаходження користувача |
| Опис | Введення місцезнаходження користувача для подальшого підбору подорожі |
| Передумова | Додаток запущено, обрана вкладка введення локації |
| Постумова | Додаток приймає локацію користувача |
| Основний потік | 1 Користувач заходить до головного меню.<br>2 Користувач вводить місцезнаходження. |
| Альтернативний потік | 1 Користувач заходить до головного меню.<br>2 Користувач вводить місцезнаходження не вірно.<br>3 Користувачеві виводиться попередження про те, що необхідно ввести правильне місцезнаходження. |

Таблиця 2.4 — Сценарій перегляду історії поїздок

| | |
|---|---|
| Ім'я | view_trips_history |
| Назва | Перегляд історії поїздок |
| Опис | Перегляд історії поїздок користувача, за весь час використання додатка. |

| | |
|---|---|
| Передумова | Додаток запущено, обрана вкладка перегляду історії поїздок. |
| Основний потік | 1 Користувач заходить на вкладку історія поїздок.<br>2 Обирає переглянути історіют |

Таблиця 2.5 — Сценарій реєстрації

| | |
|---|---|
| **Ім'я** | registration. |
| **Назва** | Реєстрація. |
| **Опис** | Реєстрація користувача у системі. |
| **Передумова** | Додаток запущено. |
| **Постумова** | Створено запис користувача у базі даних. |
| **Основний потік** | 1 Користувач відкриває додаток.<br>2 Користувач вводить потрібні дані.<br>3 Користувач натискає кнопку «зареєструватись» |
| **Альтернативний потік** | 1 Користувач відкриває додаток.<br>2 Користувач вводить потрібні дані.<br>3 Користувач отримує повідомлення про те, що введені некоректні дані. |
| **Альтернативний потік** | 1 Користувач відкриває додаток.<br>2 Користувач вводить потрібні дані.<br>3 Користувач отримує повідомлення про те, що користувач з такими даними вже існує. |

2.2.6 Розробка макетів екранних форм додатка

Виходячи з вимог до програмного забезпечення, ми створимо діаграму переходу між сторінками інтерфейсу клієнта. Принципова схема показана на рисунку 2.5

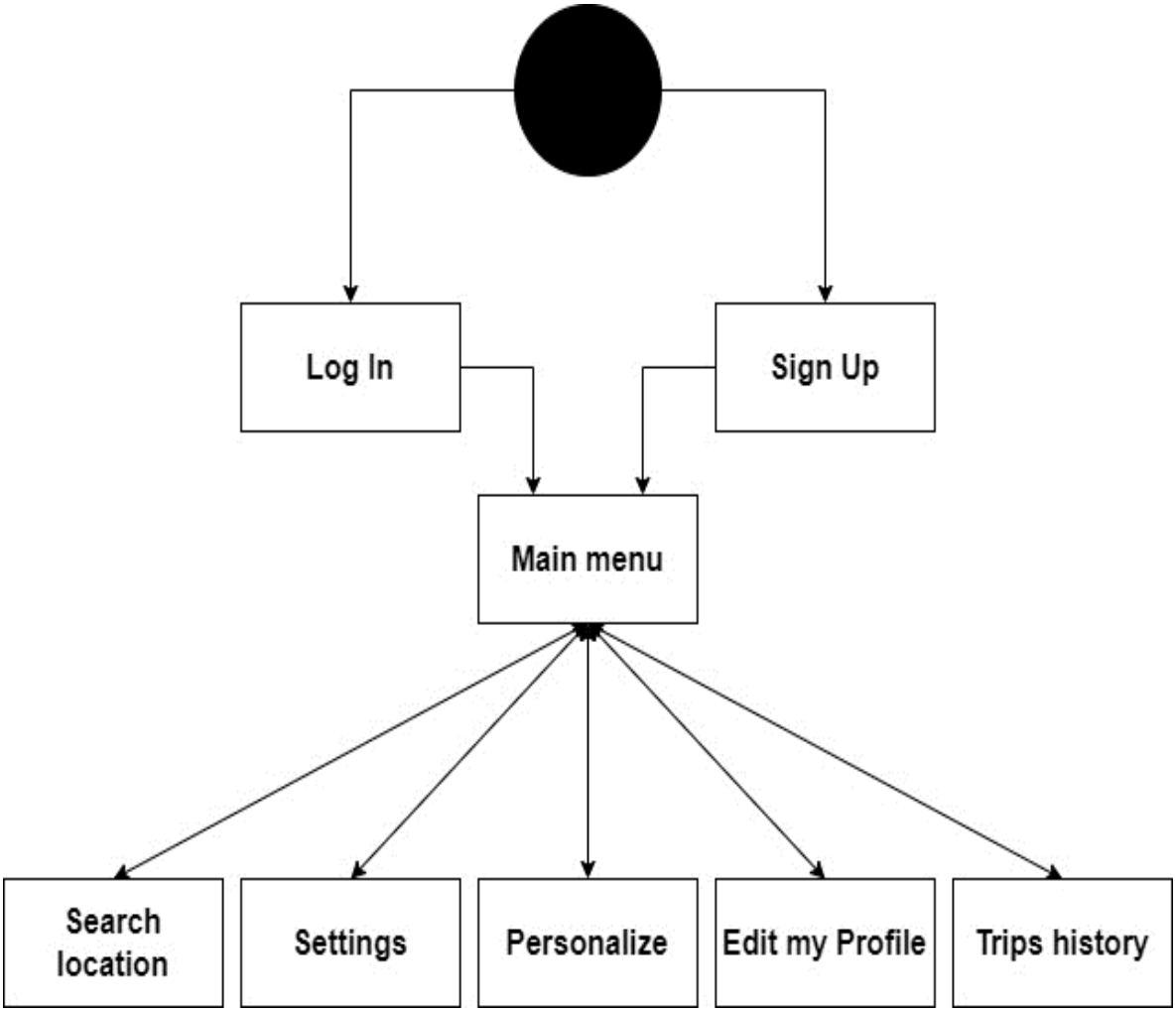


Рисунок 2.5 — Діаграма переходів між сторінками системи

Перша сторінка, яка зустрічає користувача при відкритті додатка, це початкова сторінка. Тут можна ввійти до існуючого облікового запису користувача, або створити новий.

Макет початкової сторінки зображений на рисунку 2.6. Після успішного входу, або авторизації в додатку, користувачеві буде представлено головне меню програми.

Обліковий запис, як правило, містить відомості, необхідні для ідентифікації користувача при підключенні до системи, інформацію для авторизації і обліку. Це ім'я користувача та пароль. Пароль або його аналог, як правило, зберігається в зашифрованому або хешованому вигляді. Обліковий запис може містити також додаткові анкетні дані користувача.

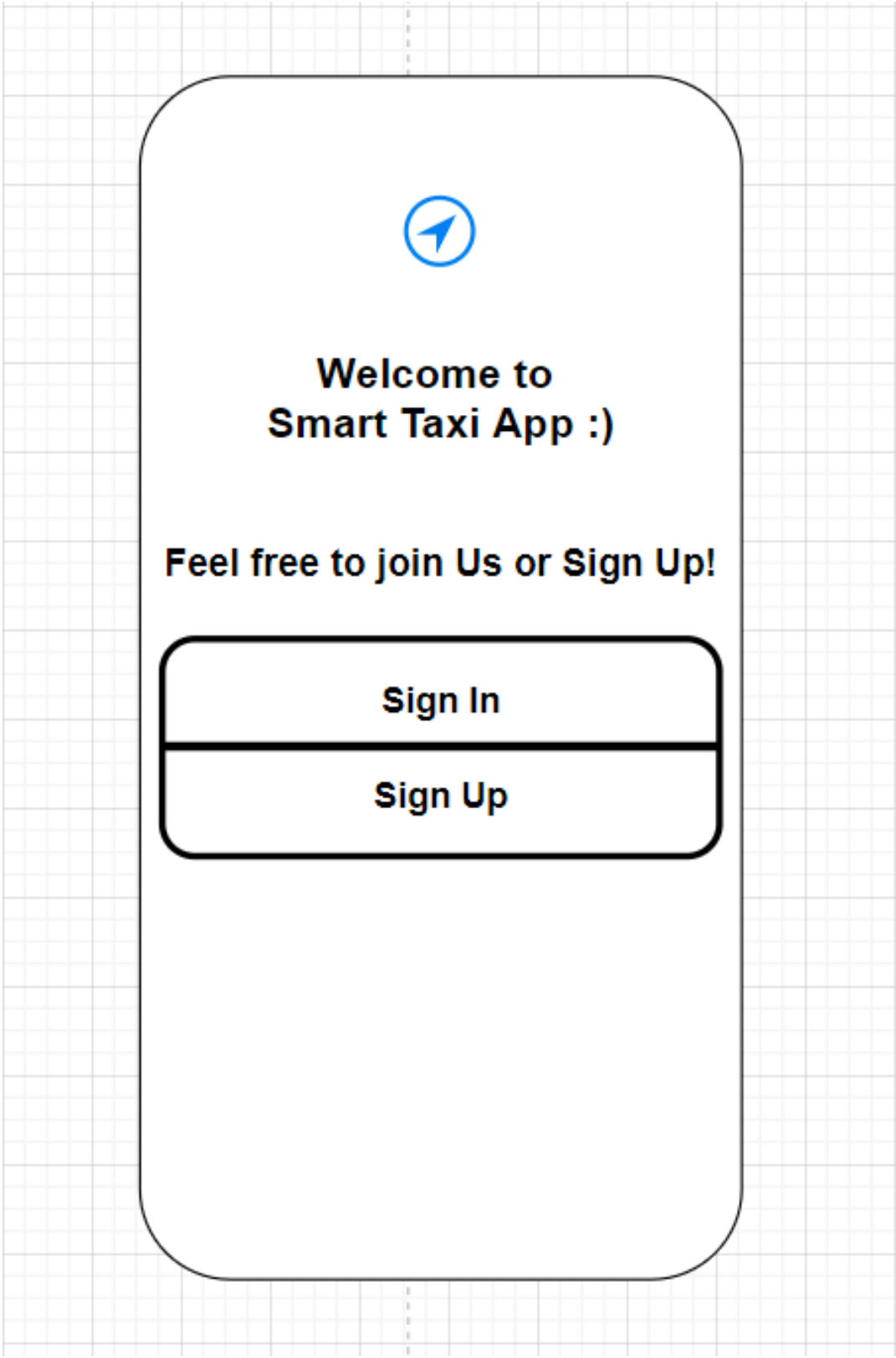


Рисунок 2.6 — Початкова сторінка додатка

Після успішного входу, або авторизації в додатку, користувачеві буде

представлено головне меню програми. На сторінці головного меню, присутні такі елементи:

— верхній бар з назвою програми та елементом “гамбургер” який відкриває доступ до додаткового меню;

— поле введення локації користувача;

— поле введення кінцевого маршруту подорожі;

— кнопка визначення місцезнаходження користувача;

— кнопка пошуку подорожі.

Макет сторінки головного меню зображений на рисунку 2.7

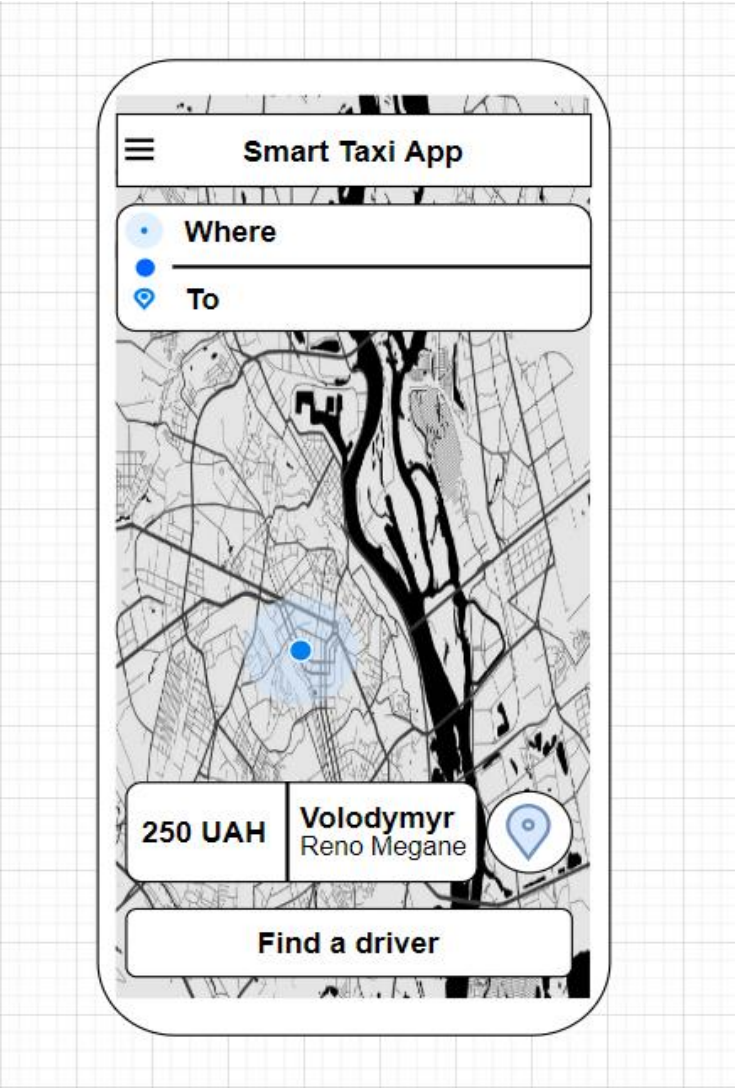


Рисунок 2.7 — Макет сторінки головного меню

Коли користувач натискає на поле введення кінцевого маршруту, відкривається сторінка найближчих до користувача локацій, та поле пошуку потрібної локації. Також, користувач може додавати локації до списку улюблених. Макет сторінки пошуку локацій зображений на рисунку 2.8

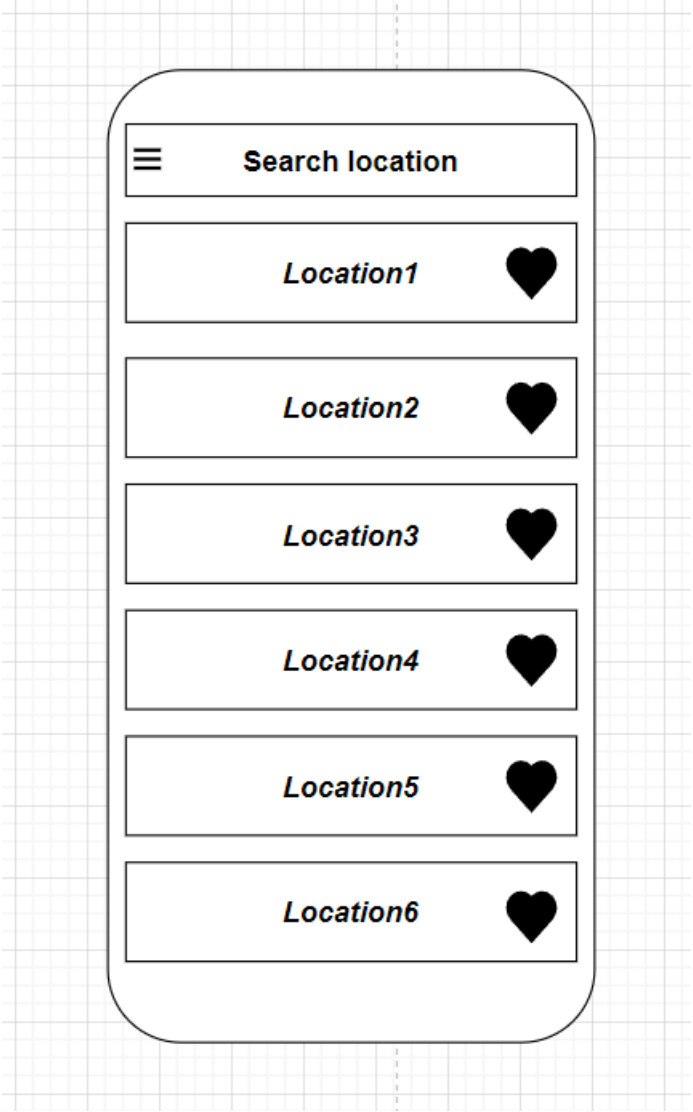


Рисунок 2.8 — Макет сторінки пошуку локацій

Зі сторінки головного меню, користувач може перейти до сторінки налаштувань. Відкривши сторінку налаштувань, користувачеві доступні такі розділи:

— кастомізувати вподобання користувача;

— внести зміни до профіля користувача;

— переглянути історію поїздок;

— переглянути інформацію про додаток;

— оцінити розробників в Google Play.

Макет сторінки налаштувань зображений на рисунку 2.9

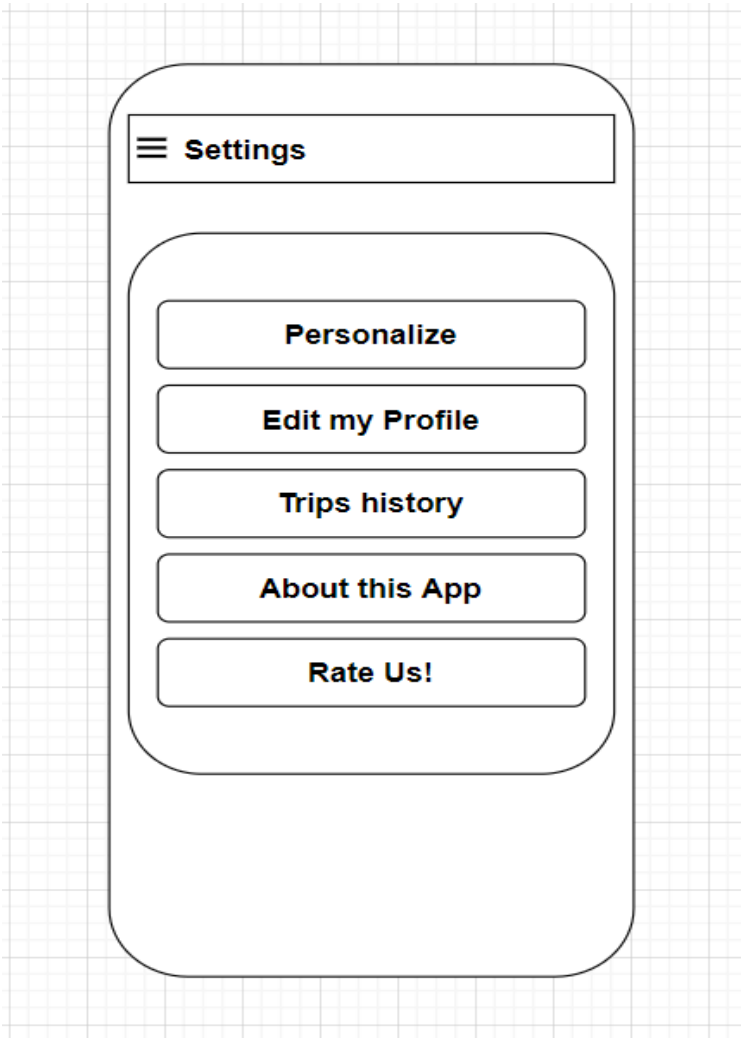


Рисунок 2.9 — Макет сторінки налаштувань

Коли користувач натискає на розділ “Кастомізація”, йому стають доступні такі опції:

— налаштування музичних вподобань в салоні таксі;

— налаштування температурних вподобань в салоні таксі;

— налаштувати список бажаних авто.

Макет сторінки кастомізації зображений на рисунку 2.10

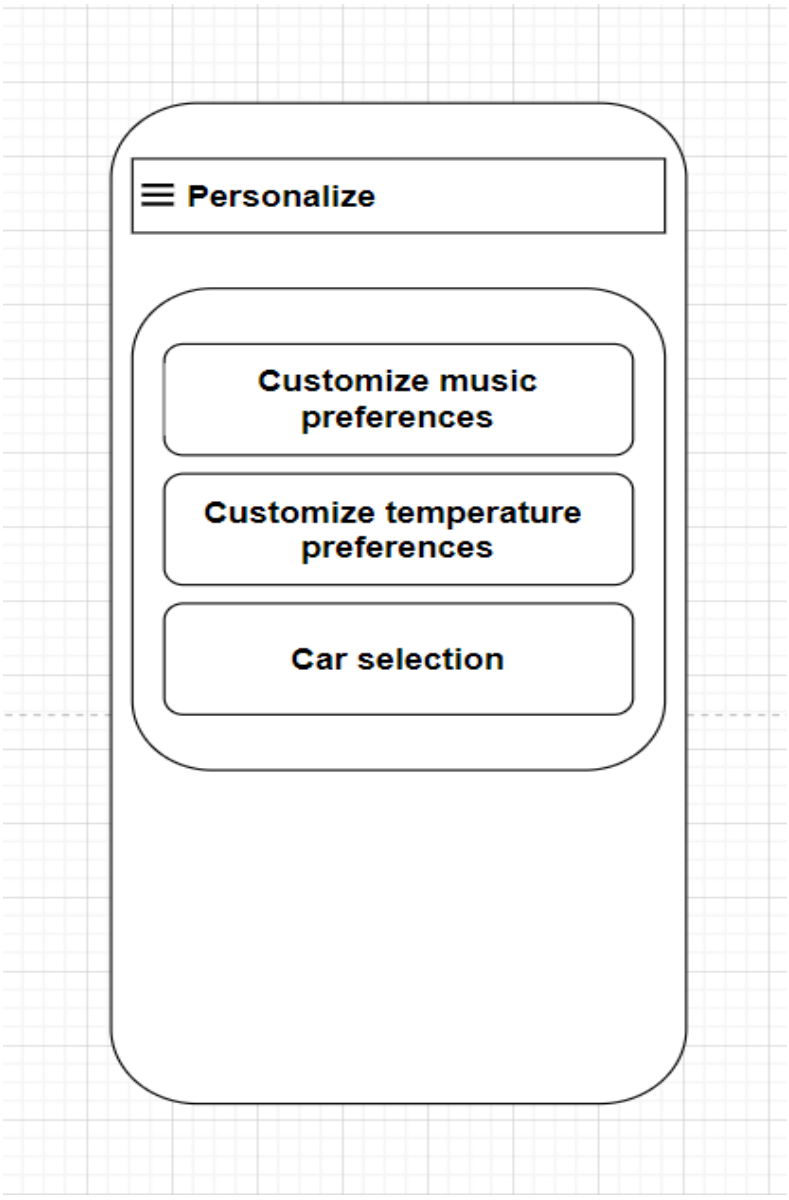


Рисунок 2.20 — Макет сторінки кастомізації

Зі сторінки налаштувань, користувач може перейти до сторінки редагування профіля. Для редагування доступні такі поля як:

— зміна нікнейму користувача;

— зміна фото профіля;

— зміна пароля користувача;

— зміна адреси пошти користувача.

Макет сторінки редагування профіля зображений на рисунку 2.11

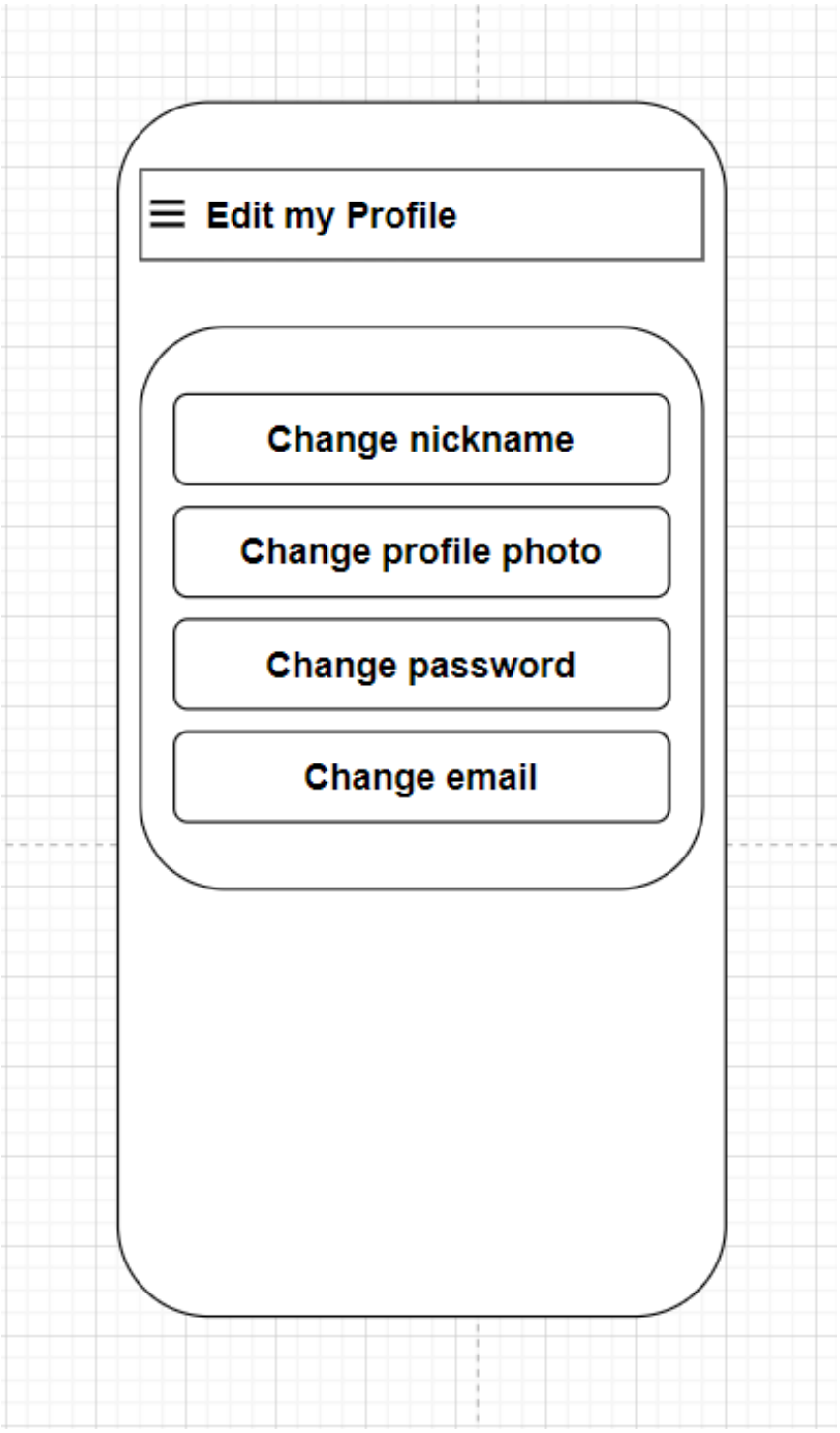


Рисунок 2.31 — Макет сторінки редагування профіля

Також, користувачеві доступне таке меню на сторінці налаштувань, як сторінка перегляду історії поїздок.

На сторінці перегляду історії поїздок користувача, позначаються раніше проведені подорожі з інформацією про локацію, водія, та ціну подорожі. Макет сторінки перегляду історії поїздок зображений на рисунку 2.12

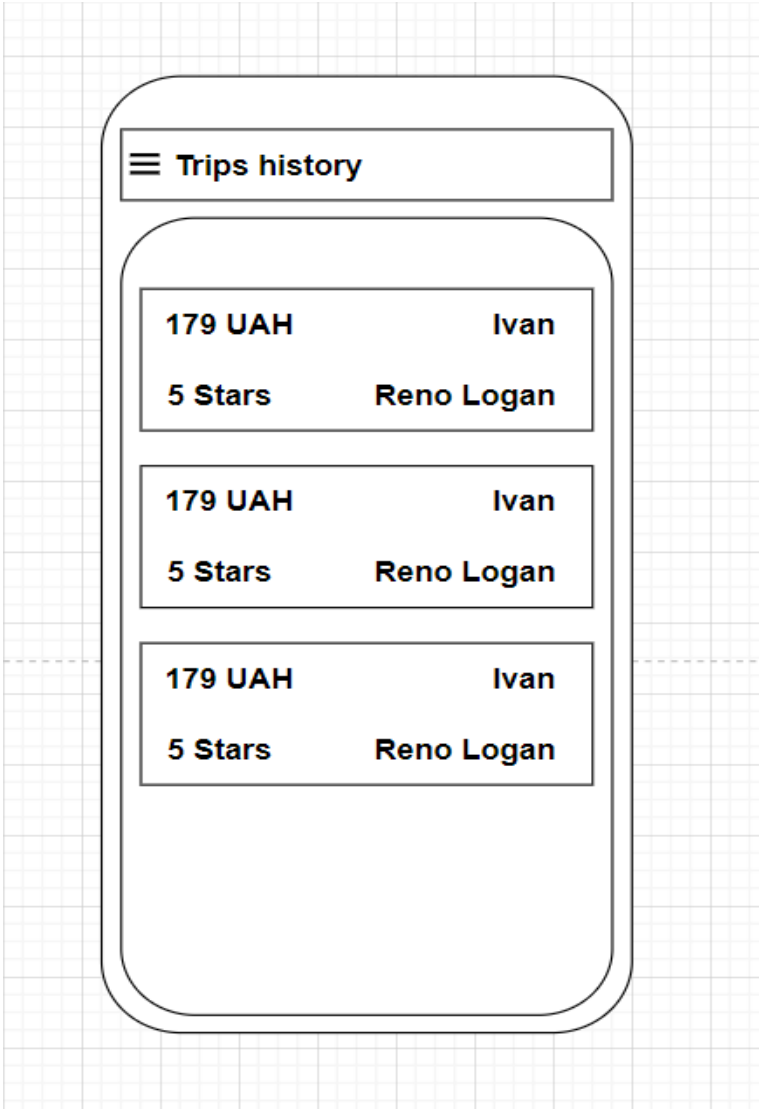


Рисунок 2.42 — Макет сторінки перегляду історії поїздок

### 2.3.7 Опис БД

Додаток використовує вбудовану реляційну базу даних SQLite. SQLite не використовує парадигму клієнт-сервер, тобто двигун SQLite не є окремим процесом, який взаємодіє з програмою, а надає бібліотеку, програма складається з бібліотеки, і двигун стає частиною програми. Тому виклики функцій бібліотеки

SQLite (API) використовуються як протокол обміну. Такий підхід зменшує накладні витрати, час відповіді та спрощує процедури. SQLite зберігає всю базу даних, включаючи визначення, таблиці, індекси та дані, в одному стандартному файлі на комп’ютері, де запускається програма

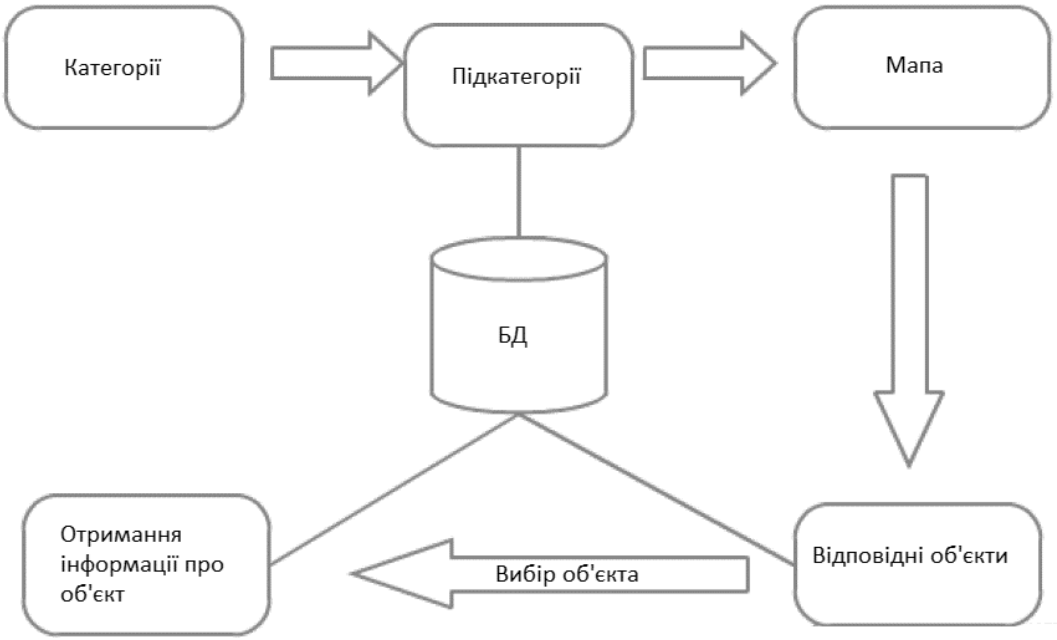


Рисунок 2.53 — Схема розподілу запитів і потоків даних від бази даних до програмного модуля

На рисунку 2.13 показана схема потоку даних. Потік даних на цій діаграмі поширюється з бази даних до інших елементів за допомогою SQL-запитів. При завантаженні програми користувач вибирає категорію. Коли вибрано категорію, меню завантажує дію «категорія», яка отримує доступ до бази даних і завантажує об’єкт на карту. Вкладки дозволяють користувачеві змінювати підкатегорії. Діяльність є текстовим полем, діяльність береться з бази даних і текстове поле заповнюється .

### 2.3.8 Опис структури БД

Діаграма на рисунку 2.14 показує зв’язки та поля існуючої таблиці.

Інформаційна база складається з трьох таблиць PLACE_TYPE (тип місцезнаходження), CONTACT (контакт) і PLACES (розташування). Дані бази даних необхідні для відображення потрібного об’єкта у відповідній підкатегорії на карті та для заповнення інформації про об’єкт.

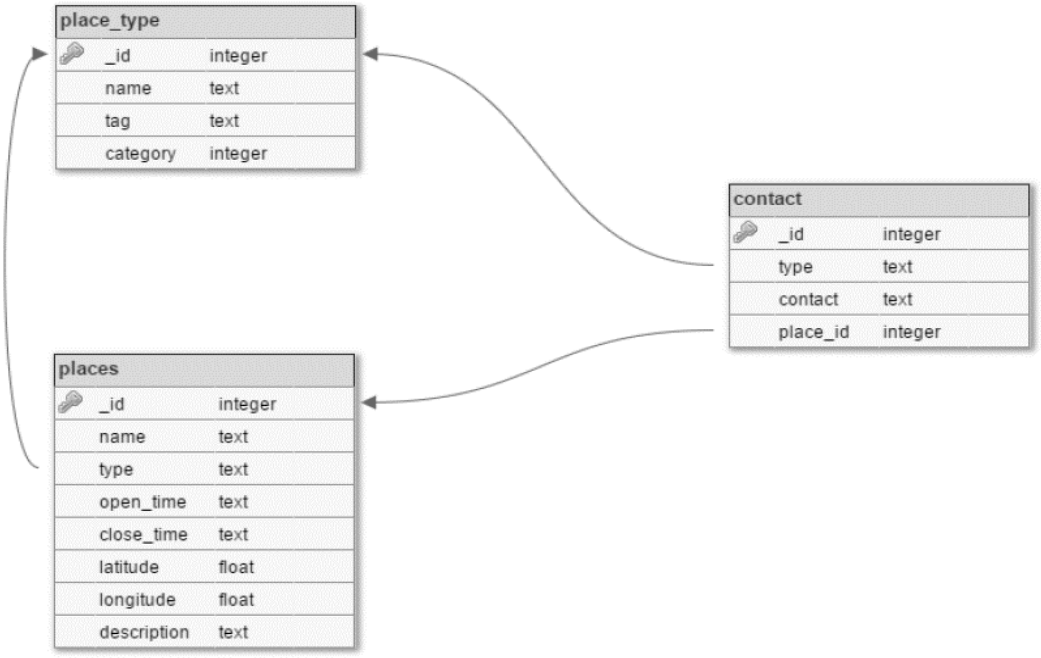


Рисунок 2.64 — Діаграма зв'язків БД

# 3. ДОСЛІДЖЕННЯ ТА АНАЛІЗ РОЗРОБЛЕНОГО МОБІЛЬНОГО ДОДАТКА

## 3.1 Аналіз інтерфейсу користувача

Після детального проектування та розробки макетів екранних форм мобільного додатка для замовлень таксі на основі статистичної обробки отриманих даних, ми розпочали детальну розробку всіх заявлених модулів. Одним з основних елементів розробленого мобільного додатка є інтерфейс користувача.

Інтерфейс користувача — це набір інструментів, які користувач використовує для зв'язку з різними пристроями (комп'ютерами чи побутовими приладами) або іншими складними інструментами (системами). Інтерфейс користувача — це тип інтерфейсу, в якому з одного боку є людина, а з іншого — машина (пристрій, програмне забезпечення). Відповідно до визначення Національного банку стандартизованих науково-технічних термінів, користувальницький інтерфейс — це набір апаратно-програмних засобів, що забезпечує взаємодію користувача з комп'ютером. Інтерфейс мобільного додатка для замовлення таксі на основі статистичної обробки отриманих даних включає:

— засоби відображення інформації, відображувану інформацію, формати і коди;

— командні режими, мову «користувач-інтерфейс»;

— пристрої та технології введення-виведення;

— діалоги, взаємодію та транзакції між користувачем та комп'ютером, зворотній зв'язок з користувачем;

— підтримку прийняття рішень в конкретній предметній області.

Порядок використання програми і документації на неї. Інтерфейс користувача часто розуміють лише як зовнішній вигляд програми. Однак насправді користувач сприймає через нього всю програму в цілому, тобто таке розуміння є надто вузьким.

### 3.1.1 Тестування інтерфейсу мобільного додатка

Для тестування прототипу застосовувалися два підходи: тестування на основі емуляції і на основі реальних пристроїв. Кожен з них слід розглянути окремо.

Тестування на основі емуляції.

Мобільний емулятор — це ресурс для емуляції чи моделювання середовища мобільного пристрою чи смартфона. Це дозволяє розробникам тестувати URL-адреси чи інші технології в операційній системі мобільного пристрою та інтерфейсі відображення.

Цей спосіб тестування передбачає використання емулятора мобільного пристрою, що імітує його поведінку у віртуальній машині. Використовувався емулятор, включений до складу комплекту інструментів для розробника, що додається до мобільної платформи (у складі SDK Android).

Частина корисності мобільного емулятора пов'язана з чуйним дизайном, ідеєю, що веб-сайти, веб-проекти та програмні продукти повинні мати можливість добре працювати на мобільних операційних системах та інтерфейсах. Мобільний емулятор, як правило, фокусується на одній конкретній мобільній платформі, наприклад, Apple iOS або Android.

Деякі мобільні емулятори доступні в Інтернеті. Вводячи URL-адресу та вибираючи пристрій, користувачі можуть отримувати ефективні симуляції через Інтернет. Інші мобільні емулятори продаються поза коробкою або використовуються іншим способом офлайн.

Використання мобільного емулятора — це те, що багато експертів вважають критичним для розвитку. Окрім повномасштабних емуляторів, які використовують вихідний код імітованої операційної системи, деякі інші засоби емуляції пов'язані із кресленнями та схемами для ранніх прототипів. Деякі експерти навіть запропонували використовувати виріз з картону та подачу паперу для імітації прокрутки на мобільному пристрої. Це допомагає розробникам

відчути макет та схеми мобільних продуктів на початку, перш ніж вони насправді розпочнуть кодування проекту. Оскільки інтерфейси мобільних пристроїв займають стільки «частку ринку» в різних частинах індустрії технологій, перенесення продуктів до мобільних пристроїв є головною частиною розвитку і тим, що дуже цінується в технічному співтоваристві.

Спосіб тестування на реальних передбачає використання реального пристрою, під управлінням операційної системи Android з версією не нижче за Android 9.0 Pie. У випадку з мобільними технологіями, особливу увагу потрібно звертати на вимоги по дистрибуції та по підтримці пристроїв. На ринку присутня досить велика кількість різноманітних пристроїв.

Є дві глобальні операційні системи iOS та Android, і різні версії цих систем підтримуються на різних пристроях. Зверніть особливу увагу на вимоги по сумісності: вони можуть як змусити вашу команду підтримувати застарілі пристрої, так і намагатись наздогнати тенденції ринку. Це допоможе зробити процес розробки та тестування більш ефективним. Що ж стосується дистрибуції — шляху передачі продукту до замовника, — то вона має бути заздалегідь чітко визначена і описана у відповідному документі, на кшталт Design Transfer Plan.

Підбір тестових пристроїв це ще одна особливість, яка обов’язково має бути проговорена з замовником. Адже це допоможе вам визначити еталонні пристрої для тестування продукту. При розробці кросс-платформенних застосунків, якщо у вас з самого початку зазначено, що продукт має бути і на iOS, і на Android, варто мати хоча б один фізичний пристрій для тестування для кожної окремої мобільної системи. Тут також варто зазначити, що підбір необхідної версії операційної системи для еталонних пристроїв залежить від вимог до застосунку.

На рисунку 3.1 зображено підготовку до тестування додатка за допомогою вбудованого емулятора пристроїв в Android Studio. Програма є спеціалізованою середовищем розробки Android-ігор і додатків, яка з 2013 року працює на програмному забезпеченні IntelliJ IDEA від компанії JetBrains. Фактично даний софт був створений для розробників, і він є універсальним інструментом для

ентузіастів програмування під платформу Андроїд, які систематично намагаються експериментувати з можливостями ОС.

Також слід зазначити, що з кожною новою версією програми розробники значно розширюють її функціонал. Вже сьогодні, наприклад, софт можна комбінувати з різними програмами. Тобто, при бажанні ви можете інтегрувати інший Android emulator for Android Studio. Що ж стосується інших переваг, якими володіє утиліта, то до них слід віднести чудово реалізовану можливість верстки в режимі реального часу. Розробляючи якийсь Андроїд-софт, вам не доведеться витрачати час на систематичне формування apk-файлу для подальшого запуску його через емулятор. Все відбувається швидко і інтуїтивно, в одному вікні. Крім того, ви по достоїнству оціните різноманіття вікон з різними розмірами. Отже, ви відразу ж зможете стежити за тим, як буде відкриватися ваш софт на різних пристроях. Важливою перевагою можна вважати і вбудовані інструменти для організації монетизації, відстеження ефективності рекламних продуктів і багато іншого.

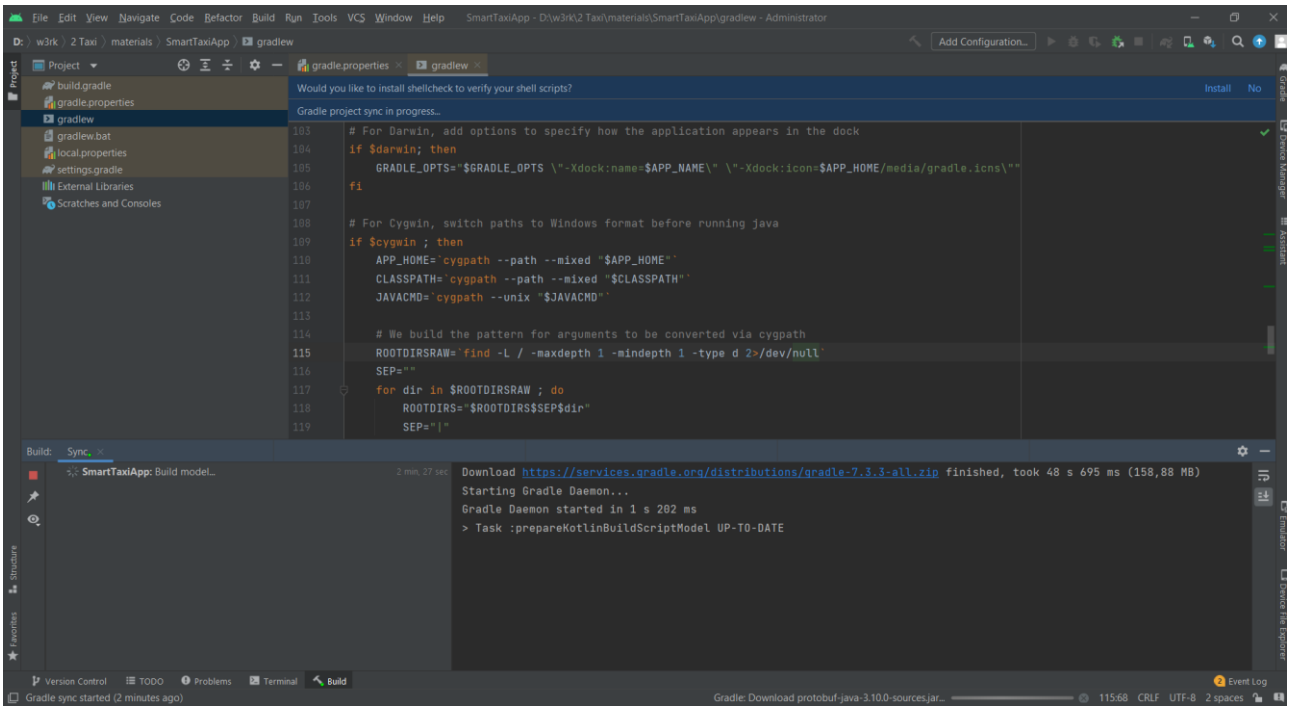


Рисунок 3.1 — Підготовка додатка до тестування

Налаштування з потрібними параметрами емулятора пристроїв Android Studio зображено на рисунку 3.2.

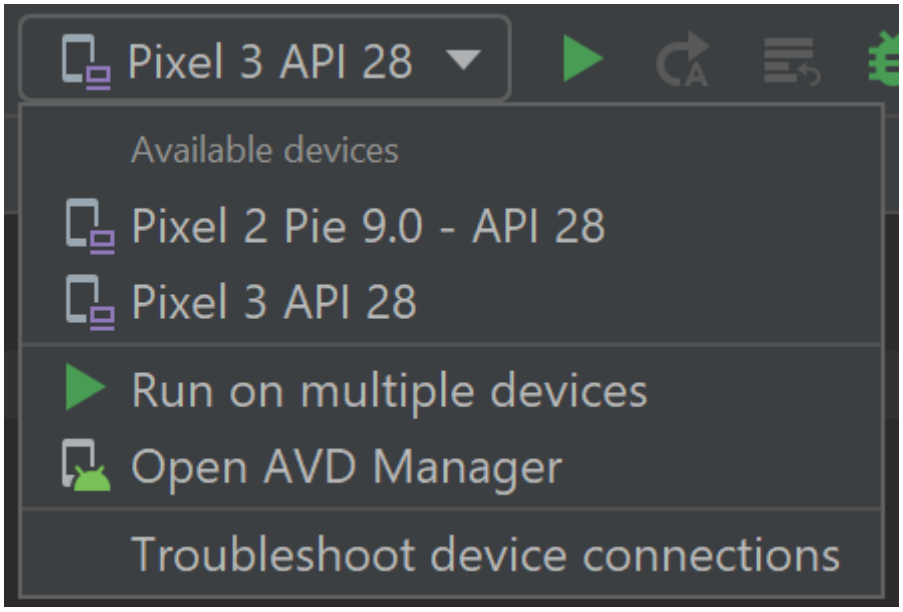


Рисунок 3.2 — Налаштування емулятора пристроїв

Результат тестування додатка зображено на рисунку 3.3.

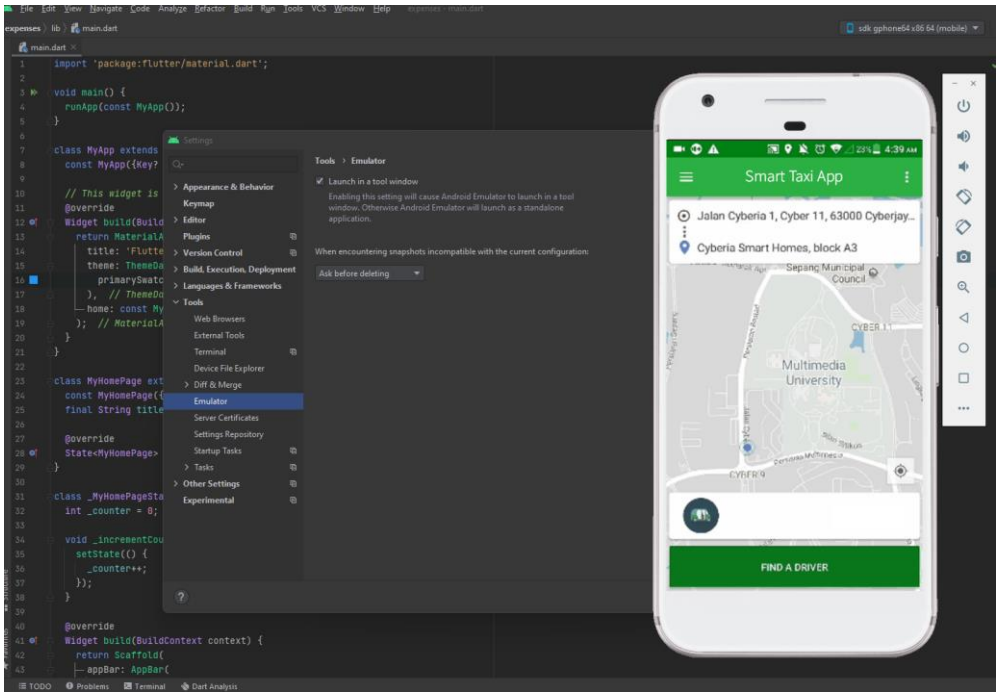


Рисунок 3.3 — Результат тестування в Android Studio Device Emulator

Після успішного тестування мобільного додатка в Android Studio Device Emulator, ми почали тестування на реальному пристрої з такими ж початковими параметрами, як і в нашому пристрої для емуляції.

На рисунку 3.4 зображено головну сторінку мобільного додатка для замовлень таксі на основі статистичної обробки отриманих даних.

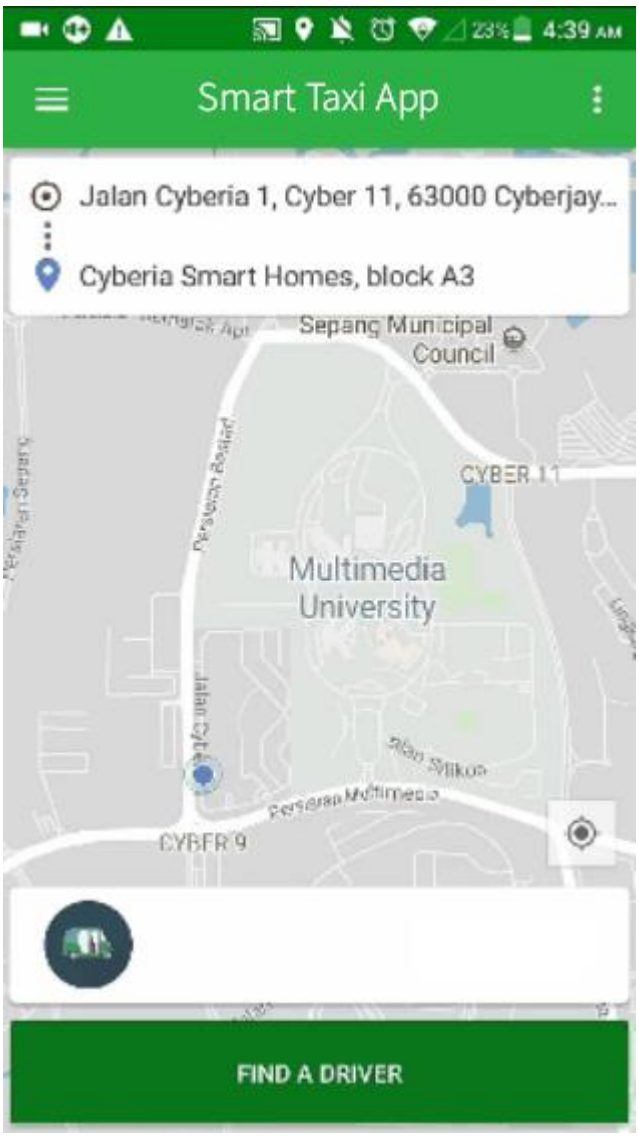


Рисунок 3.4 — Головний екран додатка

На головному екрані додатка знаходиться верхній бар з назвою програми та елементом “гамбургер” який відкриває доступ до додаткового меню. Іконка-гамбургер є класичною. Навіть якщо ви не знайомі з цією її назвою, то три чорні

смуги настільки ж знайомі, як курсор миші — постійний супутник нашого інтернет-серфінгу з першого дня, як ви стали володарем свого комп'ютера. В ході цього дослідження, заміна іконки на слово "Меню" не зробило істотного впливу на поведінку наших користувачів. Звичайно ми не можемо ці дані екстраполювати на все. У якихось країнах, на якихось мовах або пристроях, це можливо спрацювало краще або гірше. В цілому, він був настільки ж впізнаваний, як і слово "Меню".

Поле введення локації користувача. У більшості випадків, введення адрес може бути неприємним і схильним до помилок, особливо в мобільних додатках. У нашому додатку використовується елемент керування введення адреси, щоб полегшити введення адреси. Елемент керування використовує нечітку логіку, щоб запропонувати потенційні збіги під час введення тексту. Виберіть ту, яку потрібно швидко та легко ввести точну адресу. Елемент керування повертає адресу як структуровані дані. Наш додаток може отримувати таку інформацію, як місто, вулиця, муніципалітет і навіть координати широти та довготи. Дані мають формат, зручний для багатьох локальних та міжнародних форматів адрес.

Поле введення кінцевого маршруту подорожі. Ви можете додати кнопку до програми, щоб зберегти введені адреси як збір даних. Потім ви можете отримати адреси та відобразити їх у вікні керування картою. Додайте елемент керування картою та елемент керування введенням адреси до програми. Вставлення та розміщення елемента керування Кнопкою. Змініть властивість OnSelect елемента керування кнопки наступним чином. (Підказка: Скопіюйте формулу та вставте її в рядок формул або на Вкладка "Додаткові властивості", незалежно від того, що ви віддаєте перевагу. Укажіть довготу, широту та радіус (у метрах). Елемент керування почне пошук на широті і довготі, на відстань, зазначену в полі радіуса.

Інформація про водія. Профіль водія містить багато цікавих фактів. Ви можете переглянути не тільки оцінки, значки досягнень і дані про здійснені поїздки, а й позитивні відгуки чи примітки з подякою від інших клієнтів. Знайомство — запорука гарного дня й вдалої поїздки.

Як це працює:

— відкрийте додаток;

— відкрийте додаток і замовте поїздку, як завжди;

— перегляньте профіль водія;

— доповніть профіль водія.

Якщо ви ставите 5 зірок за поїздку, то можете залишити для водія позитивний відгук або примітку з подякою. У такий спосіб ви висловлюєте вдячність, а інші клієнти можуть більше дізнатися про водія.

Коли користувач натискає на поле введення кінцевого маршруту, відкривається меню найближчих до користувача локацій, та поле пошуку потрібної локації. Також, користувач може додавати локації до списку улюблених. Меню найближчих до користувача локацій зображено на рисунку 3.5.

Інтерфейси є основою взаємодії в сучасних інформаційних системах. Якщо інтерфейс якого-небудь об'єкту (комп'ютера, програми, функції) не змінюється (стабільний, стандартизований), це дає можливість модифікувати сам об'єкт, не переробляючи його принципи взаємодії з іншими об'єктами. Це дозволяє масштабувати інтерфейс, створювати інші пристрої чи функції, які будуть використовувати такий спосіб взаємодії. Під інтерфейсом розуміють не тільки пристрої, але й правила (протокол) взаємодії цих пристроїв. У контексті окремого елемента інтерфейс елемента протилежний до реалізації елементу (внутрішнього устрою та функціювання). Користувачеві елемента немає потреби знати, як впроваджено уживаний елемент, щоби керувати ним, але використовуваний елемент має надати інтерфейс керування. З посиленням використання персональних комп'ютерів та відносним зниженням обізнаності (інтересу) суспільства про важкі машини, термін «інтерфейс користувача», здебільшого, передбачає графічний інтерфейс користувача, тоді як промислові панелі керування та механізми контролю за обладнанням, частіше стосуються людино-машинної взаємодії.

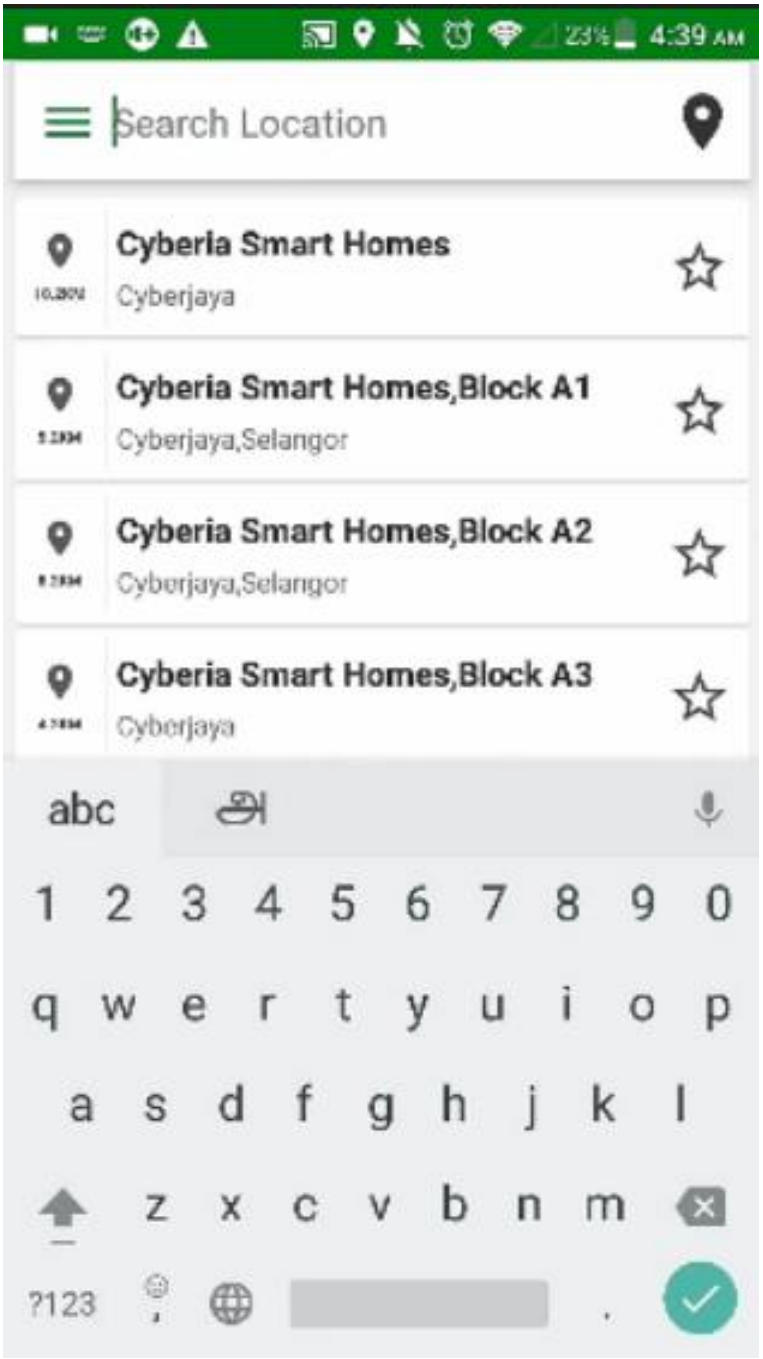


Рисунок 3.5 — Меню локацій

Після того, як користувач обирає кінцевий маршрут подорожі, мобільний додаток починає шукати водія згідно заданих параметрів. Пошук оптимальної подорожі зображено на рисунку 3.6

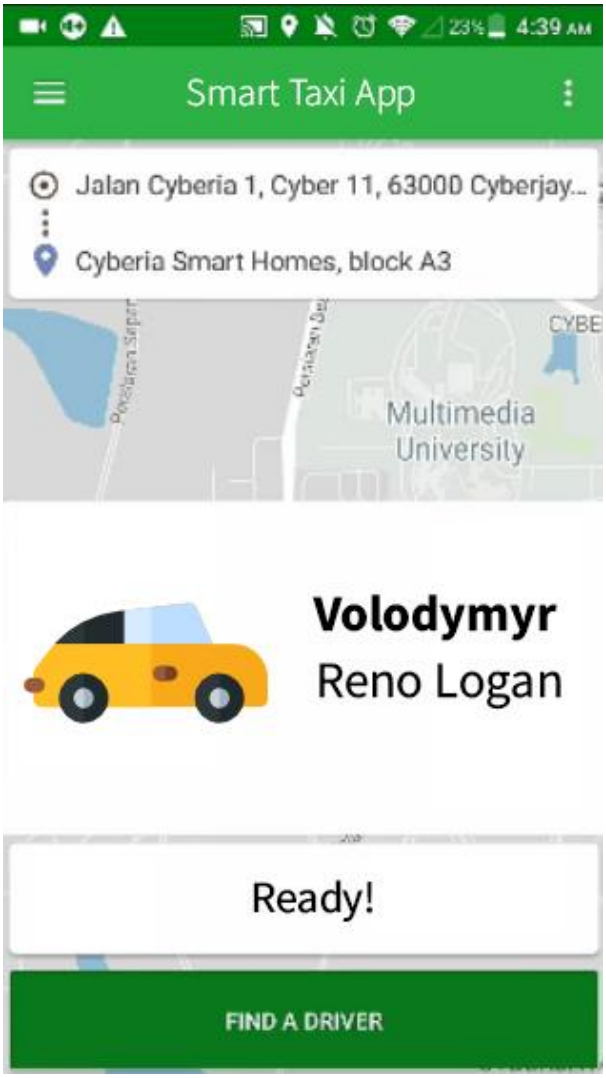


Рисунок 3.6 — Пошук оптимальної подорожі

Додаток веде статистику та дозволяє відслідковувати інтенсивність замовлення в певні проміжки часу

### 3.1.2 Обробка статистичних данних

Для великих компаній статистика передусім так як дозволяє заощадити ресурси та збільшити продуктивність, як приклад в ночі немає потреби в великій кількості працівників так як кількість замовлень стає занадто нижча зекономивши на кількості персоналу для нічних змін можна вийти в непоганий прибуток сам процес ведення статистики значно полегшує роботу бухгалтерії, адже всі данні

одразу доступні в звітах, також отримана статистика активно використовується в програмі для формування ціни на поїздку, залучення знижок та спеціальних пропозицій  також ці данні можливо використовувати для подальшої закупівлі палива,  адже завдяки інтенсивності поїздок можна приблозно порахувати затрати палива що робить внутрішню економіку підприємства більш гнучкою і дозволяє замовляти ресурси зображені на рисунку 3.7

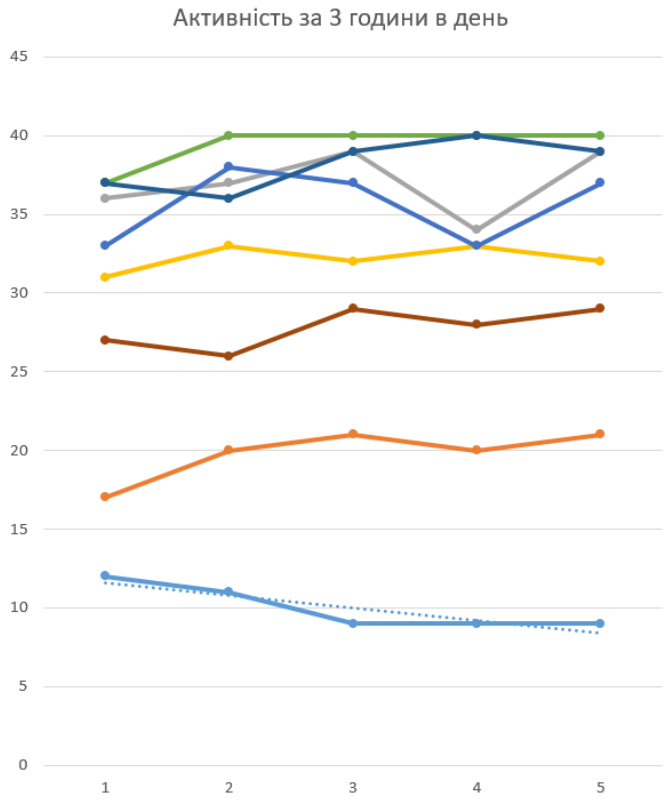


Рисунок 3.7 — Активність за 3 години в день

Популярність замовлень в компанії поступово падає що інформує керівництво про необхідність більш активної реклами послуг, яка зможе підвищити конкернтноспроможність компаніїї на фоні інших сервісів

Керівникам підприємств статистика допомогає у прийнятті рішень. Статистичні дані можуть використовуватися для прогнозування продажів, фінансового аналізу проектів капітальних витрат, побудови прогнозів прибутку для нового продукту, створення виробничих кількостей і проведення аналізу

вибірки для визначення якості продукту. Використання статистики надає реальні дані про складні ситуації. Для складання бізнес-плану, для започаткування бізнесу, як приклад сфера перевезень, для того щоб оцінити даний сегмент ринку, необхідні статистичні дані про кількість пасажирів, які користуються данними послугами, їх географію, кількість організацій, які замовлюють перевезення.

Ключовою особливістю данного додатку являється досить зручний формат зчитування данних який враховує день замовлення, час замовлення та приводить статистку середнього значення замовлень впродовж 3 годин як приклад наведенно нище в таблиці 3.1

Таблиця 3.1 — Статистичні данні

| | Пн | Вт | Сер | Чт | Пт | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1:00:00 | 5 | 6 | 4 | 5 | 4 | | | | | |
| 2:00:00 | 3 | 2 | 3 | 2 | 3 | 12 | 11 | 9 | 9 | 9 |
| 3:00:00 | 4 | 3 | 2 | 2 | 2 | | | | | |
| 4:00:00 | 3 | 4 | 3 | 4 | 3 | | | | | |
| 5:00:00 | 6 | 7 | 8 | 7 | 8 | 17 | 20 | 21 | 20 | 21 |
| 6:00:00 | 8 | 9 | 10 | 9 | 10 | | | | | |
| 7:00:00 | 9 | 10 | 10 | 8 | 10 | | | | | |
| 8:00:00 | 14 | 15 | 15 | 13 | 15 | 36 | 37 | 39 | 34 | 39 |
| 9:00:00 | 13 | 12 | 14 | 13 | 14 | | | | | |
| 10:00:00 | 12 | 11 | 12 | 13 | 12 | | | | | |
| 11:00:00 | 10 | 11 | 12 | 11 | 12 | 31 | 33 | 32 | 33 | 32 |
| 12:00:00 | 9 | 11 | 8 | 9 | 8 | | | | | |
| 13:00:00 | 10 | 13 | 12 | 11 | 12 | | | | | |
| 14:00:00 | 11 | 12 | 13 | 11 | 13 | 33 | 38 | 37 | 33 | 37 |
| 15:00:00 | 12 | 13 | 12 | 11 | 12 | | | | | |
| 16:00:00 | 10 | 11 | 11 | 10 | 11 | | | | | |
| 17:00:00 | 12 | 13 | 14 | 13 | 14 | 37 | 40 | 40 | 40 | 40 |
| 18:00:00 | 15 | 16 | 15 | 17 | 15 | | | | | |
| 19:00:00 | 14 | 13 | 16 | 15 | 16 | | | | | |
| 20:00:00 | 12 | 13 | 13 | 12 | 13 | 37 | 36 | 39 | 40 | 39 |
| 21:00:00 | 11 | 10 | 10 | 13 | 10 | | | | | |
| 22:00:00 | 10 | 9 | 10 | 11 | 10 | | | | | |
| 23:00:00 | 9 | 8 | 9 | 10 | 9 | 27 | 26 | 29 | 28 | 29 |
| 24:00:00 | 8 | 9 | 10 | 7 | 10 | | | | | |

Згідно рис. 3.8 вночі необхідна кількість диспетчерів значно нижча, а в обід навпаки. Це дозволяє більш ефективно складати графік роботи.


Рисунок 3.8 — Активність за день

# ВИСНОВКИ

Під час виконання дипломної роботи було створено програмне забезпечення для замовлень таксі на основі статистичної обробки отриманих даних у вигляді мобільного додатка.

У першому розділі було визначено концепцію програмного забезпечення для замовлень таксі на основі статистичної обробки отриманих даних. З огляду на популярність та попит мобільних додатків, було обрано реалізацію ПЗ саме у такому вигляді. Після аналізу існуючих аналогів мобільних додатків для замовлень таксі(«Uber», «Lyft», «Bolt»), було визначено їх переваги та недоліки. Серед недоліків виявлено відсутність статистичної обробки отриманих даних після отримання замовлення на поїздку для таксі. Тому, основну увагу при розробці мобільного додатку у наступних розділах буде присвячено й цьому питанню зокрема. Також було розглянуто інструменти та структуру мобільного застосунку для замовлень таксі на основі статистичної обробки отриманих даних. Мовою програмування обрано Java, середовищем розробки Android Studio. Тож, маємо все необхідне для розробки мобільного застосунку для замовлень таксі на основі статистичної обробки отриманих даних, про яку детально буде описано в третьому розділі.

У другому розділі були визначені ціль та задачі розроблення додатка. Проаналізовано вимоги до додатка. Визначено початкові дані розробки додатка, побудова діаграм використання додатка, визначено функціональні вимоги до додатка, нефункціональні вимоги до додатка. Розроблено специфікації варіантів використання додатка та макети екранних форм додатка. Зроблено архітектурне проектування додатка Розроблено та протестовано інтерфейс додатка для замовлень таксі на основі статистичної обробки отриманих даних, про яку детально буде описано в третьому розділі. Слід також зазначити, що під час розробки застосунку були враховані всі зазначені на початку вимоги, яким він мав відповідати.

У третьому розділі, було проведено тестування мобільного додатка для замовлень таксі на основі статистичної обробки отриманих даних за допомогою Android Studio Virtual Device, та реального пристрою під керуванням Android 9 Pie. Був продемонстрований та описаний інтерфейс додатка. Додаток знаходиться на pre-alpha стадії, тому заявлена кількість функцій є обмеженою, а сам додаток поводить себе достатньо не стабільно. Пропонуємо провести подальшу розробку функцій в майбутньому.

В підсумку було вирішенно поставлені задачі, а саме:

— описано концепцію програмного забезпечення;

— проаналізовано аналоги програмного забезпечення;

— порівняно аналоги програмного забезпечнення;

— обрано мову програмування;

— обрано середовище розробки;

— описана структура програмного забезпечення;

— розроблено програмне забезпечення для замовлень таксі;

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. International Energy Agency. Indicators of Energy Use and Efficiency — Understanding the Link Between Energy Use and Human Activity. International Energy Agency: Paris, 1997.

2. Palacio Ld. White paper—European transport policy for 2010: time to decide. In European Commission, 2001.

3. Shaheen SA, Sperling D, Wagner C. Carsharing and partnership management: an international perspective. Transportation Research Record: Journal of the Transportation Research Board 1999; 1666:118—124.

4. Kirby R, Bhat K. Para-transit: Neglected Options for Urban Mobility. Urban Institute: Washington D.C, 1974.

5. Vuchic V. Urban Transit Systems and Technology. John Wiley & Sons: New Jersey, USA, 2007.

6. Jeon CM, Amekudzi AA, Vanegas J. Transportation system sustainability issues in high-, middle-, and low-income economies: case studies from Georgia (US), South Korea, Colombia, and Ghana. Journal of Urban Planning and Development (ASCE) 2006; 132:172—186.

7. Correia G, Viegas JM. Carpooling and carpool clubs: clarifying concepts and assessing value enhancement possibilities through a Stated Preference web survey in Lisbon, Portugal. Transportation Research Part A: Practice 2011; 45:81—90.

8. Teal R. Shared-Ride Taxi Services as Community Public Transit: Final Report. In I. University of California (ed.). US Department of Transportation: Washington DC, 1981.

9. Gholami A, Mohaymany AS. Analogy of fixed route shared taxi (taxi khattee) and bus services under various demand density and economical conditions. Journal of Advanced Transportation 2012; 46:177—187.

10. Lee KT, Lin DJ, Wu PJ. Planning and design of a taxipooling dispatching system. Transit: Bus, Rural Public and Intercity, and Paratransit 2005; 86—95.

11. Darbera R. Taxicab regulation and urban residents' use and perception of taxi services: a survey in eight cities. 12th WCTR—World Conference in Transport Research. Lisbon, Portugal, 2010.

12. Kattan L, de Barros A, Wirasinghe SC. Analysis of work trips made by taxi in Canadian cities. Journal of Advanced Transportation 2010; 44:11—18.

13. Feibel CE. Paratransit and Urban Public Transport Policy in Low- and Medium-Income Countries: A Case Study of Istanbul, Turkey. University of North Carolina at Chapel Hill: Chapel Hill, North Carolina, 1987.

14. Cooper J, Mundy R, Nelson J. Taxi!: Urban Economies and the Social and Transport Impacts of the Taxicab. Farnham: Ashgate, 2010.

15. Lee DH, Teo SH, Wang SH, Cheo RL. Taxi dispatch system based on current demands and real-time traffic conditions. Transportation Network Modeling 2004; 2004:193—200.

16. Shrivastava M, Chande PK, Monga AS, Kashiwagi H. Taxi dispatch: a fuzzy rule approach. IEEE Conference on Intelligent Transportation Systems 1997; 978—982.

17. Von Massow M, Canbolat MS. Fareplay: an examination of taxicab drivers' response to dispatch policy. Expert Systems with Applications 2010; 37:2451—2458.

18. Wang H, Lee DH, Cheu R. Microscopic traffic simulation based dispatch modeling for taxi booking service. Sports Materials, Modelling and Simulation 2011; 187:677—682.

19. Wong KI, Bell MGH. The optimal dispatching of taxis under congestion: a rolling horizon approach. Journal of Advanced Transportation 2006; 40:203—220.

20. Alshamsi A, Abdallah S, Rahwan I. Multiagent self-organization for a taxi dispatch system. 8th International Joint Conference on Autonomous Agents and Multiagent Systems, Decker S, Sierra, Castelfranchi (eds.), Curran Associates, Inc.: New York, 2009.

21. Kim H, Oh JS, Jayakrishnan R. Effect of taxi information system on efficiency and quality of taxi services. Transportation Research Record, 2005; 96—104.

22. Kim H, Yang I, Choi K. An agent-based simulation model for analyzing the impact of asymmetric passenger demand on taxi service. Ksce Journal of Civil Engineering 2011; 15:187—195.

23. Seow KT, Dang NH, Lee D-H. Towards an automated multiagent taxi-dispatch system. IEEE Conference on Automation Science and Engineering. Scottsdale, AZ, USA, 2007.

24. Wong KI, Wong SC, Bell MGH, Yang H. Modeling the bilateral micro-searching behavior for urban taxi services using the absorbing Markov chain approach. Journal of Advanced Transportation 2005; 39:81—104.

# ДОДАТОК А
Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н.. Азаров О.Д.

" " 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської дипломної роботи

“Веб-додаток інформаційного обслуговування автопаркінгу”

08-23.БДР.032.00.000 ТЗ

Науковий керівник: доцент к.т.н.

_____ Ткаченко О. М.

Студент групи 2КІ-18б

_____ Черняховський І. Ю.

м. Вінниця — 2022

1 Підстава для використання бакалаврської кваліфікаційної роботи (БДР)

1.1 Актуальність розробки полягає у необхідності вирішення проблеми пошуку зручного додатку для пошуку водія та клієнта, шляхом розробки програмного забезпечення, шляхом якого є аналіз статистичних даних.

1.2 Наказ про затвердження теми бакалаврської дипломної роботи.

2 Мета і призначення БДР

2.1 Мета проекту — розробка зручного у використанні програмного забезпеченя для пошуку водія та клієнта й аналіз певних даних для виявлення зручності клієнта та пасажира.

2.2 Призначення розробки — виконання бакалаврського дипломного проекту із подальшим впровадженням та розвитком продукту.

3 Вихідні дані для виконання БДР

3.1 Проведення аналізу сучасних підходів до аналізу ситсемних даних додатку.

3.2 Розгляд принципів роботи додатку.

3.2 Розробка інтерфейсу та функціоналу веб-додатку.

4 Вимоги до виконання БДР

Головна вимога — розробити зручний інтерфейс до програмного забезпечення, використовуючи усі здобуті навчики та знання.

5 Етапи БДР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

6 Матеріали, що подаються до захисту БДР

До захисту подаються: пояснювальна записка БДР, ілюстративні

матеріали, протокол попереднього захисту БДР на кафедрі, відгук наукового керівника, анотації до БДР українською та іноземною мовами, довідка про відповідність оформлення БДР діючим вимогам.

Таблиця А.1 — Етапи БДР

| № з/п | Назва етапів виконання комплексної бакалаврської роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Постановка задачі роботи | 08.03.22 | Аналітичний огляд літературних джерел, розділ 1 |
| 2 | Аналіз предметної області | 09.03-18.03.22 | розділ 1 |
| 3 | Аналіз сучасних підходів до побудови інформаційних систем паркінгу | 22.03-31.03.22 | розділ 1 |
| 4 | Аналіз поняття веб-додатку | 05.04-15.04.22 | розділ 2 |
| 5 | Аналіз принципів роботи веб-додатків | 18.04-22.04.22 | розділ 2 |
| 6 | Проектування веб-додатку | 26.04-.06.05.22 | розділ 3 |
| 7 | Розробка веб-дизайну | 08.05-13.05.22 | розділ 3 |
| 8 | Розробка функціоналу веб-додатку | 18.05-20.05.22 | розділ 3 |
| 9 | Підготовка матеріалів та опис розробки інформаційної системи | 24.05-26.05.22 | Тези доповідей |
| 10 | Аналіз виконання роботи, висновки, додатки | 27.05-31.05.22 | Пояснювальна записка |
| 11 | Перевірка якості виконання бакалаврського проекту та усунення недоліків | 01.06 -.08.06.22 | Пояснювальна записка, графічний матеріал і презентація |

## 7 Порядок контролю виконання та захисту БДР

Виконання етапів графічної та розрахункової документації БДР контролюється науковим керівником згідно зі встановленими термінами. Захист БДР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

## 8 Вимоги до оформлювання та порядок виконання БДР

При оформлюванні БДР використовуються:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Едина система конструкторської документації. Основні написи»;

— документами на які посилаються у вище вказаних.

# ДОДАТОК Б

Лістинг коду основного блоку програмного забезпечення

```
package com.diploma.www.diploma;
import android.Manifest;
import android.app.Activity;
import android.app.LoaderManager;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentSender;
import android.content.Loader;
import android.content.pm.PackageManager;
import android.content.res.Resources;
import android.location.Criteria;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationManager;
import android.os.Build;
import android.os.Bundle;
import android.os.Looper;
import android.provider.Settings;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AlertDialog;
```

```
import android.support.v7.widget.CardView;
import android.text.TextUtils;
import android.util.Log;
import android.util.TypedValue;
import android.view.View;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.common.api.PendingResult;
import com.google.android.gms.common.api.ResultCallback;
import com.google.android.gms.common.api.Status;
import com.google.android.gms.identity.intents.Address;
import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationCallback;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;
```

```
import com.google.android.gms.location.LocationResult;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.location.LocationSettingsRequest;
import com.google.android.gms.location.LocationSettingsResult;
import com.google.android.gms.location.LocationSettingsStatusCodes;
import com.google.android.gms.location.SettingsClient;
import com.google.android.gms.location.places.Place;
import com.google.android.gms.location.places.PlaceBuffer;
import com.google.android.gms.location.places.Places;
import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.CameraPosition;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MapStyleOptions;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import java.io.IOException;
import java.util.List;
import java.util.Locale;
import static android.R.attr.x;
```

```
import static android.R.attr.y;
import static com.google.android.gms.location.LocationServices.FusedLocationApi;
import static
com.google.android.gms.location.LocationServices.getFusedLocationProviderClient;
import static com.tukla.www.tukla.R.id.map;
import static com.tukla.www.tukla.R.id.myLocation;
public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener,
OnMapReadyCallback, LoaderManager.LoaderCallbacks<Object>,
GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener,
        LocationListener {
    RelativeLayout products_select_option;
    ImageButton select_btn, product1, product2;
    Button myCurrentloc,book_button;
    private static final String TAG = MainActivity.class.getSimpleName();
    private final static int PERMISSION_MY_LOCATION = 3;
    private static final int REQUEST_CHECK_SETTINGS = 1000;
    private MapFragment mapFragment;
    private GoogleMap mMap;
    private GoogleApiClient googleApiClient;
    private Location mLastLocation;
    private LocationRequest request;
    View mapView;
    private boolean mRequestingLocationUpdates;
    CameraUpdate cLocation;
double latitude,longitude;
Marker now;
Geocoder geocoder;
```

```
List<android.location.Address> addresses;
@Override
protected void onCreate(Bundle savedInstanceState) {
setupLocationManager();
super.onCreate( savedInstanceState );
setContentView( R.layout.activity_main );
Toolbar toolbar = (Toolbar) findViewById( R.id.toolbar );
setSupportActionBar( toolbar );
mapFragment = (MapFragment) getFragmentManager()
.findFragmentById( R.id.map );
mapView = mapFragment.getView();
mapFragment.getMapAsync( this );
CheckMapPermission();

//        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
//        fab.setOnClickListener(new View.OnClickListener() {
//            @Override
//            public void onClick(View view) {
//                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
//                        .setAction("Action", null).show();
//
//
//            }
//        });
DrawerLayout drawer = (DrawerLayout) findViewById( R.id.drawer_layout );
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close );
```

```
drawer.setDrawerListener( toggle );
toggle.syncState();
NavigationView navigationView = (NavigationView) findViewById( R.id.nav_view
);
navigationView.setNavigationItemSelectedListener( this );
//Buttons Select Product option
select_btn = (ImageButton) findViewById( R.id.img_selected );
product1 = (ImageButton) findViewById( R.id.product_type_1_button );
product2 = (ImageButton) findViewById( R.id.product_type_2_button );
products_select_option = (RelativeLayout) findViewById(
R.id.products_select_option );
myCurrentloc=(Button) findViewById( R.id.myCLocation );
book_button=(Button)findViewById(R.id.book_button);
book_button.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
// TODO Auto-generated method stub
Intent i = new Intent(getApplicationContext(),FindDriver.class);
startActivity(i);
}
});
}
@Override
protected void onStart() {
super.onStart();
googleApiClient.connect();
//mGoogleApiClient.connect();
}
@Override
```

```
public void onBackPressed() {
DrawerLayout drawer = (DrawerLayout) findViewById( R.id.drawer_layout );
if (drawer.isDrawerOpen( GravityCompat.START )) {
drawer.closeDrawer( GravityCompat.START );
} else {
super.onBackPressed();
}
}
@Override
protected void onResume() {
super.onResume();
if(googleApiClient.isConnected()){
setInitialLocation();
}
LocationManager service = (LocationManager) getSystemService(
LOCATION_SERVICE );
boolean enabled = service.isProviderEnabled( LocationManager.GPS_PROVIDER );
// Check if enabled and if not send user to the GPS settings
if (!enabled) {
buildAlertMessageNoGps();
}
if(enabled){
//mMap.animateCamera( update );
/*
Toast.makeText( MainActivity.this, "OnResume:"+latitude+","+longitude,
Toast.LENGTH_SHORT ).show();
*/
}
}
```

```
@Override
protected void onPause() {
super.onPause();
}
@Override
protected void onDestroy() {
super.onDestroy();
if (googleApiClient.isConnected()) {
googleApiClient.disconnect();
}
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate( R.menu.main, menu );
return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
// Handle action bar item clicks here. The action bar will
// automatically handle clicks on the Home/Up button, so long
// as you specify a parent activity in AndroidManifest.xml.
int id = item.getItemId();
//noinspection SimplifiableIfStatement
if (id == R.id.action_settings) {
return true;
}
return super.onOptionsItemSelected( item );
}
```

```
@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
// Handle navigation view item clicks here.
int id = item.getItemId();
if (id == R.id.nav_camera) {
// Handle the camera action
} else if (id == R.id.nav_gallery) {
} else if (id == R.id.nav_slideshow) {
} else if (id == R.id.nav_manage) {
} else if (id == R.id.nav_share) {
} else if (id == R.id.nav_send) {
}
DrawerLayout drawer = (DrawerLayout) findViewById( R.id.drawer_layout );
drawer.closeDrawer( GravityCompat.START );
return true;
}
@Override
public void onMapReady(GoogleMap googleMap) {
mMap = googleMap;
try {
// Customise the styling of the base map using a JSON object defined
// in a raw resource file.
boolean success = mMap.setMapStyle(
MapStyleOptions.loadRawResourceStyle(
this, R.raw.map_style ) );
if (!success) {
Log.e( TAG, "Style parsing failed." );
}
```

```
} catch (Resources.NotFoundException e) {
Log.e( TAG, "Can't find style. Error: ", e );
}
if (ActivityCompat.checkSelfPermission( this,
android.Manifest.permission.ACCESS_FINE_LOCATION ) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission( this,
android.Manifest.permission.ACCESS_COARSE_LOCATION ) !=
PackageManager.PERMISSION_GRANTED) {
// TODO: Consider calling
//    ActivityCompat#requestPermissions
// here to request the missing permissions, and then over   riding
//   public void onRequestPermissionsResult(int requestCode, String[] permissions,
//                                          int[] grantResults)
// to handle the case where the user grants the permission. See the documentation
// for ActivityCompat#requestPermissions for more details.
return;
}
//This line will show your current location on Map with GPS dot
mMap.setMyLocationEnabled( true );
locationButton();
/*
Toast.makeText( MainActivity.this, "OnStart:"+latitude+","+longitude,
Toast.LENGTH_SHORT ).show();
*/
}
private void setupLocationManager() {
//buildGoogleApiClient();
if (googleApiClient == null) {
```

```
googleApiClient = new GoogleApiClient.Builder( this )
.addConnectionCallbacks( this )
.addOnConnectionFailedListener( this )
.addApi( LocationServices.API )
.addApi( Places.GEO_DATA_API )
.addApi( Places.PLACE_DETECTION_API )
.build();
//mGoogleApiClient = new GoogleApiClient.Builder(this);
}
googleApiClient.connect();
createLocationRequest();
}
protected void createLocationRequest() {
request = new LocationRequest();
request.setSmallestDisplacement( 10 );
request.setFastestInterval( 50000 );
request.setPriority( LocationRequest.PRIORITY_HIGH_ACCURACY );
request.setNumUpdates( 3 );
LocationSettingsRequest.Builder builder = new LocationSettingsRequest.Builder()
.addLocationRequest( request );
builder.setAlwaysShow( true );
PendingResult<LocationSettingsResult> result =
LocationServices.SettingsApi.checkLocationSettings( googleApiClient,
builder.build() );
result.setResultCallback( new ResultCallback<LocationSettingsResult>() {
@Override
public void onResult(@NonNull LocationSettingsResult result) {
final Status status = result.getStatus();
switch (status.getStatusCode()) {
```

```
case LocationSettingsStatusCodes.SUCCESS:
setInitialLocation();
break;
case LocationSettingsStatusCodes.RESOLUTION_REQUIRED:
// Location settings are not satisfied, but this can be fixed
// by showing the user a dialog.
try {
// Show the dialog by calling startResolutionForResult(),
// and check the result in onActivityResult().
status.startResolutionForResult(
MainActivity.this,
REQUEST_CHECK_SETTINGS );
} catch (IntentSender.SendIntentException e) {
// Ignore the error.
}
break;
case LocationSettingsStatusCodes.SETTINGS_CHANGE_UNAVAILABLE:
// Location settings are not satisfied. However, we have no way
// to fix the settings so we won't show the dialog.
break;
}
}
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
Log.d( "onActivityResult()", Integer.toString( resultCode ) );
//final LocationSettingsStates states = LocationSettingsStates.fromIntent(data);
switch (requestCode) {
case REQUEST_CHECK_SETTINGS:
switch (resultCode) {
```

```
case Activity.RESULT_OK: {
setInitialLocation();
Toast.makeText( MainActivity.this, "Location enabled", Toast.LENGTH_LONG
).show();
mRequestingLocationUpdates = true;
break;
}
case Activity.RESULT_CANCELED: {
// The user was asked to change settings, but chose not to
Toast.makeText( MainActivity.this, "Location not enabled", Toast.LENGTH_LONG
).show();
mRequestingLocationUpdates = false;
break;
}
default: {
break;
}
}
break;
}
}
private void setInitialLocation() {
if (ActivityCompat.checkSelfPermission( MainActivity.this,
android.Manifest.permission.ACCESS_FINE_LOCATION ) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission( MainActivity.this,
android.Manifest.permission.ACCESS_COARSE_LOCATION ) !=
PackageManager.PERMISSION_GRANTED) {
// TODO: Consider calling
```

```
//    ActivityCompat#requestPermissions
// here to request the missing permissions, and then overriding
//   public void onRequestPermissionsResult(int requestCode, String[] permissions,
//                                          int[] grantResults)
// to handle the case where the user grants the permission. See the documentation
// for ActivityCompat#requestPermissions for more details.
return;
}
LocationServices.FusedLocationApi.requestLocationUpdates( googleApiClient,
request, new LocationListener() {
@Override
public void onLocationChanged(Location location) {
mLastLocation = location;
double lat=location.getLatitude();
double lng=location.getLongitude();
MainActivity.this.latitude=lat;
MainActivity.this.longitude=lng;
try {
if(now !=null){
now.remove();
}
LatLng positionUpdate = new LatLng(
MainActivity.this.latitude,MainActivity.this.longitude );
CameraUpdate update = CameraUpdateFactory.newLatLngZoom( positionUpdate, 15
);
now=mMap.addMarker(new MarkerOptions().position(positionUpdate)
.title("Your Location"));
mMap.animateCamera( update );
//myCurrentloc.setText( ""+latitude );
```

```
} catch (Exception ex) {
ex.printStackTrace();
Log.e( "MapException", ex.getMessage() );
}
//Geocode current location details
try {
geocoder = new Geocoder(MainActivity.this, Locale.ENGLISH);
addresses = geocoder.getFromLocation(latitude, longitude, 1);
StringBuilder str = new StringBuilder();
if (Geocoder.isPresent()) {
/*Toast.makeText(getApplicationContext(),
"geocoder present", Toast.LENGTH_SHORT).show();*/
android.location.Address returnAddress = addresses.get(0);
String localityString = returnAddress.getAddressLine (0);
//String city = returnAddress.getAddressLine(1);
//String region_code = returnAddress.getAddressLine(2);
//String zipcode = returnAddress.getAddressLine(3);
str.append( localityString ).append( "" );
// str.append( city ).append( "" ).append( region_code ).append( "" );
// str.append( zipcode ).append( "" );
myCurrentloc.setText(str);
Toast.makeText(getApplicationContext(), str,
Toast.LENGTH_SHORT).show();
} else {
/*    Toast.makeText(getApplicationContext(),
"geocoder not present", Toast.LENGTH_SHORT).show();*/
}
// } else {
// Toast.makeText(getApplicationContext(),
```

```
// "address not available", Toast.LENGTH_SHORT).show();
// }
} catch (IOException e) {
// TODO Auto-generated catch block
Log.e("tag", e.getMessage());
}
}
} );
}
private void updateCamera(){
}
private void CheckMapPermission() {
if (Build.VERSION.SDK_INT > Build.VERSION_CODES.LOLLIPOP) {
if (ActivityCompat.checkSelfPermission( MainActivity.this,
android.Manifest.permission.ACCESS_FINE_LOCATION ) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission( MainActivity.this,
android.Manifest.permission.ACCESS_COARSE_LOCATION ) !=
PackageManager.PERMISSION_GRANTED) {
ActivityCompat.requestPermissions( MainActivity.this, new
String[]{android.Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION}, 1002 );
} else {
setupLocationManager();
}
} else {
setupLocationManager();
}
}
```

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
super.onRequestPermissionsResult( requestCode, permissions, grantResults );
switch (requestCode) {
case 1002: {
// If request is cancelled, the result arrays are empty.
if (grantResults.length > 0
&& grantResults[0] == PackageManager.PERMISSION_GRANTED) {
if (ActivityCompat.checkSelfPermission( this,
Manifest.permission.ACCESS_FINE_LOCATION )
== PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission( this,
Manifest.permission.ACCESS_COARSE_LOCATION ) ==
PackageManager.PERMISSION_GRANTED) {
setupLocationManager();
}
} else {
Toast.makeText( MainActivity.this, "Permission Denied", Toast.LENGTH_SHORT
).show();
//finish();
}
}
break;
}
}
public void getLatLang(String placeId) {
Places.GeoDataApi.getPlaceById( googleApiClient, placeId )
.setResultCallback( new ResultCallback<PlaceBuffer>() {
```

```
@Override
public void onResult(PlaceBuffer places) {
if (places.getStatus().isSuccess() && places.getCount() > 0) {
final Place place = places.get( 0 );
LatLng latLng = place.getLatLng();
try {
CameraUpdate update = CameraUpdateFactory.newLatLngZoom( latLng, 15 );
mMap.animateCamera( update );
} catch (Exception ex) {
ex.printStackTrace();
Log.e( "MapException", ex.getMessage() );
}
Log.i( "place", "Place found: " + place.getName() );
} else {
Log.e( "place", "Place not found" );
}
places.release();
}
} );
}
@Override
public void onConnected(@Nullable Bundle bundle) {
//AlertMessageNoGps();
}
@Override
public void onConnectionSuspended(int i) {
//checkLocaionStatus();
}
@Override
```

```
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
}
@Override
public void onLocationChanged(Location location) {
}
@Override
public void onPointerCaptureChanged(boolean hasCapture) {
}
@Override
public Loader<Object> onCreateLoader(int i, Bundle bundle) {
return null;
}
@Override
public void onLoadFinished(Loader<Object> loader, Object o) {
}
@Override
public void onLoaderReset(Loader<Object> loader) {
}
//GET CURRENT LOCATION BUTTON POSITION....
private void locationButton() {
MapFragment mapFragment = (MapFragment) getFragmentManager()
.findFragmentById( map );
View locationButton = ((View) mapFragment.getView().findViewById(
Integer.parseInt( "1" ) ).
getParent()).findViewById( Integer.parseInt( "2" ) );
if (locationButton != null && locationButton.getLayoutParams() instanceof
RelativeLayout.LayoutParams) {
// location button is inside of RelativeLayout
```

```
RelativeLayout.LayoutParams params = (RelativeLayout.LayoutParams)
locationButton.getLayoutParams();
// Align it to - parent BOTTOM|LEFT
params.addRule( RelativeLayout.ALIGN_PARENT_BOTTOM );
params.addRule( RelativeLayout.ALIGN_PARENT_LEFT );
params.addRule( RelativeLayout.ALIGN_PARENT_RIGHT, 0 );
params.addRule( RelativeLayout.ALIGN_PARENT_TOP, 0 );
// Update margins, set to 10dp
final int margin = (int) TypedValue.applyDimension(
TypedValue.COMPLEX_UNIT_DIP, 150,
getResources().getDisplayMetrics() );
params.setMargins( margin, margin, margin, margin );
locationButton.setLayoutParams( params );
}
}
//Button Location Search
public void myLocation(View view) {
//CHANGE ACTIVITY
Intent intent = new Intent( MainActivity.this, LocationAutoActivity.class );
startActivity( intent );
}
public void destination(View view) {
Intent intent = new Intent( MainActivity.this, LocationAutoActivity.class );
startActivity( intent );
}
//Select product option button click
public void img_selected(View view) {
products_select_option.setVisibility( View.VISIBLE );
}
```

```
//Select product option button click
public void product_type_1_button(View view) {
products_select_option.setVisibility( View.GONE );
}
//Select product option button click
public void product_type_2_button(View view) {
products_select_option.setVisibility( View.GONE );
}
//Enable Location Button
private void checkLocaionStatus() {
}
protected void buildAlertMessageNoGps() {
final AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("Please Turn ON your GPS Connection")
.setTitle( "GPS Not Enabled" )
.setCancelable(false)
.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
public void onClick(final DialogInterface dialog, final int id) {
startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
}
})
.setNegativeButton("No", new DialogInterface.OnClickListener() {
public void onClick(final DialogInterface dialog, final int id) {
dialog.cancel();
}
});
final AlertDialog alert = builder.create();
alert.show();
} }
```

# ДОДАТОК В

Лістинг коду блоку програмного забезпечення

```
# Resolve links: $0 may be a link
PRG="$0"
# Need this for relative symlinks.
while [ -h "$PRG" ] ; do
    ls=`ls -ld "$PRG"`
    link=`expr "$ls" : '.*-> \(.*\)$'`
    if expr "$link" : '/.*' > /dev/null; then
        PRG="$link"
    else
        PRG=`dirname "$PRG"`"/$link"
    fi
done
SAVED="`pwd`"
cd "`dirname \"$PRG\"`/" >/dev/null
APP_HOME="`pwd -P`"
cd "$SAVED" >/dev/null

APP_NAME="Gradle"
APP_BASE_NAME=`basename "$0"`

# Add default JVM options here. You can also use JAVA_OPTS and
GRADLE_OPTS to pass JVM options to this script.
DEFAULT_JVM_OPTS=""

# Use the maximum available, or set MAX_FD != -1 to use that value.
```

```
MAX_FD="maximum"

warn ( ) {
    echo "$*"
}

die ( ) {
    echo
    echo "$*"
    echo
    exit 1
}

# OS specific support (must be 'true' or 'false').
cygwin=false
msys=false
darwin=false
nonstop=false
case "`uname`" in
  CYGWIN* )
    cygwin=true
    ;;
  Darwin* )
    darwin=true
    ;;
  MINGW* )
    msys=true
    ;;
```

```
  NONSTOP* )
    nonstop=true
    ;;
esac

CLASSPATH=$APP_HOME/gradle/wrapper/gradle-wrapper.jar

# Determine the Java command to use to start the JVM.
if [ -n "$JAVA_HOME" ] ; then
    if [ -x "$JAVA_HOME/jre/sh/java" ] ; then
        # IBM's JDK on AIX uses strange locations for the executables
        JAVACMD="$JAVA_HOME/jre/sh/java"
    else
        JAVACMD="$JAVA_HOME/bin/java"
    fi
    if [ ! -x "$JAVACMD" ] ; then
        die "ERROR: JAVA_HOME is set to an invalid directory:
$JAVA_HOME

Please set the JAVA_HOME variable in your environment to match the
location of your Java installation."
    fi
else
    JAVACMD="java"
    which java >/dev/null 2>&1 || die "ERROR: JAVA_HOME is not set
and no 'java' command could be found in your PATH.

Please set the JAVA_HOME variable in your environment to match the
```

```
location of your Java installation."
fi

# Increase the maximum file descriptors if we can.
if [ "$cygwin" = "false" -a "$darwin" = "false" -a "$nonstop" = "false" ] ;
then
    MAX_FD_LIMIT=`ulimit -H -n`
    if [ $? -eq 0 ] ; then
        if [ "$MAX_FD" = "maximum" -o "$MAX_FD" = "max" ] ; then
            MAX_FD="$MAX_FD_LIMIT"
        fi
        ulimit -n $MAX_FD
        if [ $? -ne 0 ] ; then
            warn "Could not set maximum file descriptor limit: $MAX_FD"
        fi
    else
        warn "Could not query maximum file descriptor limit:
$MAX_FD_LIMIT"
    fi
fi

# For Darwin, add options to specify how the application appears in the
dock
if $darwin; then
    GRADLE_OPTS="$GRADLE_OPTS \"-Xdock:name=$APP_NAME\"
\"-Xdock:icon=$APP_HOME/media/gradle.icns\""
fi
```

```
# For Cygwin, switch paths to Windows format before running java
if $cygwin ; then
    APP_HOME=`cygpath --path --mixed "$APP_HOME"`
    CLASSPATH=`cygpath --path --mixed "$CLASSPATH"`
    JAVACMD=`cygpath --unix "$JAVACMD"`

    # We build the pattern for arguments to be converted via cygpath
    ROOTDIRSRAW=`find -L / -maxdepth 1 -mindepth 1 -type d
2>/dev/null`
    SEP=""
    for dir in $ROOTDIRSRAW ; do
        ROOTDIRS="$ROOTDIRS$SEP$dir"
        SEP="|"
    done
    OURCYGPATTERN="(^($ROOTDIRS))"
    # Add a user-defined pattern to the cygpath arguments
    if [ "$GRADLE_CYGPATTERN" != "" ] ; then

OURCYGPATTERN="$OURCYGPATTERN|($GRADLE_CYGPATTE
RN)"
    fi
    # Now convert the arguments - kludge to limit ourselves to /bin/sh
    i=0
    for arg in "$@" ; do
        CHECK=`echo "$arg"|egrep -c "$OURCYGPATTERN" -`
        CHECK2=`echo "$arg"|egrep -c "^-"`                        ###
Determine if an option
```

```
        if [ $CHECK -ne 0 ] && [ $CHECK2 -eq 0 ] ; then                    ###
Added a condition
            eval `echo args$i`=`cygpath --path --ignore --mixed "$arg"`
        else
            eval `echo args$i`="\"$arg\""
        fi
        i=$((i+1))
    done
    case $i in
        (0) set -- ;;
        (1) set -- "$args0" ;;
        (2) set -- "$args0" "$args1" ;;
        (3) set -- "$args0" "$args1" "$args2" ;;
        (4) set -- "$args0" "$args1" "$args2" "$args3" ;;
        (5) set -- "$args0" "$args1" "$args2" "$args3" "$args4" ;;
        (6) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" ;;
        (7) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5"
"$args6" ;;
        (8) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5"
"$args6" "$args7" ;;
        (9) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5"
"$args6" "$args7" "$args8" ;;
    esac
fi

# Split up the JVM_OPTS And GRADLE_OPTS values into an array,
following the shell quoting and substitution rules
```

```
function splitJvmOpts() {
    JVM_OPTS=("$@")
}
eval splitJvmOpts $DEFAULT_JVM_OPTS $JAVA_OPTS
$GRADLE_OPTS
```

## ДОДАТОК Г

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ
ЗАПОЗИЧЕНЬ

Назва роботи: Програмне забезпечення для прогнозування замовлень таксі на основі статистичної обробки отриманих даних

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ: кафедра обчислювальної техніки
(кафедра, факультет)

**Показники звіту подібності Unichec**

Оригінальність 87,8% Схожість 12,2%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Черняховський І. Ю.

Керівник роботи _____ Ткаченко О. М.