СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра інформаційних технологій та програмування

Пояснювальна записка

до магістерської дипломної роботи
магістр
(освітньо-кваліфікаційний рівень)
на тему Мобільна інформаційна система для аналізу та оптимізації пошуку паркувальних місць для водіїв

Виконав: студент 2 курсу, групи ІСТ-22дм
126 «Інформаційні системи та технології»
(шифр і назва спеціальності)
Шанцев В. С.
(прізвище та ініціали)
Керівник Лифар В. О.
(прізвище та ініціали)
Рецензент Меняйленко О.С.
(прізвище та ініціали)

Київ – 2023 року

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ ДО МАГІСТЕРСЬКОЇ ДИПЛОМНОЇ РОБОТИ

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

Освітньо-кваліфікаційний рівень магістр

Спеціальність 126 «Інформаційні системи та технології»

(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТП
_____ д.т.н., доц. Лифар В.О.
(підпис)
«____»_____202__р.

ЗАВДАННЯ
на магістерську дипломну роботу студенту
Шанцеву Владиславу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Мобільна інформаційна система для аналізу та оптимізації пошуку паркувальних місць для водіїв
керівник роботи Лифар Володимир Олексійович, д.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «___»_____202_року №____
2. Строк подання студентом роботи 06.12.2023
3. Вихідні дані до роботи _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)_____

5. Перелік графічного матеріалу (з точним значенням обов'язків креслень)_____

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 20 жовтня 2023

КАЛЕНДАРНИЙ ПЛАН

№ з\п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної галузі	25.10.23 – 28.10.23	
2	Пошук та аналіз існуючих рішень	29.10.23 – 04.11.23	
3	Аналіз методів та алгоритмів	05.11.23 – 10.11.23	
4	Розробка архітектури системи	11.11.23 – 17.11.23	
5	Реалізація мобільної інформаційної системи	18.11.23 – 30.11.23	
6	Тестування мобільного додатку	30.11.23 – 01.12.23	
7	Оформлення пояснювальної записки	01.12.23 – 05.12.23	
8	Підготовка та подання магістерської роботи до захисту	06.12.23 – 06.12.23	

Студент _____ Шанцев В.С
(підпис) (прізвище та ініціали)

Керівник роботи _____ Лифар В.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська дипломна робота: 67 стор., 11 рис., 20 джерел.

Мета роботи – впровадження мобільної інформаційної системи, яка дозволить водіям швидко та ефективно аналізувати та оптимізувати пошук паркувальних місць, сприяючи покращенню загальної ситуації з паркуванням в місті.

Об'єкт дослідження – система автоматизованого пілотування для автомобілів з звикористанням штучного інтелекту для оптимізації та управління поведінкою автотранспортних засобів.

Завдання дослідження:

1. Розробка користувацького інтерфейсу мобільної інформаційної системи для введення та відображення даних щодо доступних паркувальних місць.
2. Аналіз та вибір оптимальних алгоритмів для автоматизованого визначення доступності та ефективного використання паркувальних зон.
3. Реалізація системи сповіщення водіїв про вільні паркувальні місця та оптимальні маршрути до них.
4. Проведення тестувань ефективності та відгуків користувачів щодо використання мобільної інформаційної системи.
5. Аналіз впливу впровадження системи на загальний стан паркувальної інфраструктури міста та покращення міської мобільності.

Наукова новизна:

Розробка мобільної інформаційної системи для оптимізації паркування має велике значення, оскільки сприяє покращенню міської мобільності,

зменшенню транспортних заторів та ефективнішому використанню доступних паркувальних ресурсів. Дослідження спрямоване на вирішення актуальної проблеми нестачі паркувальних місць у містах, що покликане поліпшити якість життя громадян та забезпечити сталевий розвиток транспортної інфраструктури.

Проведено аналіз стану використання та розвитку штучного інтелекту в автомобільній промисловості. Розглянуто ключові аспекти впровадження ШІ в системи автомобілів та визначено основні можливості цього напряму. Реалізовано систему автоматизованого навчання автомобілів автономного руху з здатністю адаптуватися до різних дорожніх ситуацій та проведено експерименти у стимуляційному середовищі.

МОБІЛЬНА ІНФОРМАЦІЙНА СИСТЕМА, ПАРКУВАЛЬНІ МІСЦЯ, ОПТИМІЗАЦІЯ ПАРКУВАННЯ, ТРАНСПОРТНА ІНФРАСТРУКТУРА, МІСЬКА МОБІЛЬНІСТЬ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, АЛГОРИТМИ, ТЕСТУВАННЯ ЕФЕКТИВНОСТІ.

ЗМІСТ

ВСТУП

В сучасному місті, яке переживає стрімке зростання автопарку та підвищення обсягів транспортного руху, проблема пошуку паркувальних місць стає не тільки нагальною, але й стратегічно важливою. Зростання кількості автомобілів призводить до дефіциту доступних паркувальних місць, що ускладнює життя водіїв та призводить до затримок у транспортному русі.

Актуальність теми полягає в тому, що існуючі методи пошуку паркувальних місць недостатньо ефективні та не враховують різноманітні фактори, такі як часові обмеження, доступність місць та інші параметри. Шлях до вирішення цієї проблеми полягає у розробці мобільної інформаційної системи[1], яка не лише допомагатиме водіям знаходити вільні паркувальні місця, а й аналізуватиме та оптимізуватиме процес паркування.

Метою даного дослідження є розробка та впровадження мобільної інформаційної системи, яка забезпечить водіїв необхідною інформацією про вільні паркувальні місця в режимі реального часу. Основні завдання включають в себе визначення технічних вимог до системи, аналіз існуючих рішень та їх недоліків, розробку оптимальних алгоритмів пошуку та вибору паркувальних місць, а також впровадження системи в реальному середовищі та аналіз її ефективності.

Об'єктом дослідження є система пошуку та оптимізації паркувальних місць для водіїв в умовах міського середовища. Предметом дослідження є механізми та технології, які лягають в основу розробленої мобільної інформаційної системи.

Наукова новизна даного дослідження полягає у впровадженні новаторських підходів до аналізу та оптимізації пошуку паркувальних місць на основі використання сучасних технологій та алгоритмів. Практична цінність полягає в можливості покращення роботи міського транспорту та полегшенні

повсякденного життя водіїв, зменшенні заторів та економії часу.

Таким чином, дана магістерська дипломна робота має на меті вирішити актуальну проблему пошуку паркувальних місць через розробку та впровадження мобільної інформаційної системи, спрямованої на оптимізацію процесу паркування та покращення умов пересування в міському середовищі.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

Сучасне міське середовище стикається з рядом проблем, серед яких є проблема пошуку паркувальних місць. Зі зростанням кількості автомобілів та збільшенням густоти транспортного руху, водії стикаються з необхідністю ефективного використання часу для знаходження вільних місць для паркування. Ця проблема не тільки призводить до затримок та стресу для водіїв, але також сприяє утворенню транспортних заторів та негативно впливає на загальний рух у місті.

Брак паркомісць також призводить до зменшення доступності парковок для мешканців та відвідувачів міста, що ускладнює їх рух та викликає незручності. Зайняті парковочні місця на вулицях та в громадських місцях часто призводять до незаконного паркування та конфліктів між водіями.

Для вирішення цих проблем проводяться різні заходи, включаючи встановлення більше парковочних місць, вдосконалення системи паркування та використання технологій. Однак, важливим кроком у розв'язанні проблеми є розробка і впровадження мобільних додатків, які допомагають водіям знайти та забронювати паркомісце заздалегідь. Такі додатки дозволяють ефективно використовувати час та уникнути зайвих затримок, а також сприяють покращенню загального руху у місті.

Розробка такого мобільного додатку має високий потенціал для поліпшення міської інфраструктури та зручності для водіїв. Він може включати функції маршрутизації до найближчих вільних парковочних зон, відображення реального часу доступності паркомісць та можливості бронювання парковки заздалегідь. Такий додаток може бути інструментом для оптимізації використання паркомісць та зменшення нерегулярного руху водіїв у пошуку вільних місць.

Використання мобільних додатків для пошуку та бронювання паркомісць є кроком у майбутнє, де ефективне управління паркуванням може покращити якість життя в містах та зменшити вплив транспорту на довкілля. Водії зможуть швидше та зручніше знайти парковочне місце, а міста зможуть оптимізувати свою інфраструктуру паркування.

1.1 Аналіз існуючих систем для пошуку паркувальних місць

1.1.1 Google Maps

Google Maps[7] – це географічний сервіс, розроблений компанією Google. Цей додаток надає користувачам детальні мапи, навігацію, інформацію про транспорт, ресторани, парковки та багато іншого. Запущений у 2005 році, Google Maps швидко став не тільки незамінним інструментом для користувачів, а й ключовим застосунком у геолокаційній та картографічній індустріях.

Основні функції та особливості:

- Навігація:

Google Maps надає детальні маршрути та навігацію для автівок, велосипедів, пішоходів та громадського транспорту. Враховуючи реальний час, додаток пропонує оптимальні маршрути з урахуванням умов дорожнього руху.

- Інформація про парковки:

Користувачі можуть переглядати інформацію про парковки, включаючи вартість, доступність та типи місць. Також можна знаходити парковки біля конкретного місця призначення.

- Глобальні мапи:

Google Maps охоплює майже усі куточки світу, забезпечуючи глобальний огляд мап та дозволяючи користувачам досліджувати різні регіони.

- Інтеграція з Google Pay:

Сервіс інтегрований із Google Pay, що дозволяє користувачам оплачувати парковку та інші послуги безпосередньо через додаток.

- Оцінки та відгуки:

Користувачі можуть залишати відгуки, ділитися фотографіями та оцінювати місця, що допомагає іншим користувачам отримувати інформацію про певні заклади чи локації.

- Інтеграція з іншими сервісами:

Google Maps[7] підтримує інтеграцію з іншими сервісами Google, такими як Google Calendar, дозволяючи автоматично додавати події та зустрічі до маршрутів.

Переваги та Недоліки:

- Переваги:

1. Широкий функціонал та оглядова якість мап.
2. Постійне оновлення та покращення сервісу.
3. Глобальний охоплення та точність інформації.

Недоліки:

1. В деяких регіонах може бути менш деталізована інформація про парковки.

2. Залежність від доступу до Інтернету для використання всіх функцій.

Google Maps[7] залишається одним з найпопулярніших та найбільш функціональних сервісів для навігації та пошуку інформації про парковки. Його поєднання точності, охоплення та інтеграції з іншими сервісами робить його незамінним інструментом для користувачів у міському середовищі.

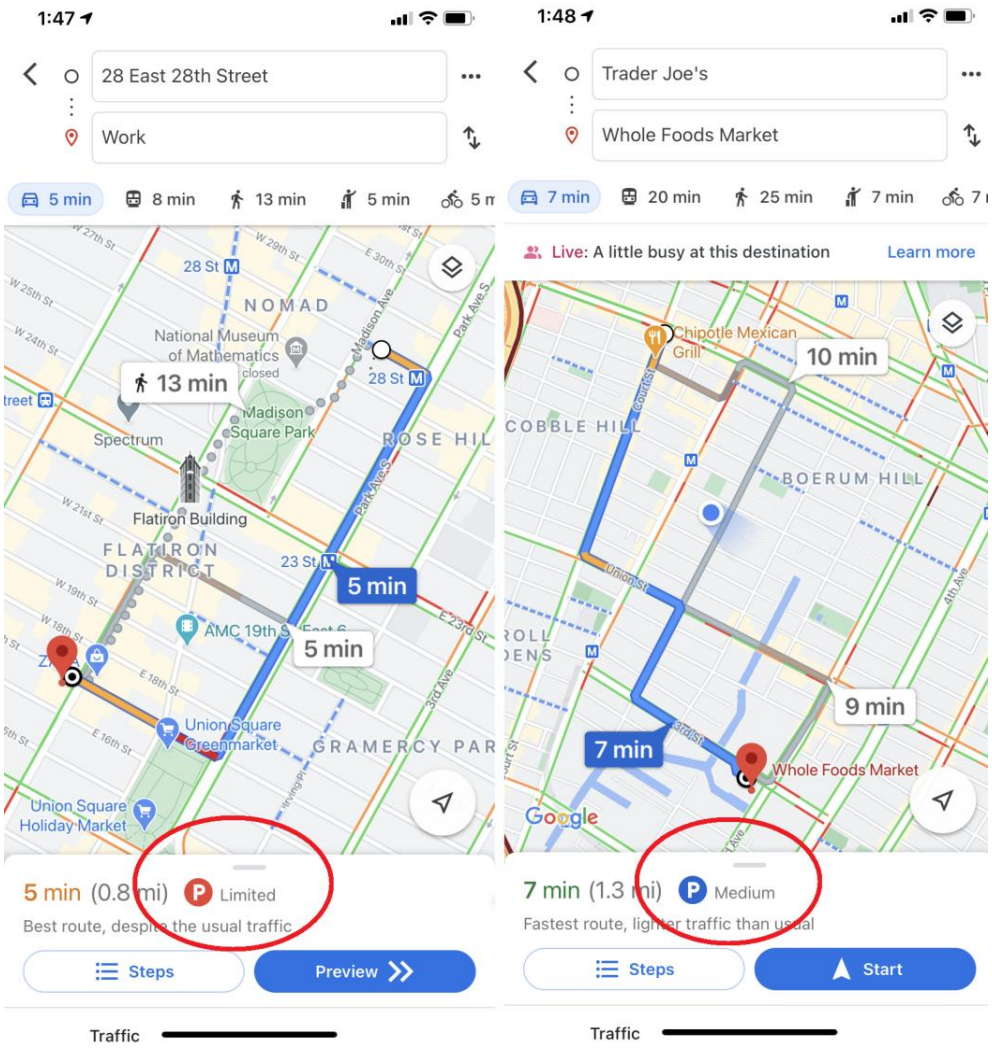

Рис. 1 – Google Maps

1.1.2 ParkMobile

ParkMobile[18] – це мобільний додаток, спрямований на полегшення

процесу паркування для користувачів у містах та різних локаціях. Розроблений компанією ParkMobile Group, цей сервіс дозволяє водіям зручно та ефективно оплачувати паркування, знаходити доступні парковочні місця та контролювати час.

Основні Функції та Особливості:

- Мобільна оплата паркування:

Користувачі можуть використовувати ParkMobile для оплати паркування, уникнення необхідності користування метров та монет.

- Знаходження та резервування паркомісць:

Додаток допомагає користувачам знаходити парковочні місця, а в деяких випадках – резервувати їх наперед.

- Таймер та сповіщення:

Функція таймера дозволяє користувачам встановлювати обмеження часу паркування, а сповіщення нагадують про завершення терміну.

- Історія та витрати:

Користувачі можуть переглядати історію своїх операцій, а також визначати витрати на паркування за певний період.

- Інтеграція з GPS:

Використання технології GPS дозволяє точно визначати місце паркування та дозволяє користувачам легко знайти своє авто.

Переваги та Недоліки:

- Переваги:

1. Зручний інтерфейс та швидка оплата паркування.
2. Велика мережа підтримуваних локацій у різних містах.
3. Можливість резервування паркомісць для забезпечення доступності.

- Недоліки:

1. Залежність від покриття сервісом певного міста чи регіону.
2. Обмеженість функціоналу у деяких менших містах.

ParkMobile виграє своїм простим та ефективним інтерфейсом для оплати паркування та можливістю знаходження та резервування паркомісць. Зручність використання та широка підтримка в багатьох місцях роблять його популярним серед користувачів[18].

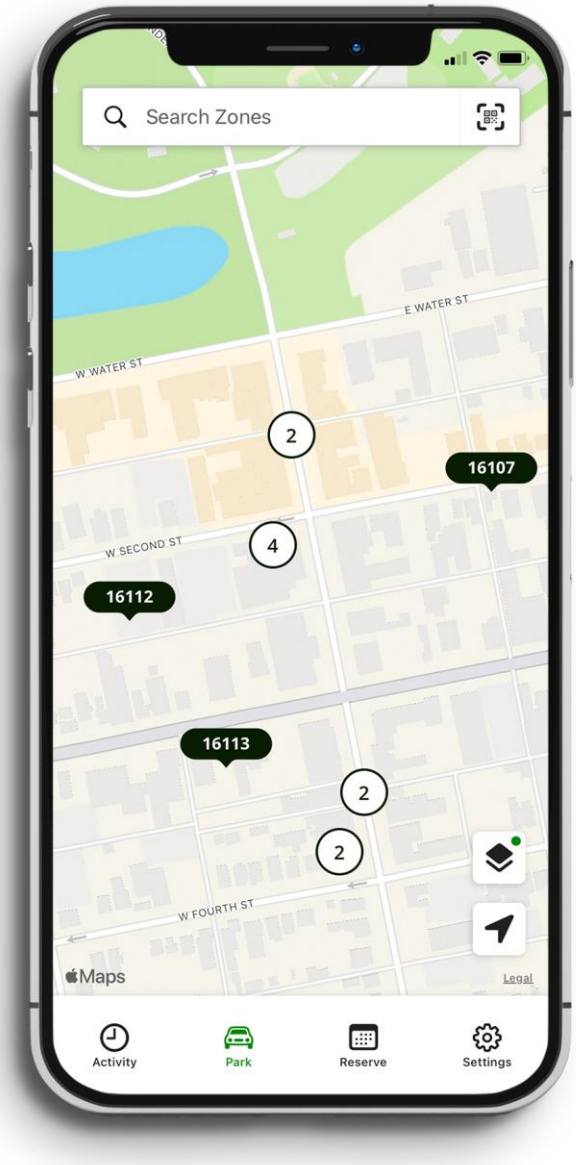

Рис. 2 – ParkMobile

1.1.3 Waze

Waze[19] є мобільним додатком для навігації та розповсюдження інформації про дорожні умови. Заснований в 2006 році і придбаний Google у 2013 році, Waze відзначається тим, що використовує дані, надані самими користувачами, для надання актуальної та точної інформації про транспорт та

дорожні умови.

Основні Функції та Особливості:

- Колективна навігація:

Waze використовує дані, які надають користувачі, для визначення оптимального маршруту на основі реального часу та даних про дорожні умови.

- Інформація про ДТП та поліцію:

Користувачі можуть повідомляти про ДТП, поліцейських та інші події на дорозі, щоб інші водії отримували оновлену інформацію.

- Глобальна спільнота:

Waze об'єднує користувачів у глобальну спільноту, яка ділиться реальними даними про дорожні умови та події.

- Інтеграція з картами та Сервісами Google:

Додаток інтегрований з Google Maps та іншими сервісами, що забезпечує широкий спектр можливостей для користувачів.

- Ефективне управління часом поїздок:

Waze враховує додаткові фактори, такі як час поїздки та інші внутрішні зручності, щоб оптимізувати маршрут.

- Повідомлення про найшвидший маршрут:

Користувачі отримують сповіщення про найшвидший маршрут та оптимальний час відправлення.

Переваги та Недоліки:

- Переваги:

1. Динамічна та актуальна інформація про дорожні умови.
2. Спільнота користувачів, яка забезпечує постійне оновлення даних.
3. Інтеграція з іншими сервісами Google.

- Недоліки:

1. Потребує активного підключення до Інтернету для повного функціоналу.
2. Залежність від активності користувачів у певному регіоні.

Waze вирізняється своєю унікальною концепцією колективної навігації та наданням реальної інформації від користувачів. Завдяки цьому додаток допомагає уникнути транспортних заторів та забезпечити ефективну навігацію для водіїв[19].

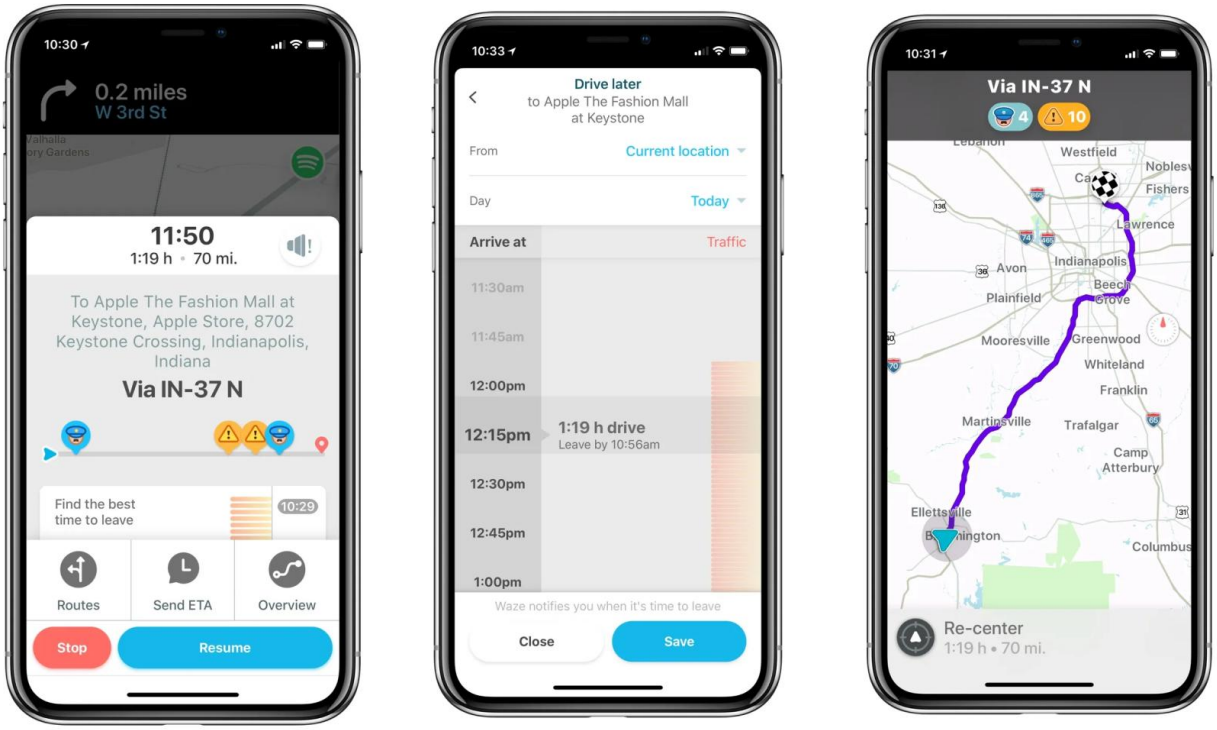

Рис. 3 – Waze

1.1.4 SpotHero

SpotHero[20] – це мобільний додаток, спрямований на забезпечення користувачів зручним та ефективним способом пошуку та резервування парковочних місць, особливо у великих містах та заторах. Заснований у 2011 році, SpotHero став популярним серед водіїв, які шукають зручні парковочні місця.

Основні функції та особливості:

- Пошук та Резервування Паркомісць:

SpotHero надає користувачам можливість знаходження та резервування парковочних місць заздалегідь, щоб гарантувати їх доступність.

- Інтерактивна мапа зони паркування:

Користувачі можуть оглядати інтерактивні мапи з позначеними зонами паркування та вибирати оптимальне місце.

- Мобільні парковочні купони та знижки:

SpotHero пропонує користувачам різноманітні купони та знижки на паркування, що дозволяє заощаджувати кошти.

- Інтеграція з GPS:

Використання технології GPS дозволяє точно визначити місце паркування та надає зручність знаходження авто.

- Історія та звітність:

SpotHero веде історію резервувань та надає звіти щодо витрат на паркування.

- Сповіщення та нагадування:

Система сповіщень та нагадувань допомагає вчасно орієнтуватися та керувати часом паркування.

Переваги та Недоліки:

- Переваги:

1. Зручний та швидкий процес пошуку та резервування паркомісць.
2. Велика мережа підтримуваних локацій у великих містах.
3. Мобільні купони та знижки додають економічну вигоду.

- Недоліки:

1. Деякі області можуть мати обмежений вибір доступних паркомісць.

2. Залежність від доступності сервісу у конкретних містах та районах.

SpotHero допомагає водіям уникати головних проблем паркування у великих містах, надаючи швидкі та ефективні можливості пошуку та резервування парко місць[20].

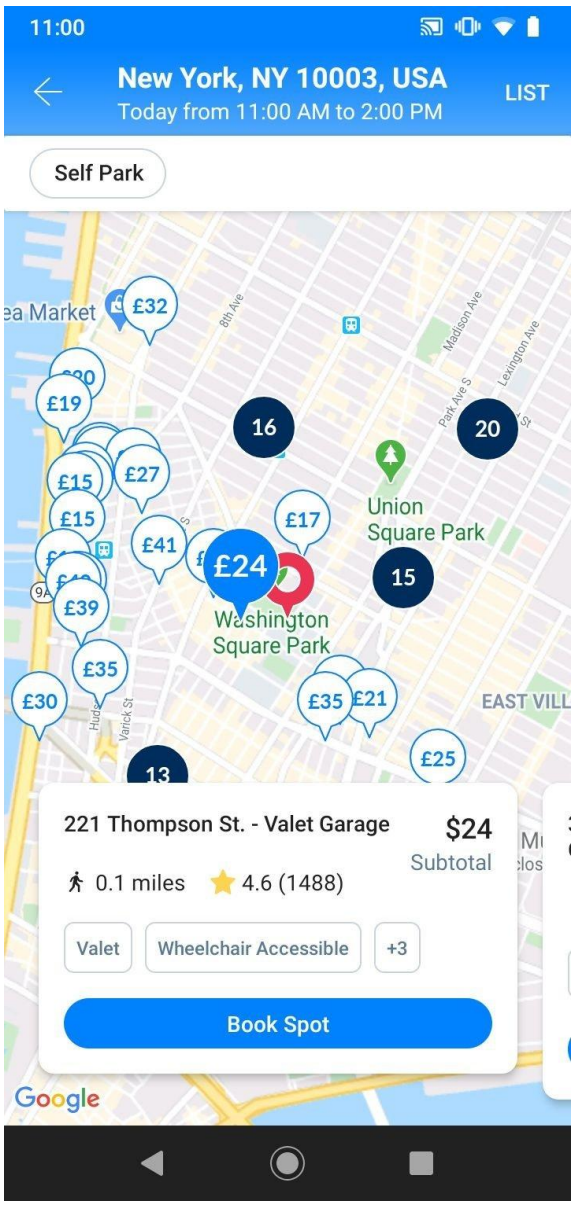

Рис. 4 – SpotHero

Загальний аналіз існуючих систем показує, що багато з них зосереджені на аспектах резервування парковочних місць і пропонують функції навігації та оплати. Однак, деякі недоліки включають невідомість доступності місць у режимі реального часу, нестійкість точності інформації та проблеми з оплатою.

1.2 Технологічні та наукові аспекти в сфері мобільних додатків для автомобілістів

У сучасному світі мобільні додатки для автомобілістів активно використовують передові технології та наукові розробки для створення інноваційних та ефективних рішень. Основні технологічні та наукові аспекти в сфері цих додатків включають[2]:

1. Геолокаційні сервіси:

Точне визначення місцезнаходження: Використання GPS технологій дозволяє додаткам точно та швидко визначати місцеположення автомобіліста, забезпечуючи актуальні та надійні дані для навігації та пошуку паркувальних місць.

2. Штучний інтелект:

Прогнозування доступності парковочних місць: Штучний інтелект використовується для аналізу великих обсягів даних, щоб прогнозувати доступність парковочних місць в різних районах та часах. Це дозволяє оптимізувати маршрути та полегшує пошук вільних місць[17].

3. Інтеграція з платіжними системами:

Легкість та безпека оплати: Мобільні додатки для автомобілістів інтегруються з різними платіжними системами, що дозволяє користувачам

легко та безпечно оплачувати паркування, резервувати місця та використовувати інші сервіси.

4. Мовні сервіси:

Мультиязичність для комфорту користувачів: Додатки підтримують українську та англійську мови, щоб надати користувачам максимальний комфорт та можливість вибору мови відповідно до їхніх вподобань.

Ці технологічні та наукові аспекти сприяють створенню інтелектуальних та користувацько-орієнтованих мобільних додатків для автомобілістів, які не тільки полегшують їхні щоденні поїздки, але й активно вирішують проблему пошуку та ефективного використання паркувальних місць[2].

РОЗДІЛ 2. ФУНКЦІОНАЛЬНИЙ АНАЛІЗ ВИМОГ

2.1 Функціональні вимоги мобільної інформаційної системи

Мобільна інформаційна система (МІС) для аналізу та оптимізації пошуку паркувальних місць має вирішувати ряд завдань, які відповідають потребам користувачів[2]. Функціональні вимоги включають:

- Навігація та оптимізація маршруту:

Визначення оптимального маршруту від поточного місцезнаходження до пункту призначення з урахуванням доступності паркувальних місць.

Можливість вибору різних критеріїв оптимізації: час, вартість, відстань.

- Пошук паркувальних місць:

Перегляд реального часу доступності парковочних місць на карті.

Система фільтрації для врахування уподобань користувачів (цінова чутливість, часова чутливість, чутливість до ходьби).

- Резервування та оплата:

Надає можливість резервування парковочних місць через додаток. Це дозволяє користувачам забезпечити гарантоване наявність парковки в потрібному місці та часі. Крім того, система інтегрується з платіжними системами, що дозволяє здійснювати безпечну та зручну оплату паркування[9].

- Мовні налаштування:

Підтримка української та англійської мов для забезпечення зручності користувачів у використанні системи.

Всі ці функції МІС спроектовані з метою забезпечити максимальну зручність та ефективність використання системи[2]. Завдяки їм користувачі можуть швидко та ефективно знаходити паркувальні місця, оптимізувати свій маршрут та зручно оплачувати паркування. Безумовно, це допоможе знайти розв'язання для багатьох проблем, пов'язаних з пошуком паркувальних місць.

2.2 Нефункціональні вимоги: продуктивність, безпека, зручність користування

- Продуктивність:

Система повинна забезпечувати швидкий доступ до інформації та реагування на запити користувачів у режимі реального часу[14].

Мінімальні часові затримки при розрахунках маршруту та відображенні інформації на мапі.

- Безпека:

Захист особистих даних користувачів, включаючи інформацію про місцезнаходження та дані оплати.

Використання шифрування для забезпечення безпеки передачі даних.

- Зручність користування:

Інтуїтивний і зрозумілий інтерфейс для користувачів різного рівня технічної підготовки[1].

Можливість налаштування основних параметрів системи зручно та швидко.

2.3 Аналіз потреб цільової аудиторії

Цільова аудиторія даної мобільної інформаційної системи включає в себе

водіїв, які активно користуються автомобілями у міському середовищі. Розуміння їхніх потреб та вимог є ключовим етапом у розробці функціоналу та інтерфейсу системи. Основні аспекти, які слід враховувати при створенні мобільної інформаційної системи (МІС) для паркування, включають наступне:

- Ефективний пошук парковочних місць:

Водії бажають максимально швидко та легко знаходити вільні парковочні місця, щоб оптимізувати свій час та зусилля. Система повинна надавати точні та актуальні дані щодо доступних місць, враховуючи фактори, такі як відстань до пункту призначення, цінові категорії та інші параметри.

- Приділення уваги вартості та часу:

Користувачі оцінюють можливість вибору опцій, які враховують їхні уподобання, чутливість до ціни, часу чи ходьби до місця призначення. Система повинна надавати різні варіанти паркування з відповідними відомостями щодо цін, часу призначення та інших параметрів.

- Зручність та безпека використання додатку:

Важливо мати інтуїтивно зрозумілий інтерфейс[14], який забезпечить простоту навігації та використання додатку. Крім того, можливість резервування та оплати паркування повинна бути забезпечена безпечно та без зайвих труднощів. Додаток повинен забезпечувати конфіденційність та захист особистої інформації користувачів.

Зрозуміння цих функціональних та нефункціональних вимог є вирішальним для створення МІС, яка відповідає потребам користувачів та забезпечує їм зручний, ефективний та безпечний досвід використання. Це дозволить системі ефективно взаємодіяти з водіями та вирішувати основні завдання, спрямовані на поліпшення процесів пошуку та оптимізації паркувальних місць.

2.4 Огляд актуальності проблеми дослідження.

Паркування в містах стає надзвичайно актуальною та серйозною проблемою, і це є наслідком кількох ключових факторів. Зростання автомобільного парку, обмежений фізичний простір та недостатня розвиненість паркувальної інфраструктури ускладнюють ефективне управління паркувальними ресурсами та підвищують рівень неспроможності існуючих систем.

- Фактори, що погіршують ситуацію:

Зростання автопарку: Зі збільшенням кількості автомобілів в місті виникає невідповідність між попитом на паркувальні місця та їх реальною кількістю. Це призводить до появи надмірної конкуренції за паркувальні зони та збільшення часу пошуку вільного місця.

Обмежений фізичний простір: Міська територія обмежена, існує обмежена кількість доступного місця для паркування. Це зумовлює потребу у вдосконаленні систем паркування та раціональному використанні обмежених ресурсів.

Недостатня розвиненість паркувальної інфраструктури: Нерозвинена інфраструктура паркування ускладнює управління та взаємодію між водіями та паркувальними зонами. Відсутність ефективних систем регулювання та контролю може спричинити хаотичне використання паркувальних місць.

- Основні проблеми паркування в містах:

Нестача паркувальних місць: Значний ріст автопарку породжує дефіцит паркувальних місць, що призводить до ситуації, коли водії мають обмежений вибір інтересуючих їх зон для залишення автомобілів.

Транспортні затори: Пошук паркувального місця в містах зазвичай супроводжується безліччю маневрів та об'їздів, що веде до утворення

транспортних заторів. Це негативно впливає на рух транспорту та збільшує час подорожей.

Забруднення повітря: Непослідовне паркування призводить до зайвих викидів шкідливих речовин, що сприяє погіршенню якості повітря в місті та може мати серйозні наслідки для здоров'я громадян.

Ці проблеми створюють актуальну необхідність у впровадженні інноваційних та ефективних рішень, таких як мобільні інформаційні системи для аналізу та оптимізації пошуку паркувальних місць.

2.5 Сучасні підходи до оптимізації паркування

Оптимізація паркування в сучасних умовах вимагає інноваційних технологій та продуманих стратегій для вдосконалення управління та оптимізації використання паркувальних ресурсів[15]. Зазначені підходи репрезентують лише частину великого спектру інструментів, що допомагають у подоланні проблем паркування в містах.

- Сенсорні технології:

Використання сучасних сенсорів грає критичну роль у покращенні доступності та ефективності паркувальних зон. Сенсори можуть моніторити стан кожного паркувального місця, надсилаючи дані про його використання та статус в реальному часі. Це дозволяє системам оптимізації розподілу місць реагувати на зміни та забезпечувати максимальну доступність.

- Інтелектуальні системи управління:

Розробка інтелектуальних систем управління враховує широкий спектр факторів, таких як трафік, попит, часові розподіли та інші. Вони використовують алгоритми машинного навчання та аналізу даних для розуміння та передбачення патернів поведінки водіїв[15]. Такі системи

дозволяють ефективно розподіляти паркувальні ресурси, мінімізуючи час пошуку водіями та зменшуючи транспортні затори.

- Електронні платформи для резервування:

Створення мобільних додатків для резервування паркувальних місць стає дедалі популярнішим рішенням. Водії можуть заздалегідь зарезервувати парковку, обираючи зручний час та регіон. Це допомагає уникнути стресів та невизначеності при пошуку місця.

- Розвиток "розумних" паркувальних зон:

Використання технологій Інтернету речей (IoT) для створення "розумних" паркувальних зон відкриває широкі перспективи[13]. Датчики, розташовані в парковках, можуть взаємодіяти між собою та з мережею, адаптуючись до змін ситуації. Наприклад, система може автоматично резервувати місця для власників електромобілів або перерозподіляти простір у разі надзвичайних подій.

Ці сучасні підходи відкривають нові можливості для подолання проблем паркування та покращення загального транспортного досвіду в містах.

2.6. Мобільні інформаційні системи у транспортній сфері.

Мобільні інформаційні системи у транспортній сфері є необхідною складовою для раціонального використання транспортних засобів та поліпшення загального транспортного досвіду. Ці системи впроваджують інноваційні технології, спрямовані на підвищення мобільності та оптимізацію транспортних процесів[16].

1. Навігаційні додатки:

Навігаційні додатки визначають оптимальні маршрути для водіїв, враховуючи різні фактори, такі як трафік, дорожні роботи та події на дорозі. Вони не лише допомагають уникнути транспортних заторів, але й надають інформацію щодо паркування, вказівки на доступні парковочні зони та сервіси, що покращують загальний комфорт подорожей.

2. Системи сповіщення:

Ці системи інформують водіїв про наявність вільних паркувальних місць, надаючи реально-часову інформацію через мобільні додатки або SMS-повідомлення. Окрім цього, вони можуть надавати рекомендації щодо оптимального часу для пошуку паркувального місця та інші корисні поради.

3. Додатки для управління транспортом:

Системи управління транспортом дозволяють відстежувати рух транспортних засобів у реальному часі[16]. Вони надають інформацію про розташування автобусів, таксі чи інших транспортних засобів, що допомагає водіям планувати свої маршрути та уникати зайвого чекання.

4. Інтеграція з громадським транспортом:

Додатки, які поєднують інформацію про паркування та громадський транспорт, дозволяють водіям зручно комбінувати різні види транспорту для максимально ефективного переміщення в місті. Це сприяє підвищенню доступності та зручності для користувачів громадського та приватного транспорту.

Цей огляд наводить лише кілька прикладів того, як мобільні інформаційні системи в транспортній сфері активно сприяють поліпшенню мобільності та підвищенню ефективності управління транспортними процесами. Зазначені

рішення виступають важливим елементом у стратегії оптимізації транспортної інфраструктури та покращення якості життя мешканців міст.

2.7 Опис систем та технологій, використовуваних у мобільних інформаційних системах для паркування.

У сучасних мобільних інформаційних системах для паркування використовується ряд технологій та систем для забезпечення ефективного та зручного користування. Основні компоненти та технології включають[17]:

- Сенсорні технології:

Системи використовують сенсори, розташовані на паркувальних місцях, щоб визначити їхню доступність та стан. Сенсори можуть бути вбудовані у дорожне покриття, або вони можуть використовувати оптичні та акустичні технології для виявлення наявності автомобіля.

- Системи глобального позиціонування (GPS):

Використання GPS дозволяє точно визначати місцезнаходження автомобілів і визначати їхнє розташування на мапі парковки. Це полегшує водіям знаходження вільних парковочних місць та надає інформацію про оптимальний шлях до них.

- Інтернет речей (IoT):

Використання IoT дозволяє реальному часі взаємодіяти між сенсорами, аналітичними системами та іншими компонентами. Це створює інтелектуальні паркувальні системи, які можуть адаптуватися до змін ситуації та оптимізувати використання місць.

- Мобільні додатки:

Використання мобільних додатків дозволяє водіям отримувати реально-часову інформацію про доступні парковочні місця, резервувати їх, а також оплачувати паркування. Мобільні додатки можуть використовувати технології нотифікацій для надання важливої інформації.

2.8 Алгоритми аналізу та оптимізації паркувальних місць.

Алгоритми, які визначають ефективний та оптимальний спосіб використання парковочних місць, ґрунтуються на важливому комплексному підході до обробки інформації. В основі цього підходу стоять ряд ключових алгоритмічних елементів, які спільно працюють для покращення доступності та ефективності паркування[2]:

- Машинне навчання:

Використання алгоритмів машинного навчання виявляється важливим елементом для аналізу та прогнозування звичайних патернів паркування. Система, за допомогою навчання на великій кількості даних, може розпізнавати тенденції та пристосовуватися до змін у звичках водіїв. Наприклад, вона може прогнозувати популярні місця для паркування у певний час доби чи враховувати попит під час заходів в місті.

- Аналіз трафіку:

Алгоритми аналізу трафіку враховують потоки транспорту та руху на парковці. Це дозволяє системі ефективно розподіляти парковочні місця відповідно до інтенсивності руху, уникати заторів і забезпечувати доступність місць у найбільш завантажених часах.

- Оптимізація маршрутів:

Алгоритми оптимізації маршрутів враховують не лише фактор доступності парковочних місць, але й намагаються підібрати оптимальний маршрут для водіїв. Це може включати оцінку відстані, часу подорожі, вартості паркування та інших факторів, що впливають на вибір маршруту.

Використання цих алгоритмів дозволяє створювати інтелектуальні системи паркування, які динамічно реагують на зміни у середовищі та потреби користувачів. Такий підхід сприяє максимальній ефективності управління парковочними місцями, забезпечуючи водіям швидкий та зручний доступ до необхідного паркувального простору.

РОЗДІЛ 3. ПРОЄКТУВАННЯ МОБІЛЬНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Принципи побудови користувацького інтерфейсу мобільної додаткової системи.

Інтуїтивний дизайн – це не лише естетичний аспект, але й стратегічна концепція, спрямована на максимальне спрощення взаємодії користувача з мобільною інформаційною системою для паркування[11]. В основі цього принципу лежить:

- Легкість Орієнтації:

Спрощення навігації через чітку структуру та розташування ключових елементів на інтерфейсі. Водії повинні швидко розуміти, як взаємодіяти з додатком та виконувати необхідні завдання.

- Мінімалізація Кліків:

Максимізація доступності функцій за декілька кліків. Це допомагає водіям ефективно виконувати операції, такі як пошук, резервування та оплата паркування, з економією їхнього часу.

- Інтуїтивні Символи та Зображення:

Використання зрозумілих ікон та графічних елементів для позначення функцій, що допомагає водіям легко розпізнавати та використовувати їх.

- Реалізація функціоналу відповідно до потреб:

Виправлення потреб користувачів - це ключовий етап при створенні мобільної інформаційної системи для паркування. Це включає:

- Стратегічне Розміщення Функцій:

Позиціонування ключових функцій, таких як пошук вільних місць, резервування та оплата паркування, на видимих місцях інтерфейсу, де вони легко доступні.

- Персоналізація Можливостей:

Надання можливостей налаштувань для користувачів, які дозволяють їм вибирати опції відповідно до власних уподобань, таких як ціна, час та відстань до місця паркування.

- Підтримка Мультимедіа та Інтерактивності:

Використання мультимедійних елементів та інтерактивності робить взаємодію з додатком не лише корисною, але й цікавою:

- Візуалізація Даних:

Використання діаграм, графіків та інших візуальних елементів, що полегшують розуміння статистики та інформації про паркування.

- Інтерактивні Сповіщення:

Надсилання сповіщень у формі зрозумілих повідомлень, а також можливість взаємодії з ними для подальшої інформації або виконання конкретних дій.

Ці принципи і технології не лише роблять систему ефективною, але й гарантують задоволення потреб користувачів, забезпечуючи їм комфорт[11], зручність та приємний досвід використання мобільної інформаційної системи для паркування.

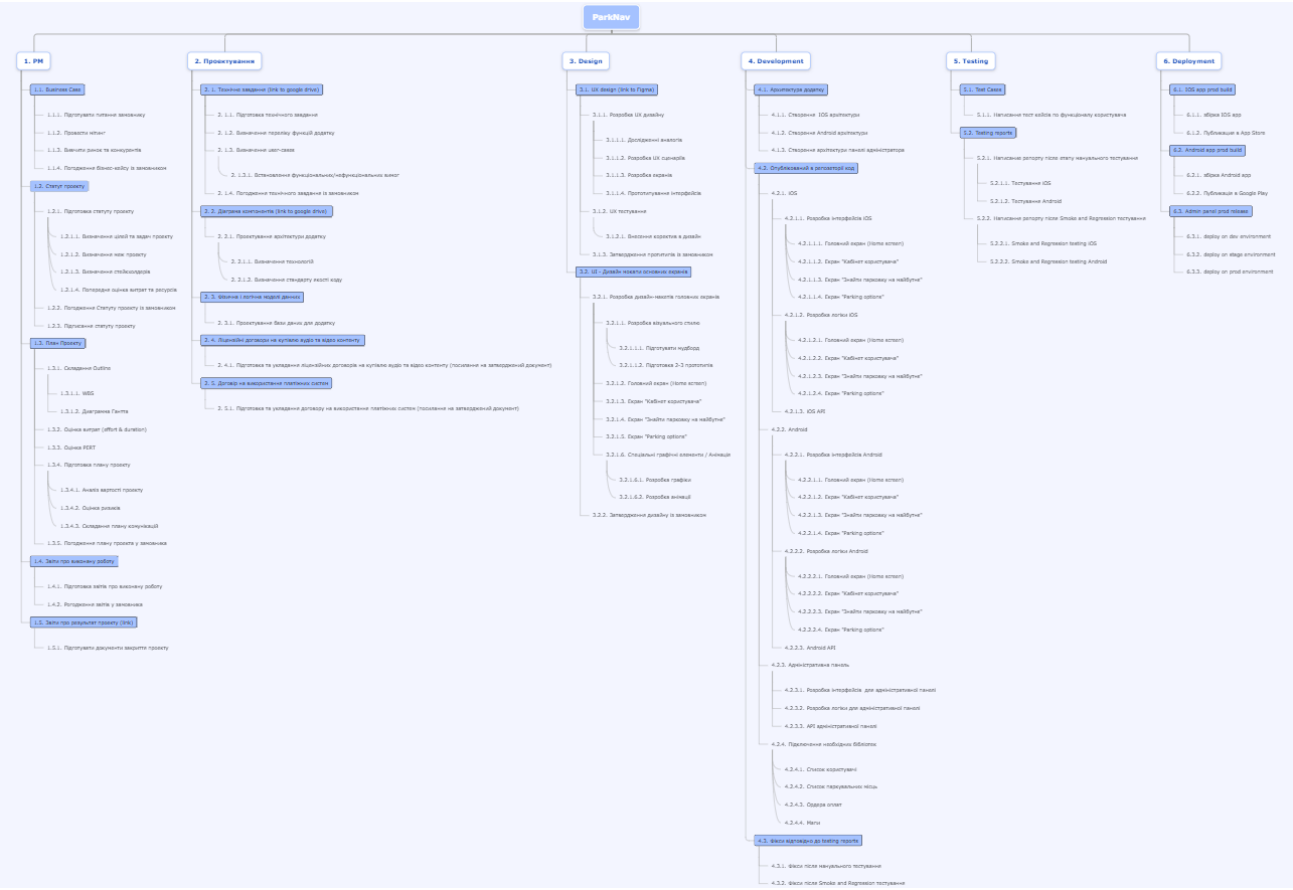

Рис. 5 – Життєвий шлях проєкту

3.2 Вибір платформи та технологій.

Для реалізації ефективної та високопродуктивної мобільної інформаційної системи було вибрано наступні платформи та технології:

3.2.1 Операційна система

Додаток розробляється для операційних систем Android[10] для максимального охоплення аудиторії.

Android - операційна система для мобільних пристроїв, розроблена компанією Google. Вона була представлена у вересні 2008 року та стала однією з найпопулярніших мобільних платформ у світі. Android базується на ядрі Linux та має відкритий вихідний код, що дозволяє розробникам адаптувати та розширювати систему під свої потреби.

Архітектура:

- Linux Kernel: В основі Android знаходиться ядро Linux, яке відповідає за низькорівневі операції пристрою та взаємодію з апаратною частиною.
- Операційна Система Android: Включає в себе сервіси та API для взаємодії з пристроєм, а також для виконання високорівневих завдань.
- Додатки: Android додатки використовують Java або Kotlin для програмування. Кожен додаток працює в окремому процесі та має власні ресурси та файли.
- Application Framework: Надає набір інструментів та сервісів для спрощення розробки додатків, включаючи управління життєвим циклом додатку, графічний інтерфейс, мережеві запити та багато іншого.

Розширені можливості:

- Підтримка Різних Пристроїв: Android працює на різних пристроях, від смартфонів та планшетів до телевізорів та автомобільних систем.
- Система Сповіщень: Користувачі можуть легко отримувати та управляти сповіщеннями, які стосуються їхніх додатків та системи.
- Спільність та Інтеграція: Android забезпечує можливість спільної роботи між додатками та інтеграцію з іншими сервісами, такими як Google Drive, Gmail, Google Maps тощо.

Android[10] є важливою складовою сучасного мобільного ландшафту, надаючи користувачам та розробникам різноманітні можливості та інструменти для розробки та використання мобільних додатків.

3.2.2 Мова програмування

Розробка додатку відбувається за допомогою використання Kotlin[5] для

розробки для платформи Android для забезпечення ефективності та оптимальності коду.

Kotlin - це статично типізована мова програмування, яка була розроблена компанією JetBrains і вперше представлена в 2011 році. Однак вона набула особливої популярності як офіційна мова розробки для Android у 2017 році. Kotlin[5] є мовою, яка працює на віртуальній машині Java (JVM) та може бути використана для розробки різноманітних застосунків.

Особливості:

- Сумісність з Java: Kotlin повністю сумісна з Java, що означає, що код Kotlin може бути легко інтегрований в існуючі проекти на Java, і навпаки.
- Безпечність та зручність: Мова дозволяє розробникам уникнути багатьох типових помилок завдяки системі статичного типізування та інтелектуальним функціям.
- Коротший синтаксис: Kotlin надає коротший та більш експресивний синтаксис порівняно з Java, що сприяє швидшому написанню коду та покращує читабельність.
- Null-безпечність: Kotlin вбудовує концепцію null-безпечності, що допомагає уникнути помилок, пов'язаних із значеннями null.
- Розширення функціональності: Kotlin включає в себе функціональні конструкції, такі як лямбда-вирази та вирази вищих порядків, що полегшують написання функціонального коду.

Kotlin став важливою альтернативою для Java у світі розробки, завдяки своїм функціональним можливостям, безпечності та підтримці великою спільнотою.

3.2.3 Фреймворк

Використання фреймворку Flutter[6] для забезпечення

кросплатформенності та зменшення часу розробки.

Flutter - це відкритий кросплатформений фреймворк для розробки мобільних додатків, який був розроблений компанією Google. Основною його особливістю є можливість створення однаково виглядаючих та працюючих додатків для різних платформ, таких як Android та iOS, використовуючи єдину кодову базу[6].

Особливості:

- Кросплатформенність: Однаковий код може бути використаний для створення додатків для різних платформ, що спрощує розробку та підтримку.
- Гаряче Перекомпілювання: Гаряче перекомпілювання дозволяє швидко внесення змін у код та миттєво оновлювати інтерфейс додатка без перезапуску.
- Багатий Візуальний Дизайн: Flutter має потужний набір вбудованих віджетів та можливостей для створення красивого та різноманітного інтерфейсу користувача.
- Доступність Усіх API Платформ: Flutter дозволяє взаємодіяти з API платформи напряму, що робить його гнучким та забезпечує повний контроль.
- Широкий Вибір Плагінів: Flutter має велику спільноту, що розробляє різноманітні плагіни для спрощення роботи з різними функціями та сервісами.
- Документація та Спільнота: Google активно підтримує Flutter, надаючи високоякісну документацію та широку спільноту розробників, що полегшує навчання та розвиток.
- Інтеграція з Firebase: Проста інтеграція з Firebase забезпечує розробників потужними інструментами для збереження даних,

аутентифікації та інших сервісів.

Flutter став популярним вибором для розробників завдяки своїй продуктивності, гнучкості та можливості швидкого впровадження змін. Завдяки активному розвитку та підтримці з боку Google, Flutter продовжує набирати популярність серед розробників мобільних додатків.

3.2.4 База даних

Використання Firebase Realtime Database[8] для забезпечення реального часу та масштабованості бази даних.

Firebase Realtime Database - це хмарна база даних від Google, яка призначена для ефективного зберігання та синхронізації даних в реальному часі. Вона розроблена для роботи в різних платформах, включаючи Android, iOS та веб-додатки, і надає розробникам простий та потужний механізм для забезпечення доступу до даних у режимі реального часу[4].

Основні характеристики:

- Реальний час: Одна з ключових особливостей Firebase Realtime Database - це надання можливості синхронізації даних в реальному часі між всіма підключеними пристроями. Це дозволяє отримувати оновлення негайно після будь-яких змін у базі даних.
- JSON-подібна Структура Даних: Дані зберігаються у вигляді JSON-подібних об'єктів, що робить їх легко розуміти та обробляти.
- Широкий Доступ: Забезпечується доступ до бази даних через різні платформи та мови програмування, включаючи JavaScript, Java, Objective-C, Swift, і інші.
- Офлайн Режим: Firebase Realtime Database працює в офлайн-режимі, дозволяючи користувачам взаємодіяти з даними навіть без активного Інтернет-з'єднання. При наявності мережі всі зміни автоматично синхронізуються.

- Автентифікація та авторизація: Інтегрується з іншими службами Firebase для забезпечення безпеки даних через механізми аутентифікації та авторизації користувачів[12].

Використання в розробці:

- Мобільні Додатки: Firebase Realtime Database широко використовується для розробки мобільних додатків, зокрема на платформах Android та iOS.
- Веб-Додатки: Розробники можуть використовувати Firebase Realtime Database для зберігання та синхронізації даних у веб-додатках.
- Ігрова Розробка: В ігровій індустрії ця база даних дозволяє створювати мультиплеерні та онлайн ігри з реальним часом.

Інтеграції:

- Інші Firebase Сервіси: Firebase Realtime Database легко інтегрується з іншими сервісами Firebase, такими як Firebase Authentication, Firebase Cloud Functions, Firebase Hosting та іншими.
- API та SDK: Надається багатомовний API та SDK для різних мов програмування та платформ.

Безпека та масштабованість:

- Шифрування Даних: Дані в Firebase Realtime Database шифруються за допомогою SSL, щоб забезпечити безпеку передачі даних.
- Масштабованість: Firebase Realtime Database автоматично масштабується для виконання потреб вашого додатка, навіть при збільшенні обсягу даних та трафіку.

Firebase Realtime Database визначається простотою використання, реальним часом та широким спектром можливостей, що роблять його популярним серед розробників, особливо в галузі мобільних та веб-розробок[8].

3.3 Архітектура системи

Архітектурна модель системи базується на концепції клієнт-сервер[15]. **Основні компоненти включають:**

- Клієнтська частина: Реалізована за допомогою Flutter[6], забезпечує інтерфейс користувача та взаємодію з сервером.
- Сервер: Використання сервера Firebase для збереження та обробки даних, забезпечення автентифікації користувачів та обробки запитів в реальному часі.
- Бізнес-логіка: Реалізована на сервері та частково на клієнтській стороні для оптимізації продуктивності та відгуку системи[3].

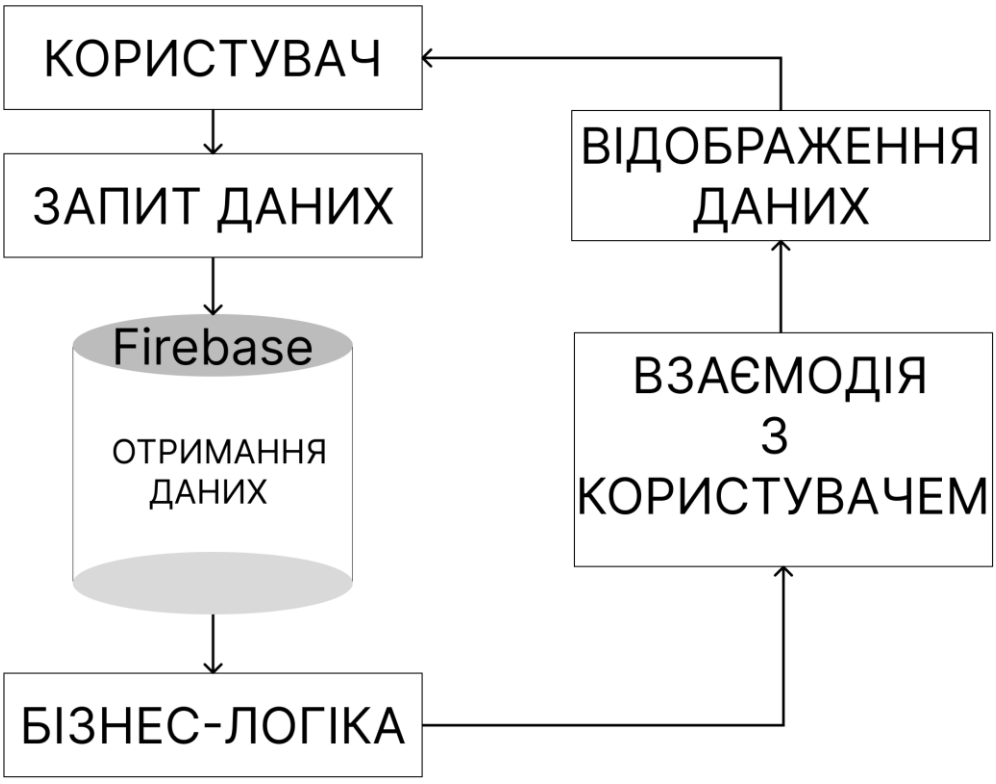

Рис. 6 – Архітектурна модель

3.4 Проектування інтерфейсу користувача

Дизайн інтерфейсу користувача розробляється з урахуванням простоти

використання та ефективного взаємодії користувача з додатком[14]. Основні елементи включають:

- Картографічний інтерфейс: Відображення мапи з маркованими парковочними місцями та оптимальним маршрутом.
- Фільтри та упорядкування: Можливість вибору опцій фільтрації за ціною, часом, відстанню тощо.
- Оплата та резервування: Зручні форми для оплати та резервування парковочних місць.

3.5 База даних та зв'язок із сервером

Firebase Realtime Database[8] став ключовим елементом для досягнення ефективної та надійної функціональності системи. Ця технологія забезпечує надійний механізм зберігання та синхронізації користувацьких даних, що є критичним для забезпечення послідовності та відповідності інформації у реальному часі.

Завдяки Firebase Realtime Database можна забезпечити негайну передачу та оновлення даних між пристроями користувачів та сервером. Це дозволяє користувачам додатку отримувати актуальну інформацію про парковочні місця, резервації та інші важливі аспекти системи в реальному часі.

Використання Firebase Realtime Database гарантує, що дані зберігаються в безпечному форматі, забезпечуючи високий рівень доступності та захист від втрати інформації. Це важливо для надання зручної роботи для кінцевих користувачів, які можуть швидко та ефективно використовувати додаток для пошуку, резервації та оптимізації паркувальних місць.

Firebase Realtime Database допомагає підтримувати високий стандарт якості продукту, гарантуючи, що дані завжди актуальні, доступні та безпечно збережені.

РОЗДІЛ 4. ТЕСТУВАННЯ МОБІЛЬНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

В ході реалізації мобільної інформаційної системи для аналізу та оптимізації пошуку паркувальних місць для водіїв були розроблені та імплементовані ключові компоненти системи.

4.1 Авторизація

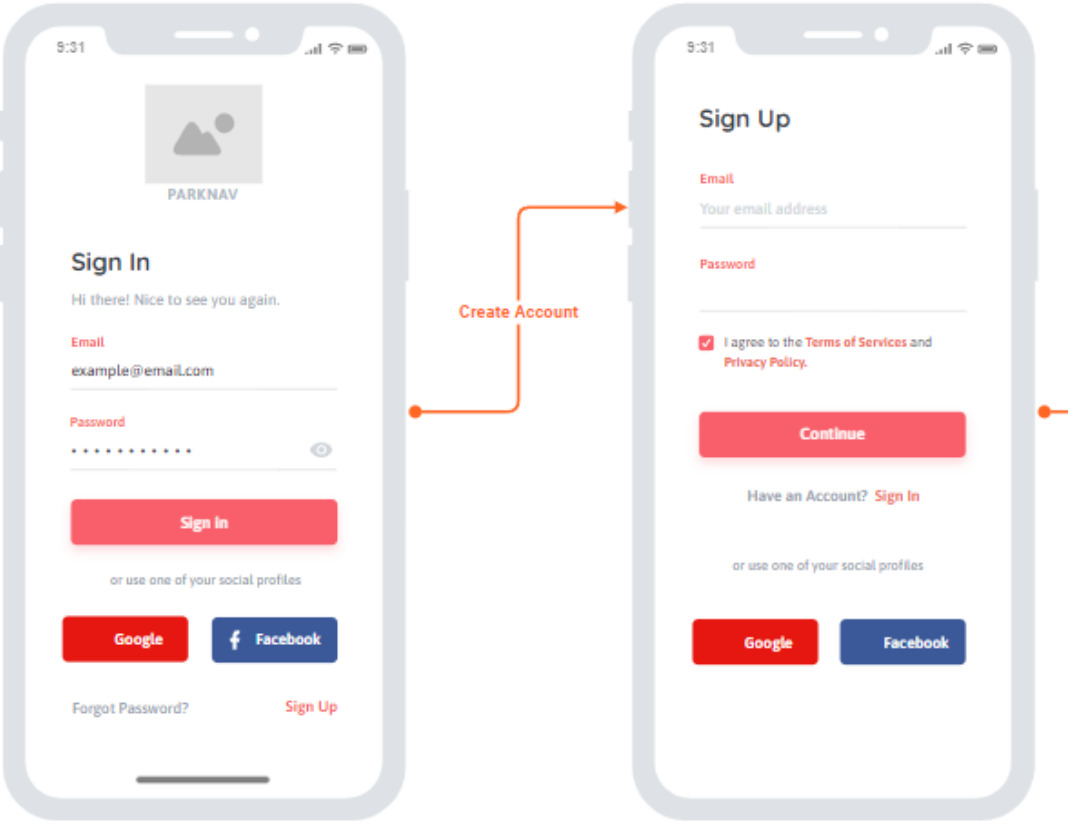

Рис. 7 – Вікно авторизації

4.2 Меню

Користувач має широкий спектр можливостей управління та використання додатку. Сервіс надає йому великий функціонал, що включає в себе редагування особистого профілю, перегляд історії своїх поїздок та паркувань, додавання та управління платіжними засобами. Крім того, користувач може налаштовувати

різноманітні параметри додатку для оптимізації його використання відповідно до особистих уподобань.

Зокрема, можливість редагування профілю дозволяє користувачу змінювати особисті дані, такі як фотографія, контактна інформація, марка авто.

Перегляд історії поїздок і паркувань надає можливість аналізувати минулі дії та витрати, що є корисним для ведення особистого обліку та планування майбутіх маршрутів.

Додавання та керування платіжними засобами дозволяє користувачеві легко та безпечно здійснювати оплату за паркування через додаток.

Налаштування додатку включає в себе різні опції, такі як мова інтерфейсу, повідомлення та інші параметри, які дозволяють користувачеві налаштовувати додаток відповідно до його особистих вподобань та потреб.

Розділ підтримки надає користувачеві можливість звертатися за допомогою, вирішення проблем чи отримання відповідей на питання, що сприяє покращенню загального досвіду використання додатку.

Рис. 8 – Меню застосунку

4.3 Домашня сторінка

Домашня сторінка демонструє карту та поточне розташування користувача на ній.

Режим “Показати парковку” - показує найближчу парковку до розташування водія.

Кнопка “Фільтри” - дає змогу обрати та налаштувати пошук парковки з урахуванням уподобань користувача. Одразу після вибору парковки водієм, він отримує напрямок в якому необхідно рухатись (GET DIRECTION)

Коли водій наблизився до свого парковочного місця програма йому пропонує запаркуватись (PARK IN HERE)

Після підтвердження паркування система показує інформацію про парковку: безкоштовний термін паркування, ціна після закінчення безкоштовного часу, таймер зі зворотнім відліком для безоплатної парковки або загальний час паркування для платних парковок.

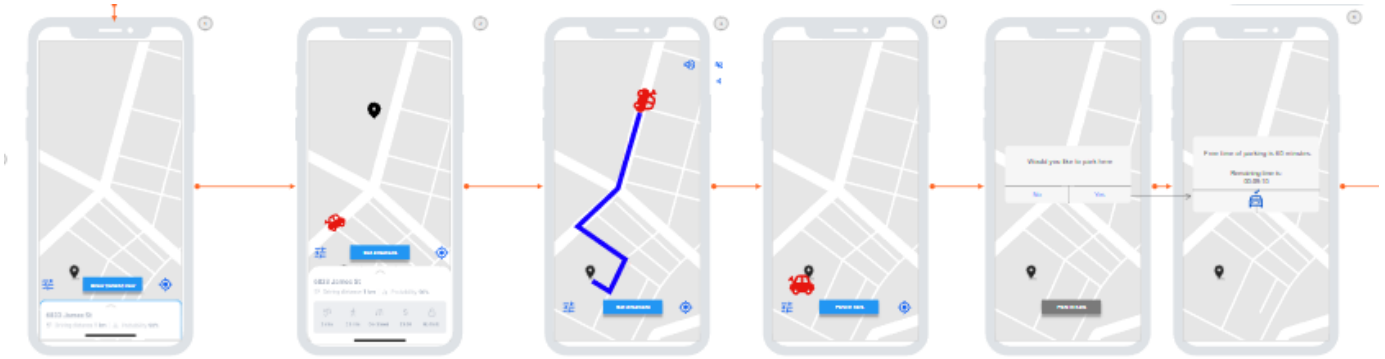

Рис. 9 – Домашня сторінка та її функціональність

4.4 Оплата та завершення паркування

Після оплати користувач отримує повідомлення про статус оплати Success / Try again. В разі успішної оплати користувач отримує QR-code для контролю на парковочному місці.

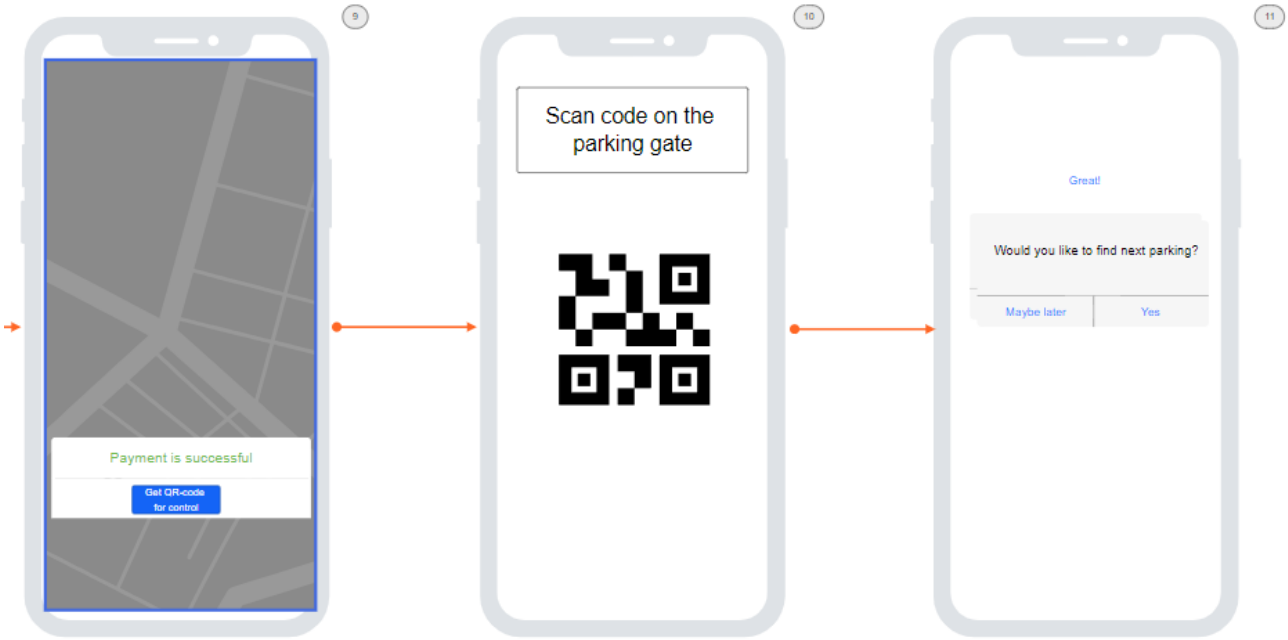

Рис. 10 – Оплата парковки

Коли водій хоче завершити паркування або покинути парковку, система показує йому PARK OUT:

- Якщо парковка платна та він перевищив безоплатний термін перебування на ній, система показує вартість яку треба оплатити та переводить його на сервіс оплати, який він вказав в Settings, Payments.
- При безоплатній парковці водій тисне PARK OUT та покидає місце парковки.

Рис. 11 – Завершення часу паркування

Режим “Plan journey” - дозволяє заздалегідь забронювати паркувальне місце для майбутньої поїздки.

Кнопка “Фільтри” - дає змогу обрати та/або налаштувати пошук паркувального місця для його подальшого бронювання з урахуванням уподобань користувача, шляхом заповнення інформації щодо: місця початку маршруту, місця призначення, дати, часу та іншої додаткової інформації, необхідної для пошуку місця для бронювання.

Після обрання відповідних критеріїв та натисканні кнопки “Done” користувач отримує повідомлення про підтвердження бронювання місця.

У разі підтвердження бронювання система видає йому Booking is successful.

Далі коли водій хоче розпочати дорогу до забронюваного місця він отримує напрямок в якому необхідно рухатись GET DIRECTION.

ВИСНОВОК

У кваліфікаційній дипломній роботі було розглянуто та розроблено мобільний додаток для пошуку та резервування парковочних місць, спрямований на оптимізацію процесу паркування у містах. Зазначеній меті було досягнуто завдяки використанню мови програмування Kotlin та технологій Firebase для реалізації різноманітних функціональних можливостей.

У розділі 1 роботи проведено аналіз проблем паркування та визначено основні завдання дослідження. Висвітлено мету та об'єкт дослідження, а також актуальність теми у контексті проблем міської мобільності. Представлено методи дослідження, серед яких огляд існуючих мобільних додатків для пошуку парковок, розгляд технологічних аспектів розробки та вивчення алгоритмів оптимізації маршрутів.

У розділі 2 проведено аналіз існуючих мобільних додатків для пошуку парковок, проаналізовано їх переваги та недоліки. Розглянуто Google Maps, який відомий своєю широкою функціональністю та глобальним охопленням. Також розглянуто ParkMobile, який спрощує оплату паркування та резервування місць, а також SpotHero, який дозволяє знаходити та резервувати паркомісця у великих містах. Крім того, проведено аналіз Waze, який використовує дані користувачів для оптимізації маршрутів та інформації про дорожні умови.

У розділі 3, представлено розроблений мобільний додаток, включаючи його функціональність та можливості, а також використані технології та інтеграції. Враховано чутливість користувача до ціни, часу та ходьби при виборі оптимального маршруту.

У розділі 4 надано висновки щодо розробленого додатку, його переваг та обмежень. Зазначено, що додаток може значно полегшити процес пошуку та резервування парковочних місць для користувачів, що дозволяє ефективно

використовувати час та зменшувати стреси, пов'язані з паркуванням у місті. Також, розглянуто питання оновлень та підтримки.

Загалом, розроблений мобільний додаток має потенціал вирішувати проблеми паркування у містах та полегшувати життя водіям, забезпечуючи зручні та ефективні можливості використання парковочних місць.

СПИСОК ЛІТЕРАТУРИ

1. "Mobile Cloud Computing: Models, Implementations, and Applications" by Meikang Qiu, Wenyun Dai, Keke Gai, 2017.с. 23 – 76.

2. "Mobile Computing: Technology, Applications, and Service Creation" by Talukder and Yavagal, 2007.с. 94 – 157.

3. "Mobile Application Development, Usability, and Security" by Subhendar M., 2016.с. 45 – 105.

4. "Location-Based Services and Geo-Information Engineering" by Allan Brimicombe, 2015.с. 120 – 156.

5. "Kotlin Programming: The Big Nerd Ranch Guide" by Josh Skeen, David Greenhalgh, 2019.с. 98 – 178.

6. "Flutter in Action" by Eric Windmill, 2019.с. 65 – 88.

7. "Google Maps JavaScript API Cookbook" by Alper Dincer, Balkan Uraz, 2013.с. 114 – 165.

8. "Firebase Essentials - Android Edition: Kickstart your Android app development with Firebase" by Neil Smyth, 2016.с. 173 – 202.

9. "Mobile Payment Systems: Secure Network Architectures and Protocols" by Wen Chen Hu, 2015.с. 34 – 69.

10. "Android Programming: The Big Nerd Ranch Guide" by Bill Phillips, Chris Stewart, Kristin Marsicano, 2017.с. 145– 187

11. "The Art of Mobile App Security" by Neil Bergman, Daniel R. Cid, 2014.с.144 – 168.

12. "Android Security Internals: An In-Depth Guide to Android's Security Architecture" by Nikolay Elenkov, 2014.с. 154 – 165.

13. "Human-Computer Interaction: An Empirical Research Perspective" by I. Scott MacKenzie, 2012.с.156 – 187.

14. "Designing Mobile Interfaces: Patterns for Interaction Design" by Steven

Hoober, Eric Berkman, 2011.c.14 – 37.

15. "Effective UI: The Art of Building Great User Experience in Software" by Jonathan Anderson, John McRee, Robb Wilson, 2010.c.145 – 187.

16. "User Interface Design and Evaluation" by Debbie Stone, Caroline Jarrett, Mark Woodroffe, Shailey Minocha, 2005.c. 34 – 97.

17. "The Mobile Application Hacker's Handbook" by Dominic Chell, Tyrone Erasmus, Shaun Colley, Ollie Whitehouse, 2015.c. 41 – 84.

18. ParkMobile Official Website – 2023 – URL: https://parkmobile.io/

19. Waze Official Website – 2023 – URL: https://www.waze.com/ua/live-map/

20. SpotHero Official Website – 2023 – URL: https://spothero.com/

ДОДАТОК А

```
package `in`.parknav.application

import `in`.parknav.components.account.AccountPreference
import `in`.parknav.components.api.models.guard.ParamGuard
import `in`.parknav.components.api.models.parking.area.ParamParkingArea

class AppAccount {



    private var mParkingArea: ParamParkingArea? = null
    private var mGuard: ParamGuard? = null

    private var mParkingAreaId: String? = null
    private var mGuardId: String? = null



    private var isPreferenceLoaded = false

    fun hasAccount(): Boolean {
        return mGuard!=null
    }

    fun reset() {
        AccountPreference.getInstance().clearPrefs()
        mGuard = null
        mGuardId = null
```

```
        mParkingAreaId = null
    }

    fun getGuard(): ParamGuard? {

        return mGuard
    }



    fun getGuardId(): String?{

        if(!isPreferenceLoaded){
            loadPreference()
        }
        return mGuardId
    }

    fun setGuard(guard: ParamGuard): Boolean {
//        info { "Guard: $guard" }
        mGuard = guard
        mParkingAreaId = guard.parkingAreaId

        AccountPreference.getInstance().setGuardId(mGuard?.id)

AccountPreference.getInstance().setParkingAccountId(mGuard?.parkingAccountId)
        AccountPreference.getInstance().setParkingAreaId(mGuard?.parkingAreaId)
```

```
    loadPreference()
    return true
}

fun setParkingArea(parkingArea: ParamParkingArea?) {
    mParkingArea = parkingArea
}

fun getParkingArea(): ParamParkingArea? {
    return mParkingArea
}

fun getParkingAreaId(): String? {

    if(!isPreferenceLoaded){
        loadPreference()
    }
    return mParkingAreaId
}

private fun loadPreference(){
   mGuardId =  AccountPreference.getInstance().getGuardId()
    mParkingAreaId =  AccountPreference.getInstance().getParkingAreaId()
    isPreferenceLoaded = true

}
```

```
}

    const val DATE_FORMAT_HOUR = "h:mm aa"

    const val FORMAT_DATE_READABLE = "dd MMM yyyy"

    const val FORMAT_DATE_READABLE2 = "dd/MM h:mm a"

    fun getFormattedDateTime(time: Long, template: String, context: Context): String
{
        var locale = getCurrentLocale(context)
        if(locale==null){
            Locale.getDefault()
        }
        val localizedPattern = getLocalizedPattern(template, locale!!)

        return SimpleDateFormat(localizedPattern, locale).format(Date(time))
    }

    fun getLocalizedPattern(template: String, locale: Locale): String? {
        return DateFormat.getBestDateTimePattern(locale, template)
    }

    fun getIsoFormattedDate(context: Context, date: Date): String {

//        info { "getIsoFormattedDate: $date" }

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
```

```
            val localDate = date.toInstant().atZone(ZoneId.systemDefault())
//.toLocalDateTime()
//            info { localDate.toString() }
            val formatter = DateTimeFormatter.ISO_INSTANT
            val text = localDate.format(formatter)
            return text
        } else {
            val sdf = SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ssZ",
getCurrentLocale(context))
            val formattedDate: String = sdf.format(Date())
            return  formattedDate
        }


    }

    fun getDateFromFormattedString(context: Context, dateString: String): Date {

//        info { "getIsoFormattedDate: $date" }

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val localDate = Instant.parse(dateString) //.atZone(ZoneId.systemDefault())
//.toLocalDateTime()
//            info { localDate.toString() }
//            Instant.now().
            val formatter = DateTimeFormatter.ISO_INSTANT
```

```
            val date = Date(localDate.toEpochMilli())
            return date
        } else {
            val sdf = SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ssZ",
getCurrentLocale(context))
            val date = sdf.parse(dateString)
            return  date
        }


    }

    private fun getCurrentLocale(context: Context): Locale {
        return if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
            context.resources.configuration.locales.get(0) ?: Locale.getDefault()
        } else {
            context.resources.configuration.locale ?: Locale.getDefault()
        }
    }

    fun convertMillisToTime(timeInMillis: Long): String {
        val minutes = ((timeInMillis / 1000) / 60)
        val seconds = ((timeInMillis / 1000) % 60)
        return String.format("%02d:%02d", minutes, seconds)
    }

    fun getHourFromTimeStamp(timestamp: Long): String {
```

```
        var date: String = ""
        val dateFormat = DATE_FORMAT_HOUR
        try {
            date = DateFormat.format(dateFormat, timestamp).toString()
        } catch (e: Exception) {
            e.printStackTrace()
        }

        return date
    }

}
package `in`.parknav.utils

import android.util.Log

object UtilRegex {
    fun getInputBoxes(inputFormat: String, input: String): List<String> {
        var regex = Regex("([^\\s|\\-|/]{1,}|[\\s|\\-|/]{1,})")
        var regexSymbol = Regex("[\\s|\\-|/]{1,}")

        var matched = regex.findAll(inputFormat);
        var formatList = matched.map { matchResult -> matchResult.value }.toList()
        var inputList = regex.findAll(input).map { matchResult -> matchResult.value
}.toList();
//        Log.d("UtilRegex:", "getInputBoxes: $formatList $inputList")
```

```
        var result = mutableListOf<String>()

        var formatIndex = 0;

        // TODO check if input length exceed
        // TODO while loop until text is formatted
        for (i in inputList.indices) {

            if (formatIndex < formatList.size) {

                var formatText = formatList[formatIndex]
                var inputTxt = inputList[i]

                var inputIsSymbol = regexSymbol.matches(inputTxt)
                var formatIsSymbol = regexSymbol.matches(formatText)
//                Log.d("UtilRegex:", "getInputBoxes 1 : $i $inputIsSymbol
$formatIsSymbol")

                if (inputIsSymbol == formatIsSymbol && inputIsSymbol) {
                    result.add(formatText)
                    formatIndex++;
//                        Log.d("UtilRegex:", "getInputBoxes 1 : $i $inputIsSymbol
$formatIsSymbol")
                } else if (inputIsSymbol == formatIsSymbol && !inputIsSymbol &&
inputTxt.length == formatText.length) {
                    result.add(inputTxt)
                    formatIndex++;
```

```
} else if (inputIsSymbol) {
   formatIndex++;
} else {
   var expText = inputTxt;


   while (formatIndex < formatList.size && expText.isNotEmpty()) {
      formatText = formatList[formatIndex]
      formatIsSymbol = regexSymbol.matches(formatText)

      if (formatIsSymbol) {
         result.add(formatText)



      } else {

         if (expText.length > formatText.length) {
            result.add(expText.substring(0, formatText.length))
            expText = expText.substring(formatText.length, expText.length)
         } else {
            result.add(expText)
            expText = ""
         }

      }
      formatIndex++
```

```
            }


        }
    }

    return result;
  }
}
package `in`.parknav.utils

import `in`.parknav.R
import android.app.Activity
import android.content.res.Configuration
import android.graphics.Color
import android.os.Build
import android.util.TypedValue
import android.view.View
import android.view.Window
import
android.view.WindowInsetsController.APPEARANCE_LIGHT_STATUS_BARS
import androidx.annotation.ColorInt
import androidx.annotation.RequiresApi
import androidx.core.content.ContextCompat

object UtilTheme {
```

```
    @JvmStatic fun setDefaultStatusBar(view: View, activity: Activity, @ColorInt
color: Int) {
        val window: Window = activity.getWindow()



        var mode = activity.getResources().getConfiguration().uiMode and
Configuration.UI_MODE_NIGHT_MASK

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            val wic = view.windowInsetsController

           if(mode == Configuration.UI_MODE_NIGHT_YES){
              wic?.setSystemBarsAppearance(
                 0,
                 APPEARANCE_LIGHT_STATUS_BARS
              )
           }else{
              wic?.setSystemBarsAppearance(
                 APPEARANCE_LIGHT_STATUS_BARS,
                 APPEARANCE_LIGHT_STATUS_BARS
              )
           }
//            wic?.setSystemBarsAppearance(
//               APPEARANCE_LIGHT_STATUS_BARS,
//               APPEARANCE_LIGHT_STATUS_BARS
//            )
            window.setStatusBarColor(color)
```

```
            return
        } else if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            var flags = view.systemUiVisibility

            if(mode == Configuration.UI_MODE_NIGHT_NO){
                flags  = flags or View.SYSTEM_UI_FLAG_LIGHT_STATUS_BAR

            }
            view.systemUiVisibility = flags
            window.setStatusBarColor(color)
            return
        }


/*        val color = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R
&& a.isColorType) {
            // windowBackground is a color
            a.data
        } else {
            // windowBackground is not a color, probably a drawable
            ContextCompat.getColor(activity.applicationContext, R.color.white)
        }*/

// set any light background color to the status bar. e.g. - white or light blue

// set any light background color to the status bar. e.g. - white or light blue
```

```
    }

    @JvmStatic
    fun setDarkStatusBar(view: View, activity: Activity, @ColorInt color: Int) {
        val mode =
            activity.getResources().getConfiguration().uiMode and
Configuration.UI_MODE_NIGHT_MASK

        val window: Window = activity.getWindow()

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {

            val wic = view.windowInsetsController
//            wic?.setSystemBarsAppearance(
//                0,
//                APPEARANCE_LIGHT_STATUS_BARS
//            )
            if (mode == Configuration.UI_MODE_NIGHT_YES) {

                wic?.setSystemBarsAppearance(
                    APPEARANCE_LIGHT_STATUS_BARS,
                    APPEARANCE_LIGHT_STATUS_BARS
                )
            } else {
                wic?.setSystemBarsAppearance(
                    0,
                    APPEARANCE_LIGHT_STATUS_BARS
```

```
            )
        }

//          wic?.setSystemBarsAppearance(
//              APPEARANCE_LIGHT_STATUS_BARS,
//              APPEARANCE_LIGHT_STATUS_BARS
//          )
        window.setStatusBarColor(color)
        return
    } else if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        var flags = view.systemUiVisibility


        if (mode == Configuration.UI_MODE_NIGHT_YES) {

            flags = flags or (View.SYSTEM_UI_FLAG_LIGHT_STATUS_BAR)
        } else {

            flags = flags.minus(View.SYSTEM_UI_FLAG_LIGHT_STATUS_BAR)
        }
        view.systemUiVisibility = flags
        window.setStatusBarColor(color)
    }

    return
}
```

```
    @JvmStatic fun setAccentStatusBar(view: View, activity: Activity) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            var flags = view.systemUiVisibility
//            flags = flags or View.SYSTEM_UI_FLAG_
            view.systemUiVisibility = flags
            activity.window.statusBarColor = ContextCompat.getColor(
                activity.applicationContext,
                R.color.colorAccent
            )
        }
    }

    @JvmStatic fun setStatusBar(view: View, activity: Activity, color: Int, isDefault:
Boolean) {
//        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
//            var flags = view.systemUiVisibility
//            flags = flags or View.SYSTEM_UI_FLAG_
            if(isDefault){

                setDefaultStatusBar(view, activity, color)
            }else{

                activity.window.statusBarColor = color
            }
```