

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ
ОБЛІКУ МІСЦЬ ПАРКУВАННЯ АВТОМОБІЛІВ
DEVELOPMENT OF SOFTWARE FOR CAR
PARKING SPACES MANAGEMENT

Виконав(ла): студент(ка) 4 курсу, групи КНТ-122
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ВАСИЛЬСВ М.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ПОЛЯКОВ М.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОРОТКИЙ О.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. СУББОТІН
« » 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ВАСИЛЬЄВА Максима Віталійовича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка програмного забезпечення для обліку місць паркування автомобілів. Development of Software for Car Parking Spaces Management

керівник проєкту (роботи) д.т.н., професор Поляков Михайло Олексійович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «07» квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 06 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ПОЛЯКОВ М.О., професор		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання «07» квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Максим ВАСИЛЬЄВ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Михайло ПОЛЯКОВ
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
89 с., 28 рис., 5 табл., 4 дод., 19 джерел.

ПАРКУВАННЯ, АВТОМОБІЛЬ, TYPESCRIPT, REACT NATIVE,
ЗАСТОСУНОК, МОВА ПРОГРАМУВАННЯ, ПРОГРАМНЕ
ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для автоматизації обліку місць паркування автомобілів.

Предмет дослідження – програмні засоби для автоматизації обліку місць паркування автомобілів.

Мета роботи – збільшення швидкості обробки даних про зайнятість паркувальних майданчиків та зменшення часу пошуку вільного місця за рахунок автоматизації обліку.

Матеріали, методи та технічні засоби: мова програмування TypeScript, фреймворк React Native.

Результати. Розроблено програмне забезпечення для автоматизації обліку місць паркування автомобілів за допомогою мови програмування TypeScript та фреймворку React Native.

Практична цінність роботи полягає у програмній реалізації застосунку для автоматизації обліку місць паркування автомобілів.

Висновки. Реалізовано застосунок для обліку місць паркування.

Висновки. Виконано проєктування програмного забезпечення для автоматизації обліку місць паркування. Розроблено програмне забезпечення для автоматизації обліку місць паркування. Здійснено тестування розробленого програмного забезпечення для автоматизації обліку місць паркування.

Галузь використання – паркування автомобілів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 89 pages, 28 figures, 5 tables, 4 appendices, 19 sources.

PARKING, CAR, TYPESCRIPT, REACT NATIVE, APPLICATION, PROGRAMMING LANGUAGE, SOFTWARE.

The object of the study is the process of developing software for automating the accounting of car parking spaces.

The subject of the study is software for automating the accounting of car parking spaces.

The purpose of the work is to study and develop software for automating the accounting of car parking spaces.

Materials, methods and technical means: TypeScript programming language, React Native framework.

Results. Software for automating the accounting of car parking spaces has been developed using the TypeScript programming language and the React Native framework.

The practical value of the work lies in the software implementation of the application for automating the accounting of car parking spaces.

Conclusions. An application for parking space management has been implemented.

Conclusions. The software for automating parking space management was designed. Software for automating parking space management was developed. Testing of the developed software was carried out. Field of use – car parking.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	12
1.1 Поняття паркувального майданчика.....	12
1.2 Способи оплати за користування паркувальним майданчиком	19
1.3 Поняття систем для управління паркувальним майданчиком	21
1.4 Порівняння програмного забезпечення для управління паркуванням автомобілів	24
1.5 Висновки за розділом 1	35
2 Матеріали і методи	36
2.1 Вибір мови програмування	36
2.2 Вибір фреймворку для створення програмного забезпечення для автоматизації обліку місць паркування автомобілів	40
2.3 Висновки за розділом 2	44
3 Опис програми	46
3.1 Структура програмного забезпечення.....	46
3.2 Основний алгоритм роботи програмного забезпечення	48
3.3 Проєктування інтерфейсу	51
3.4 Висновки за розділом 3	53
4 Експлуатація, тестування та експериментальне дослідження програми.....	54
4.1 Призначення й умови застосування програми.....	54
4.2 Умови виконання програми	55
4.3 Виконання програмного забезпечення.....	55
4.4 Тестування програмного забезпечення	60
4.5 Висновки за розділом 4	62
Висновки.....	63
Перелік джерел посилання	65

Додаток А Технічне завдання.....	67
Додаток Б Опис програми	70
Додаток В Текст програми	75
Додаток Г Слайди презентації.....	83

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

TS – TypeScript;

БД – база даних;

ОС – операційна система;

ПЗ – програмне забезпечення.

ВСТУП

В сучасних реаліях, коли інформаційні технології розвиваються шаленими темпами, а міське населення постійно зростає, питання належного керування транспортною інфраструктурою набуває надзвичайної ваги. Збільшення кількості автотранспорту у містах призводить до гострої нестачі місць для паркування, ускладнень у транспортному русі та зниження рівня комфорту для водіїв. Застарілі методи обліку та контролю використання паркувальних зон, які переважно ґрунтуються на ручній обробці, або напівавтоматизованих підходах, демонструють низьку ефективність, вимагають значних зусиль і схильні до помилок. З огляду на це, виникає нагальна потреба у створенні передових програмних рішень, здатних забезпечити автоматизацію процесів обліку, моніторингу та управління паркувальними територіями.

Актуальність цієї кваліфікаційної роботи зумовлена необхідністю впровадження інноваційних інформаційних систем, що дозволяють оптимізувати використання наявних площ для паркування, покращити якість обслуговування споживачів послуг та забезпечити миттєвий доступ до актуальних відомостей про вільні місця. Розробка спеціалізованого програмного забезпечення для обліку паркомісць сприятиме підвищенню загальної ефективності функціонування міської інфраструктури, мінімізації часових витрат на пошук місця для авто та зменшенню загального транспортного навантаження.

Об'єктом дослідження, є процес створення програмного забезпечення, спрямованого на автоматизацію обліку паркувальних місць для автомобілів. У рамках аналізу будуть розглянуті ключові етапи формування програмного продукту, включаючи збір вимог, проєктування структури, безпосередню реалізацію та тестування системи.

Предметом дослідження виступають програмні інструменти та засоби, які застосовуються для автоматизації процесу обліку паркування. Особливу

увагу приділено сучасним розробкам у сфері мобільних застосунків, що забезпечують інтуїтивно зрозумілий інтерфейс для користувача та гарантують високий рівень продуктивності системи.

Метою цієї роботи є проведення дослідження та подальша розробка програмного забезпечення для автоматизації обліку місць паркування. Для досягнення окресленої мети передбачається виконання комплексу завдань, зокрема: аналіз відповідної предметної сфери, визначення функціональних вимог до системи, обґрунтований вибір технологічних засобів для розробки, проєктування архітектури програмного рішення, реалізація програмного коду та проведення його всебічного тестування.

У процесі виконання роботи використовуються сучасні матеріали, методи та технічні засоби, серед яких варто відзначити мову програмування TypeScript та фреймворк React Native. Застосування цих технологій дає змогу створити кросплатформений мобільний додаток, що здатний коректно працювати, зберігаючи при цьому високий показник продуктивності та зручності у використанні.

Наукова новизна даної роботи полягає у запровадженні комплексного підходу до розробки програмного забезпечення для автоматизації обліку паркувальних місць, який поєднує передові методи архітектурного проєктування із сучасним застосуванням мобільних платформ. Запропоноване рішення враховує специфіку функціонування паркувальних комплексів та реальні потреби кінцевих користувачів.

Практична цінність отриманих результатів полягає у створенні програмного застосунку, який може знайти своє застосування для автоматизації обліку паркувальних місць у різних контекстах — від невеликих приватних паркувальних зон до великих міських паркувальних об'єктів. Впровадження розробленого ПЗ дозволяє підвищити ефективність керування парковками, скоротити операційні витрати на їх обслуговування та покращити загальну якість сервісу.

У результаті виконання роботи розроблено програмне забезпечення для автоматизації обліку паркувальних місць, використовуючи мову програмування TypeScript та фреймворк React Native. Розроблений мобільний додаток надає можливість вести точний облік зайнятих і вільних паркомісць, спрощує взаємодію між користувачами та адміністраторами системи, а також значно підвищує зручність користування послугами паркування.

У рамках дослідження було пройдено повний цикл розробки програмного продукту: починаючи з етапу аналізу вихідних вимог і до кінцевого тестування готового рішення. Проведені випробування підтвердили працездатність та ефективність розробленого програмного забезпечення, а також його повну відповідність усім встановленим вимогам.

Сфера застосування результатів цієї роботи охоплює напрямки організації та управління паркувальними процесами. Створений застосунок цілком придатний для впровадження у діяльність комерційних організацій, що надають послуги паркування, а також у муніципальні системи управління транспортною інфраструктурою міста.

Таким чином, кваліфікаційна робота сфокусована на вирішенні актуального практичного завдання — автоматизації обліку паркомісць шляхом розробки сучасного програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття паркувального майданчика

Парковка, або автостоянка — це впорядкована ділянка, призначена для тимчасового залишення транспортних засобів. Зазвичай цей термін стосується зони, виключно відведеної для паркування, з твердим або напівтвердим покриттям. У більшості регіонів, де автотранспорт є основним способом пересування, паркувальні майданчики є невід’ємною частиною міських та приміських ландшафтів. Торгові центри, арени для спортивних змагань та подібні об’єкти часто мають величезні площі, відведені під паркування.

Паркувальні зони, як правило, є джерелом забруднення водних ресурсів через значну площу водонепроникних покриттів, а також через те, що в більшості з них або зовсім відсутні, або є мінімальні системи контролю стічних вод. Багато районів сьогодні також вимагають певного рівня озеленення на парковках, щоб створити тінь і допомогти зменшити вплив асфальтованих поверхонь на формування ефектів «острова тепла». Численні місцеві органи влади регламентують мінімальну кількість паркомісць для таких об’єктів, як магазини (залежно від площі) та житлові комплекси (за кількістю спалень). Органи влади вимагають виділення певної частки місць для осіб з інвалідністю, які мають відповідний дозвіл на паркування. Сучасні паркувальні системи використовують різноманітні технології, щоб дозволити водіям сплачувати за паркування, допомагати їм знаходити вільні місця та отримувати свої автівки, а також покращувати загальний досвід паркування.

Дослідження паркувальних площ у США, проведене у 2015 році, показало, що середня вартість будівництва, без урахування вартості придбання землі, становила приблизно 24 000 доларів США за місце на наземній стоянці та 34 000 доларів США за місце на підземній парковці. Дональд Шуп, професор містобудування в Каліфорнійському університеті в Лос-Анджелесі, у 2023 році оцінив, що загальна кількість місць для паркування у Сполучених

Штатах коливається між 700 мільйонами та 2 мільярдами, що дорівнює від 2,5 до 7,0 місць на один автомобіль [1].

Вплив масового паркування в межах міст давно є предметом дискусій. Заміна історичних будівель гаражами та парковками спричинила виникнення рухів за збереження історичної спадщини у багатьох містах. Величезні території, відведені під паркування, широко вважаються тими, що руйнують пішохідну структуру міста, оскільки максимізують зручність для кожної окремої споруди, але водночас ускладнюють пересування піших між ними [2]. Великі асфальтовані ділянки іноді називають «паркувальними кратерами», «паркувальними пустелями» та подібними термінами, що підкреслює їхню «безлюдну» натуру та перепони, які вони створюють для пішохідного руху [3].

Політики міського планування, такі як норми мінімального та максимального забезпечення паркувальними місцями, можуть впливати на габарити приватних паркувальних зон.

З огляду на тенденцію до створення життєздатніших та зручніших для пішоходів районів, мінімальні норми паркування (політика, яка вимагає наявності певної мінімальної кількості місць для паркування біля кожної будівлі) зазнали критики як від прихильників вулиць, зручних для життя, так і від забудовників. Протягом певного періоду уряд Великої Британії радив місцевим радам встановлювати максимальні норми паркування, щоб стримувати використання автомобілів. Американські міста, такі як Вашингтон, округ Колумбія, наразі розглядають можливість скасування мінімальних обмежень на паркування, як спосіб збільшення житлового простору для мешканців, одночасно стимулюючи користування громадським транспортом. Паркувальні зони, спеціально обладнані для велосипедів, також стають дедалі поширенішими як відповідь на зростаючу екологічну свідомість та турботу про здоров'я. Вони можуть включати стійки та замки для велосипедів, а також сучасніші інженерні рішення для забезпечення безпеки та зручності. Наприклад, у Токіо зростає кількість велосипедних парковок, які використовують автоматизовані системи паркування.

Зусилля, спрямовані на зменшення площ, відведених під парковки, з метою зниження залежності від автомобілів, були реалізовані у Пекіні, Мехіко, Делі та низці міст Каліфорнії. Портленд, Міннеаполіс, Остін скасували обов'язкову вимогу щодо мінімальної кількості паркувальних місць. Станом на 2 листопада 2023 року Остін (Техас) є найбільшим містом у США, яке вчинило такий крок, – це має на меті заохотити пішохідний рух, їзду на велосипеді, використання громадського транспорту, зменшити вартість житла та збільшити кількість житлових одиниць, які можна збудувати на території міста [4].

У Швеції та Данії законодавчо визначено два типи стоянки для авто: на вуличних шляхах та дорогах загального користування, або на приватних ділянках. Недотримання правил паркування на громадських дорогах вважається правопорушенням, що тягне за собою штрафні санкції. Недотримання правил паркування на приватній території (навіть якщо вона належить муніципалітету) класифікується як порушення договірних умов і карається додатковим збором за паркування (у шведській мові: *kontrollavgift* – плата за перевірку). Для власника транспортного засобу різниця є несуттєвою, і відповідальність завжди лежить на ньому [4].

На території Сполученого Королівства існують дві категорії місць для паркування автомобілів: на державній або на приватній власності. Поліція здійснюватиме розслідування будь-якого зареєстрованого інциденту на державній території, але не має юридичного зобов'язання робити це, якщо подія сталася на приватній землі. Дорога загального користування визначається Законом про дорожній рух 1972 року та Положенням про Закон про дорожній рух 1972 року (із змінами) 1988 року (SI 1988/1036) як: «Дорога» у контексті Англії та Уельсу означає будь-яке шосе та будь-яку іншу дорогу, до якої має доступ громадськість, включаючи мости, через які проходять ці дороги. Існує також відповідне рішення Палати лордів щодо цього питання [4].

Співробітники цивільної служби контролю стежать за дотриманням правил паркування на громадських автостоянках, що належать місцевим органам влади. Це включає випадки, коли не було придбано квиток для оплати (якщо такий потрібен)/паркування поза межами розмітки/та інші порушення.

У США міністерства транспорту кожного штату встановлюють відповідні пропорції паркувальних місць для осіб з інвалідністю на приватних об'єктах та у громадських зонах паркування. За певних умов може знадобитися збільшення кількості спеціально виділених місць. Ці зарезервовані місця регулюються відповідно до Керівних принципів доступності Закону про американців з інвалідністю [4].

Можна застосувати шлагбауми, аби унеможливити паркування без здійснення оплати.

Сучасні паркінги впроваджують різноманітні технології, щоб допомогти водіям знаходити вільні місця через системи навігації та інформації про паркування, полегшити маневрування транспортних засобів та покращити загальний досвід паркування. Це включає адаптивне освітлення, датчики, системи внутрішнього позиціонування (IPS) та опції мобільної оплати. Наприклад, у торговому центрі Santa Monica Place у Каліфорнії на кожному паркувальному модулі встановлені камери для підрахунку зайнятих місць та допомоги у пошуку залишених авто [5].

На відкритих парковках GPS може бути використаний для запам'ятовування розташування транспортного засобу (деякі програми автоматично зберігають місцезнаходження при вимкненні двигуна, коли смартфон втрачає з'єднання з Bluetooth автомобіля) [5].

Щодо закритих паркінгів, один із методів – фіксація власної Wi-Fi сигнатури (рівень сигналу від кількох виявлених точок доступу) для запам'ятовування розташування авто. Іншою опцією є використання додатків для смартфонів, які застосовують інерційні обчислення, фіксують повороти, здійснені автомобілем у приміщенні, зіставляють час проходження між поворотами та використовують алгоритми машинного навчання для

визначення приблизної позиції припаркованого авто на основі плану чи схеми поверху [6].

Існують постачальники послуг онлайн-бронювання, щоб допомогти водіям автоматично знаходити місця для довготривалого паркування, а також забезпечити суттєву економію тим, хто бронює заздалегідь. Вони оперують технологіями управління запасами в реальному часі, відображаючи доступні місця, відсортовані за вартістю та відстанню від аеропорту.

Також, існують мобільні застосунки, які пропонують послуги бронювання місць для тривалого паркування, подібні до онлайн-сервісів або агрегаторів бронювання. Деякі мобільні програми для довгострокового паркування також містять покрокові карти для пошуку паркомісця, як-от ParkJockey, що працює у США та Великій Британії.

Сонячні навіси для парковок — це сонячні панелі, змонтовані на навісних конструкціях над паркувальними місцями. Їхнє встановлення обходиться до двічі дорожче, ніж звичайні відкриті сонячні панелі, через додаткові матеріали, необхідні для підняття конструкції, щоб під нею могли стояти автомобілі. Вони також можуть слугувати захистом для авто від суворих погодних умов та надмірного тепла [6].

Паркувальні зони, як правило, є джерелом забруднення води через значні площі, покриті водонепроникними матеріалами. Майже вся дощова вода (за винятком випаруваної), що випадає на таку поверхню, перетворюється на поверхневий стік. Для запобігання затопленням та небезпечній дорожній обстановці, ці майданчики проектуються з метою відведення та збору цього стоку. Разом із дорогами, парковки часто є головним каталізатором забруднення води у міських ландшафтах.

Автомобілі постійно виділяють забруднюючі речовини; до найбільш істотних належать бензин, машинне мастило та важкі метали. Вони утворюються як у побічних продуктах згоряння пального, так і містяться у герметиках на основі асфальту та кам'яновугільної смоли, що

використовуються для обслуговування покриття. Чимало паркінгів також є значним джерелом побутового сміття, яке потім потрапляє до водойм.

Традиційно стічні води з парковок спрямовувалися безпосередньо у зливову каналізацію, струмки, дренажні канали чи навіть у систему санітарної каналізації. Проте більшість великих міських агломерацій нині вимагають встановлення спеціальних споруд для контролю зливових вод на нових об'єктах. До типових рішень належать утримуючі резервуари, інфільтраційні ставки та фільтраційні траншеї. Деякі новітні розробки включають системи біоутримання, які активно залучають рослинність для поглинання та фільтрації забруднюючих речовин. З іншого боку, велика кількість вже існуючих об'єктів майже не мають, або мають дуже обмежені можливості для контролю стоку [7].

Сучасним варіантом є впровадження проникних мощених поверхонь, таких як спеціальна цегла, пористий бетон, камінь, тротуарна плитка із зазорами або спеціальні килимки, виготовлені з протекторів шин. Ці матеріали дають змогу дощовій воді просочуватися у ґрунт через природні щілини в покритті. Забруднення може осідати на поверхню ґрунту на цій обмеженій ділянці, але цей ґрунт, як правило, ефективно фільтрує воду перед її подальшим просочуванням. Однак це може спричинити проблеми, якщо забруднювачі потрапляють у ґрунтові води, особливо у районах, де "нижче за течією" здійснюється забір води для питного постачання [7].

Сьогодні багато регуляторних норм також вимагають мінімального озеленення на паркувальних майданчиках. Зазвичай це зводиться до висадки дерев для створення тіні. Хоча клієнти давно надають перевагу затіненим місцям у спеку, власники паркувальних сервісів тривалий час скептично ставилися до посадки дерев через додаткові витрати на прибирання території.

Мощені поверхні сприяють формуванню ефектів "міського теплового острова" двома основними шляхами. По-перше, це надмірне накопичення тепла. Темні покриття та вузькі простори, утворені міськими будівлями, затримують більше сонячної енергії. Відбивна здатність (альбедо) покриття

порівняно з природними поверхнями є критичною, оскільки вище альbedo означає нижчі температури. Асфальт, найтемніший, має коефіцієнт сонячного відбиття лише 5-10 відсотків. Світліші покриття можуть мати показник 25 відсотків і більше. Для ґрунтів та різних видів рослинності значення альbedo варіюється від 5 до 45 відсотків. Другою причиною теплових островів є низька здатність дорожніх та будівельних матеріалів утримувати вологу. Ці матеріали є водонепроникними, тому вода недоступна для розсіювання сонячного тепла через процес випаровування [8].

Дослідження продемонстрували, що висаджування дерев суттєво знижує температуру на відкритих, асфальтованих ділянках. В одному дослідженні в Алабамі денна літня температура у центрі голої парковки сягала 49 °С, тоді як у зоні з невеликим острівцем дерев температура не перевищувала 32 °С. Також було встановлено, що кожне додатково висаджене дерево з повноцінною кроною може сприяти додатковому зниженню температури на 1 °F.

Нещодавно парковки почали розглядати як найбільш підходящу нерухомість для монтажу великих сонячних електростанцій, що дає додаткову перевагу у вигляді затінку для автомобілів, припаркованих під ними.

Для забезпечення паркувальних місць потрібна значна площа — приблизно 25 квадратних метрів, або 270 квадратних футів, на одне паркомісце. Це означає, що паркувальним зонам часто потрібна більша земельна ділянка, ніж співвідносним офісним чи торговельним будівлям, якщо більшість відвідувачів і працівників прибувають власним транспортом. Це призводить до асфальтування великих територій [8].

Деякі паркувальні комплекси обладнуються зарядними станціями для електромобілів. У регіонах з особливо суворими зимами на більшості паркомісць передбачено доступ до електроенергії для підігрівачів двигунів, оскільки антифризу може бути недостатньо для запобігання застиганню.

Через збільшення кількості автомобілів та сприяння утворенню міських теплових островів завдяки матеріалам, використаним у їхньому будівництві, парковки є відповідальними за значні викиди парникових газів.

1.2 Способи оплати за користування паркувальним майданчиком

Для збору коштів із водіїв за користування паркувальним майданчиком застосовують різноманітні технологічні рішення.

На багатьох парковках облаштовані шлагбауми. Клієнт під'їжджає до паркомату на в'їзді, натискає кнопку для отримання талона, забирає його (шлагбаум тоді підіймається), і в'їжджає на територію. Для виїзду необхідно пред'явити талон касиру у виїзній будці та внести належну суму, після чого касир відчиняє шлагбаум.

У 1954 році з'явилися перші автоматизовані паркінги, де водій, маючи щомісячну абонементну картку-ключ, міг в'їжджати та виїжджати з паркінгу, керуючи підйомом/опусканням стріли.

У новітніх системах використовуються автоматичні термінали оплати. Водій пред'являє талон та сплачує необхідну суму ще до повернення до авто. Потім він під'їжджає до виїзного терміналу та знову ж таки пред'являє талон. Якщо оплату не здійснено, шлагбаум не відчиниться. Це змушує клієнта або скористатися інтеркомом для зв'язку зі співробітником, або здати назад до пункту оплати чи каси [9].

На деяких великих парковках при аеропортах у Сполучених Штатах водій може скористатися кредитною картою у вхідному терміналі замість отримання паперового талона. Коли водій використовує ту саму картку на виїзді з парковки, відповідна плата за паркування автоматично розраховується та списується з картки.

На певних парковках водії віддають свої талони касирам, які можуть знаходитись у окремій касі або стійці (часто розташованих подалі від в'їздів/виїздів). Такі пункти називають «шроф-офісами» або просто «шроф» у

певних місцях Гонконгу та інших регіонах Східної Азії, під впливом гонконгських традицій. Якщо талон не оплачено, шлагбаум залишається закритим. Останніми роками касирів та персонал «шроф» часто замінюють автоматами. Інший спосіб оплати – водій розраховується з працівником на в'їзді на паркінг, а виїзд обладнаний смугою з односпрямованими шипами, що пропускають машини лише на виїзд [9].

Також можуть використовуватися паркувальні автомати, де водії платять наперед за час, необхідний для паркування на вибраному місці. Компанія Pango (гра слів «плати та рушай»), заснована в Ізраїлі у 2007 році, розробила мобільний застосунок, що дозволяє користувачам знаходити та оплачувати доступні паркомати. Враховуючи облік; додаток також можна використовувати для оплати паркінгу в гаражах. Облікові записи користувачів прив'язані до способу оплати, система запам'ятовує місцезнаходження авто та дає змогу ділитися інформацією про паркування з друзями у Facebook. За невелику щомісячну плату за зареєстроване авто користувачі можуть підписатися на текстові нагадування незадовго до закінчення часу паркування, а у деяких містах можна продовжити час паркування через мобільний. Філадельфія стимулює ротацію паркувальних місць, застосовуючи прогресивну шкалу плати при подовженні часу паркування. Інший додаток, Streetline, головна функція якого – допомагати водіям знаходити вільні місця за допомогою смартфонів, має таймер для повернення до паркомата до закінчення сесії та фільтр для вибору між вуличними та позавуличними місцями; він також інтегрується з камерою телефону, щоб користувач міг сфотографувати своє авто [10].

На інших паркінгах діє система «сплати та демонструй», де талон купується в автоматі та розміщується на панелі приладів. Інспектори з контролю паркування патрулюють територію для забезпечення дотримання правил.

Схожою є система, де оплата за паркування здійснюється мобільним телефоном через SMS-повідомлення, що містить номерний знак. У цьому

випадку віртуальний касир реєструє авто та час відправлення повідомлення, а згодом, коли потрібно, надсилається нове SMS. Фактична оплата відбувається через рахунок за мобільний зв'язок [10].

Починаючи з 1978 року, у Великій Британії стало можливим попереднє бронювання паркомісць через спеціалізовані фірми, наприклад, BCP. Це поширено на всіх аеропортах, великих портах та у містах.

1.3 Поняття систем для управління паркувальним майданчиком

Складові системи для управління паркуванням:

- керування паркувальними місцями — це процес налагодження роботи парковок, щоб гарантувати, що кожен користувач об'єкта може знайти місце для свого автомобіля;
- програмне забезпечення для управління паркуванням — це, як правило, веб-застосунок чи мобільна програма, що використовується для впорядкування, спостереження та контролю паркувальних зон. Воно автоматизує розподіл місць, облік часу паркування, контроль дотримання умов бронювання та інші завдання, пов'язані з адмініструванням паркінгу [11];
- система керування паркуванням охоплює різноманітні методи, які застосовуються для обслуговування паркувальної ділянки. Вони варіюються від простих аркушів паперу та паперових дозволів до сучасних рішень із підтримкою Bluetooth. Кожному методу притаманні свої особливості функціонування, переваги та недоліки;
- "розумні" системи керування паркуванням інтегрують передові технології, зокрема Інтернет речей (IoT), сенсори, електронні табло, RFID, камери та NFC. Софт для управління паркінгом взаємодіє з цими "розумними" комплексами [11].

Системи управління паркуванням охоплюють широкий спектр робочих принципів, кожен з яких впливає на функціональність програмного забезпечення. Оскільки софт для управління паркуванням створюється для

роботи в межах цих систем, перед порівнянням різних виробників важливо зрозуміти існуючі типи програмного забезпечення та шляхи їхньої інтеграції з конкретними апаратними рішеннями [11].

Програмні рішення для адміністрування паркінгу можна класифікувати залежно від принципу їхньої роботи — того підходу, який використовується для спрощення управління паркувальним простором. Ось перелік поширених типів, а також їхні плюси та мінуси:

- системи, засновані на квитках, спроможні відслідковувати час перебування та суму до сплати, видаючи водіям квитки зі штрих-кодами чи QR-кодами. Вони ідеально підходять для комерційних парковок, оскільки сприяють збору оплати. Існують варіації квиткових систем, що використовують мобільні додатки для управління процесом. Проте вони все одно потребують регулярного обслуговування бар'єрів і можуть давати збій у години пікового навантаження;

- системи на базі RFID зчитують унікальні RFID-мітки або наклейки, вмонтовані на лобове скло, щоб дозволити або заборонити в'їзд автотранспорту на паркінги та у гаражі. Хоча вони забезпечують швидку та точну ідентифікацію дозволених авто, їхнє впровадження та постійна експлуатація тягнуть за собою значні витрати. До того ж, мітки можуть із часом виходити з ладу або губитися. Частина або весь операційний процес можна також адмініструвати через програмне забезпечення;

- системи на основі сенсорів функціонують зі спільним використанням камер та IoT-датчиків. Вони фіксують наявність транспортних засобів на паркомісцях, розпізнають номерні знаки та надають актуальну інформацію про вільні місця. Незважаючи на те, що вони дають точні дані про завантаженість паркінгу, їхнє встановлення та подальше обслуговування є фінансово обтяжливим;

- системи з підтримкою Bluetooth дають змогу водіям в'їжджати та виїжджати з паркувальної зони, використовуючи свій смартфон або BLE-пристрій як безпроводний ключ. Замість того, щоб діставати талон чи

використовувати пластикову картку, смартфон водія, оснащений Bluetooth, взаємодіє з приймачем біля виїзду, надаючи доступ. Це найбільш зручний варіант, що не вимагає фізичних дій. Водіям у багатьох випадках не потрібно опускати скло, шукати талон чи натискати кнопки [11].

Порівняння видів систем для паркування представлено в таблиці 1.1.

Таблиця 1.1 – Порівняння видів систем для паркування

Функція	Системи паркування Bluetooth	Системи паркування на основі бар'єрів/тікерів	Системи паркування LPR	Системи паркування RFID	Розумні системи паркування на базі Інтернету речей
Вартість	Помірна початкова вартість	Вища початкова вартість	Висока початкова вартість	Висока початкова вартість	Висока початкова вартість
Легкість впровадження	Помірна	Помірна	Комплекс	Помірна	Комплекс
Безпека	Помірна	Від середнього до високого	Висока	Висока	Висока
Ефективність	Помірна	Від середнього до високого	Висока	Висока	Висока
Масштабованість	Помірна	Помірна	Висока	Помірна	Висока
Технічне обслуговування	Низьке	Помірне(апаратне та програмне забезпечення)	Помірне (апаратне та програмне забезпечення)	Помірне (апаратне забезпечення та теги)	Від середнього до високого
Складність технологій	Від низького до помірного	Помірна	Висока	Помірна	Висока
Зручність для користувача	Помірна	Помірна	Висока	Висока	Висока

Будь-яке програмне забезпечення для керування паркувальними місцями має надавати наступні можливості:

- моніторинг доступності місць у реальному часі;
- відстеження показників завантаженості та аналіз динаміки використання;
- контроль за тим, хто має вхід до паркінгу;
- можливість точного визначення розташування паркувальних зон.

Впровадження системи управління паркуванням несе за собою низку вагомих переваг. Ось основні з них:

- зменшення навантаження на адміністрацію. Дозволяючи користувачам самостійно здійснювати бронювання та оплату, вивільняється робочий час відповідальних за об'єкт осіб для виконання інших першочергових завдань;

- оптимізація використання паркувальних площ. Завдяки інформаційним дашбордам з даними про наявність вільних місць у реальному часі, керівники об'єктів здатні ефективно контролювати та регулювати завантаженість простору;

- отримання аналітичних відомостей про використання. Платформи управління паркінгом постачають важливі дані, необхідні для обґрунтованого прийняття рішень;

- посилення рівня безпеки. Застосовуючи механізми контролю доступу, організації можуть гарантувати, що на територію гаража чи майданчика потраплятимуть виключно авторизовані транспортні засоби. [11]

1.4 Порівняння програмного забезпечення для управління паркуванням автомобілів

У сучасному світі вибір потрібного програмного забезпечення для керування паркуванням є ключовим. В роботі розглянуто декілька популярних варіантів програмного забезпечення для керування паркуванням.

Awaio – комплексне управління паркувальними місцями для гібридних робочих схем. Awaio є мобільно-орієнтованою платформою, що спрощує адміністрування всіх спільних офісних ресурсів – парковок, робочих столів, переговорних кімнат, зарядних станцій для електромобілів та інших – з єдиної точки контролю (рисунок 1.1) [11].

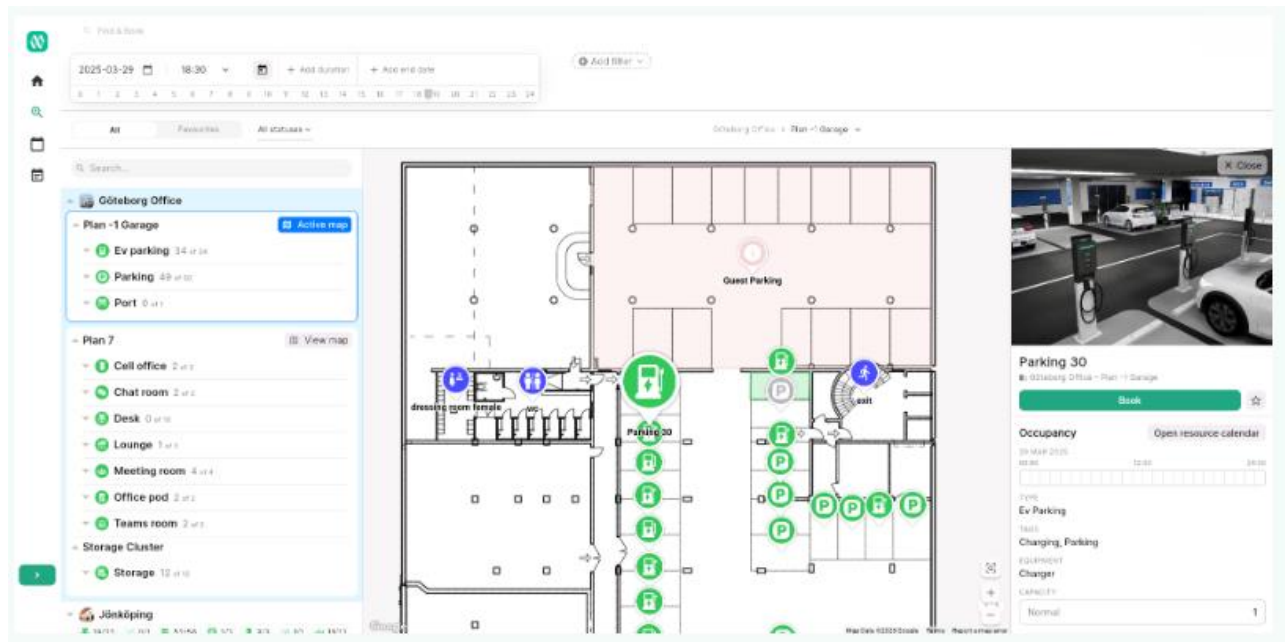


Рисунок 1.1 – Платоформа Awaio [11]

Користувачі платформи можуть легко перевірити доступність у реальному часі правувальних місць та забронювати їх наперед. [11]

У розділі "Паркування та Мобільність" платоформи Awaio користувачі мають змогу:

- миттєво бачити, які паркомісця вільні, а які зайняті;
- заздалегідь резервувати місця на певні дати й час;
- фільтрувати пошук за наявністю зарядних станцій для електромобілів, коли це потрібно;
- користуватися інтерактивною навігацією для швидкого знаходження зарезервованої позиції;
- звільняти заброньоване місце одним кліком, якщо плани змінилися, роблячи його миттєво доступним для інших.

Ключові особливості:

- динамічний розподіл для спільного використання обмежених паркувальних ресурсів;
- налаштовувані та гнучкі правила резервування;
- єдиний центр аналітики;
- динамічний розподіл для обмеженої кількості паркувальних місць.

Awaio дає компаніям змогу максимально ефективно використовувати обмежений паркінг, дозволяючи корисувачам розподіляти місця на основі фактичного закріплення, а не фіксованого [11].

Переваги:

- надає повний набір інструментів для управління всіма ресурсами: паркування, зарядні станції тощо;
- впровадження не вимагає інвестицій в додаткове обладнання чи витрат на обслуговування;
- модульна структура ціноутворення робить рішення масштабованим і достатньо гнучким для адаптації;
- підтримує необмежену кількість користувачів;
- співробітники можуть авторизуватися через свій корпоративний акаунт, уникаючи необхідності створювати окремий пароль;
- вбудований платіжний шлюз для обробки розрахунків за паркування для гостей або внутрішніх оплат [11].

Недоліки:

- поки що не реалізовано функціонал управління користувачами;
- ціноутворення [11].

Parkade – програмне забезпечення для управління паркуванням у житлових та багатоквартирних комплексах. Parkade – це програмний інструмент для керування паркуванням, орієнтований на керуючих житловою нерухомістю, що дозволяє обслуговувати багатоквартирні будинки, готелі та кондомініуми [12]. Його мобільний та веб-додатки надають засоби для бронювання місць, обробки платежів та реєстрації фактів неправильного паркування (рисунок 1.2).

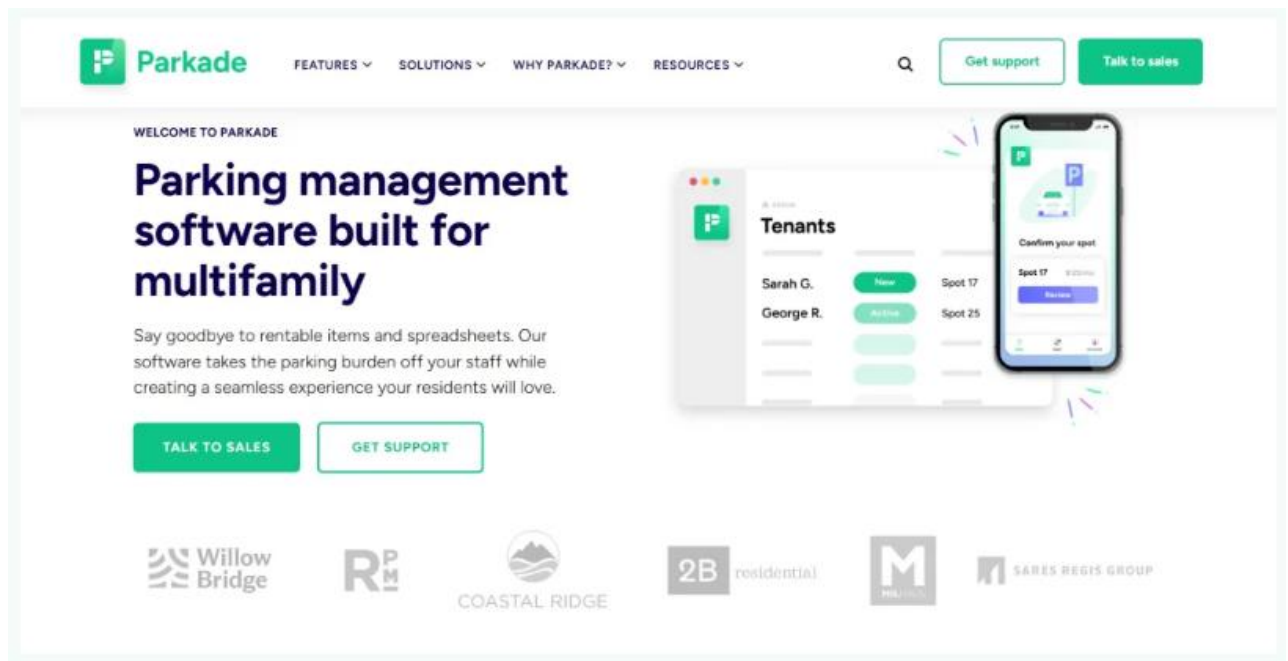


Рисунок 1.2 – Платофрма Parkade [12]

Parkade також пропонує опції керування паркуванням для гостей. Через мобільний додаток мешканці можуть самостійно зарезервувати місце для гостя, або звернутися до адміністратора для внесення гостя до системи [12].

Якщо мешканець обирає другий варіант, гість отримає унікальне посилання, яке він використовуватиме для доступу до зарезервованого паркувального місця.

Ключові особливості:

- паркінг для гостей;
- керування надходженнями;
- правила бронювання;
- інтеграція з системами управління нерухомістю (property management systems);
- мобільний додаток;
- короткострокове паркування.

Переваги:

- інтегрується з існуючими системами управління паркуванням;
- пропонує оперативну клієнтську підтримку;

- інтерфейс мобільного додатку є досить інтуїтивним.

Недоліки:

- інформація про вартість послуг не є публічно доступною.

Ronspot – безкоштовна програма для контролю паркувальних місць для невеликих офісних просторів. Ronspot надає можливість користувачам бронювати паркомісця, столи та переговорні кімнати безкоштовно протягом перших 12 місяців [12]. Хоча ця пропозиція має обмеження щодо кількості організацій, вона підтримує необмежену кількість користувачів, дозволяючи працівникам резервувати місця аж на три дні вперед (рисунок 1.3).

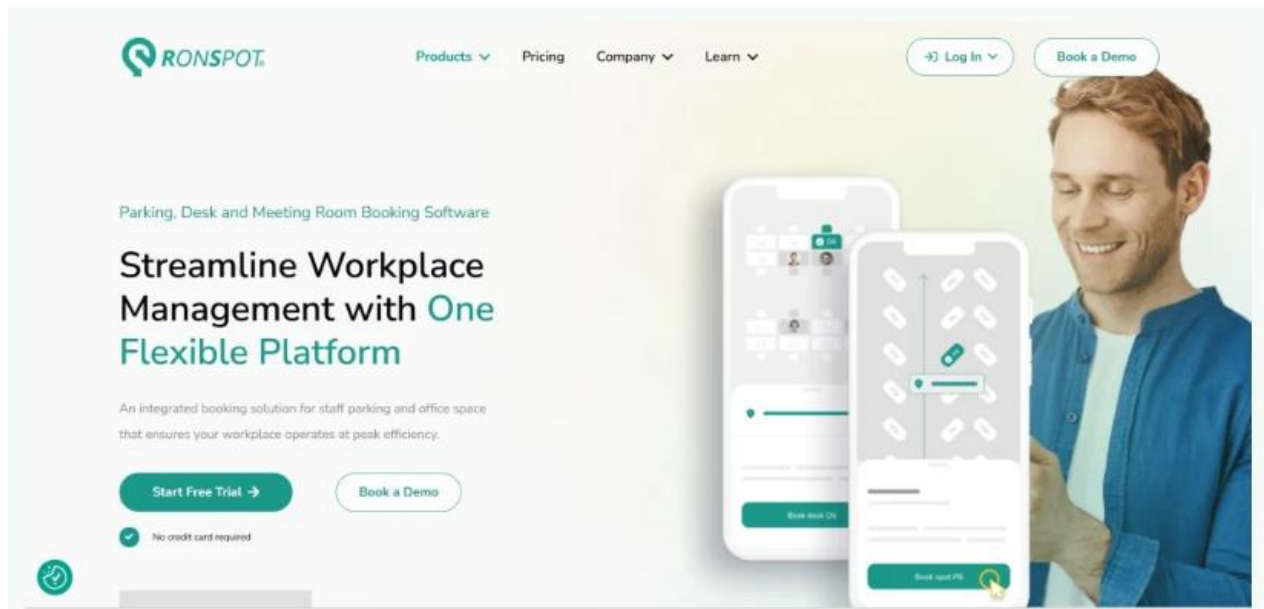


Рисунок 1.3 – Платформа Ronspot [12]

Через адмін-панель Ronspot керівники мають інструмент, щоб розподіляти кредити на бронювання серед своїх працівників. Щойно ці кредити вичерпаються, співробітникам доведеться очікувати наступного тижня, щоб знову мати можливість бронювати [12]. Така система не дає накопичувати місця та допомагає зменшити конфлікти навколо паркування.

Ronspot також обладнано інтерактивною картою, яка дає змогу працівникам перевіряти вільні місця та резервувати їх на своїх смартфонах лише за два кліки. Однак, на відміну від Awaio, карта Ronspot не відображає,

хто саме зайняв кожне місце, – це компроміс, який зменшує прозорість для адміністратора, але водночас підвищує рівень приватності для персоналу [12].

Ключові особливості:

- регуляції та умови бронювання;
- аналітичний дашборд;
- система кредитів на бронювання;
- автоматична реєстрація;
- інтерактивна схема;
- інтеграції з системами доступу.

Переваги:

- прозора система тарифікації;
- наявність безкоштовного тарифу, яким можна користуватися рік перед переходом на платний;
- наявність функцій для управління робочими зонами та кімнатами для зустрічей.

Недоліки:

- ключові можливості, такі як інтеграція з шлагбаумами, розширені аналітичні звіти, попереднє бронювання та управління відвідувачами, доступні лише у найдорожчому пакеті;
- інтерактивну карту може завантажувати лише команда Ronspot.

ParkingPass – програмне забезпечення для управління публічними паркінгами. ParkingPass ідеально підходить для керуючих нерухомими активами, які обслуговують різноманітні об'єкти, включаючи університетські містечка, муніципалітети та житлові комплекси (кондомініуми) [12]. Його найбільш цінною функцією є можливість гостьового паркування, що дозволяє мешканцям самостійно реєструвати своїх гостей без участі адміністративного персоналу (рисунок 1.4).



Рисунок 1.4 – Платоформа ParkingPass [12]

Це програмне забезпечення здатне інтегруватися з камерами для зчитування номерних знаків, гарантуючи в'їзд на парковку лише авторизованим транспортним засобам [12].

На відміну від більшості платформ для контролю громадських парковок, ParkingPass повністю працює без паперових документів. Усі дозволи, штрафи та квитки мають цифровий формат, що позиціонує його як екологічне рішення.

Ключові особливості:

- контактна мапа;
- автоматизоване оформлення дозволів;
- платіжний шлюз;
- система розпізнавання номерів;
- панель управління спільнотою;
- паркувальна звітність.

Переваги:

- цифрові дозволи на паркування;
- оперативна служба підтримки клієнтів;
- вбудована функція `lpr` (розпізнавання номерів) спрощує ідентифікацію несанкціонованих авто.

Недоліки:

- потребує спеціалізованого обладнання та має запутаний процес початкового налаштування;
- дані щодо вартості послуг не розголошуються (недоступні);
- право перегляду інтерактивної карти мають лише адміністратори.

ParkHub – програмне забезпечення для контролю паркування в університетах та на заходах. Хоча рішення ParkHub здебільшого використовується навчальними закладами, воно також обслуговує аеропорти, місця проведення подій, муніципалітети та комерційну нерухомість [12]. Його ключова перевага полягає в інтеграції з головними системами продажу квитків. Це дає можливість організаторам контролю паркування на заходах приймати оплати за місця безпосередньо з кредитних карток (рисунок 1.5).



Рисунок 1.5 – Платоформа ParkHub [12]

Ключові особливості:

- дані про завантаженість парковок у реальному часі;
- портативна точка продажу (pos);
- інтеграція з платіжними системами цифрового типу.

Переваги:

- надає відшліфовані та чітко структуровані аналітичні дані щодо паркування;

- пропонує навчальні матеріали для нових та чинних адміністраторів паркінгів;

- команда підтримки клієнтів оперативно реагує на запити.

Недоліки:

- витрати на встановлення та обслуговування апаратного забезпечення можуть суттєво збільшити бюджет;

- на відміну від інших рішень у списку, має лише функції, спрямовані на управління паркувальними зонами.

Parkable — програмне забезпечення для управління парковками комерційних будівель та офісів. За допомогою Parkable, менеджери об'єктів можуть оперативно знаходити та віддавати вільні місця орендарям і персоналу, мінімізуючи простой та втрати доходу [12]. Крім того, Parkable надає можливість співробітникам та мешканцям бронювати, спільно користуватися та оплачувати паркомісця, не залучаючи адміністрацію (рисунок 1.6).

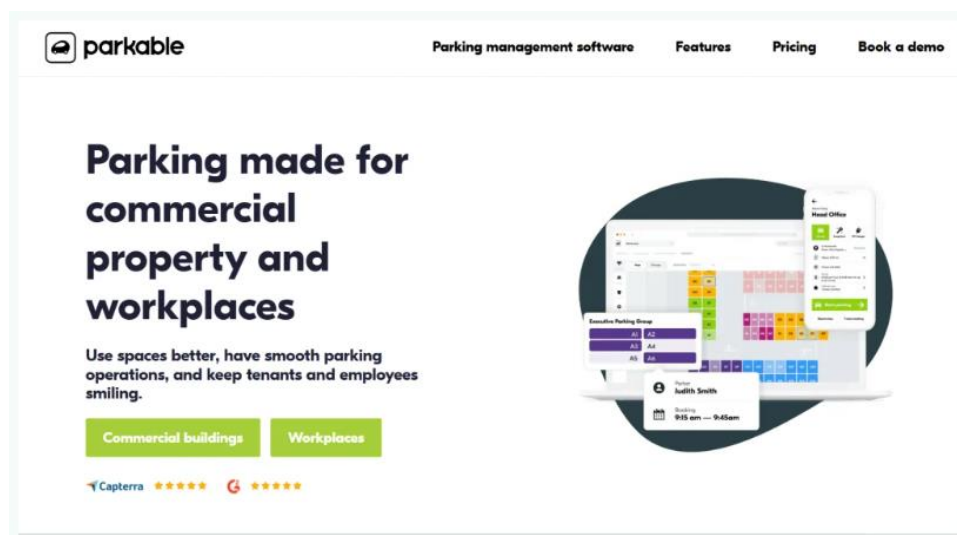


Рисунок 1.6 – Платоформа Parkable [12]

Ключові особливості:

- детальні звіти про використання та отриманий дохід;
- можливість самостійного бронювання;

- інтеграції з системами контролю доступу;
- зарядка для електротранспорту;
- управління паркуванням для відвідувачів;
- системи спільного пулу місць;
- інтерактивна карта.

Переваги:

- користувачі можуть резервувати та здійснювати оплату через мобільний додаток;
- мобільний додаток має простий та зручний інтерфейс;
- висока оперативність служби підтримки;
- функція самообслуговування зменшує адміністративне навантаження.

Недоліки:

- пропонує лише можливості для управління паркувальним простором;
- інтерактивна карта доступна лише для перегляду з адміністративного інтерфейсу користувача.

Підбиваючи підсумки зробленого аналізу, варто констатувати: кожна із проаналізованих платформ має свої особливості використання та націлена на різні сегменти споживачів [12].

Вибір певної платформи визначається габаритами паркуального майданчика, фінансовим планом та функціональними потребами користувачів.

З огляду на представлені ознаки, логічне буде зіставити їх за найбільш важливими критеріями.

Порівняльний аналіз розглянутих платформ представлено в таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз платформ для управління паркуванням [12]

Хар-ка	Awaio	Parkade	Ronspot	ParkingPass	ParkHub	Parkable
Особливості	Спільне використання та управління ресурсами на гібридних робочих місцях.	Підходить для багатоквартирних будинків.	Безкоштовний пробний період.	Управління громадськими парковками.	Для університетів та розваг, або спортивних заходів.	Призначений для комерційних будівель та робочих місць.
Відмінності від аналогів	Динамічний обмін для обмеженої кількості місць.	Динамічне ціноутворення Цілодобова підтримка.	Розподіл простору за допомогою системи бронювання.	Цифровий дозвіл на паркування.	Інтегрується з основними платформами продажу квитків.	Алгоритм Fairchance, який розподіляє місця між користувачами.
Вимоги до обладнання	Немає	Так	Так	Так	Так	Так
Ціноутворення	Модульне та гнучке.	Недоступно для громадськості.	Наявні безкоштовні та платні тарифні плани	Недоступно для громадськості.	Недоступно для громадськості.	Недоступно для громадськості.
Аналіз та звітність	Так	Так	Так	Так	Так	Так

Результати проведеного аналізу показали можливості сучасних технологій. Зараз у користувачів є можливість обирати з величезної кількості доступних рішень. Крім того, можна стверджувати, що розглянуте програмне забезпечення, має низку характерних вад, притаманних переважній більшості проаналізованих систем. Таким чином, виникає нагальна потреба у розробці такого програмного продукту, що здатен був би суттєво нівелювати виявлені недоліки.

1.5 Висновки за розділом 1

У даному розділі було проведено ґрунтовний аналіз предметної області, пов'язаної з організацією та управлінням паркувальними майданчиками. Розглянуто сутність паркувальних систем, їх роль у сучасній міській інфраструктурі та основні проблеми, що виникають у процесі їх функціонування. Встановлено, що зростання кількості транспортних засобів та обмеженість паркувального простору обумовлюють необхідність впровадження автоматизованих інформаційних рішень.

У ході дослідження проаналізовано існуючі способи організації паркування та оплати за користування паркувальними місцями, а також визначено основні підходи до побудови систем управління паркінгом. Особливу увагу приділено сучасним технологіям, зокрема використанню мобільних застосунків.

Також було здійснено класифікацію програмного забезпечення для управління паркуванням та визначено ключові функціональні вимоги до таких систем, серед яких моніторинг доступності місць, автоматизація бронювання, контроль доступу та аналітика використання ресурсів. Проведений огляд сучасних програмних рішень показав, що кожна платформа має свої переваги та обмеження залежно від сфери застосування, рівня автоматизації та вартості впровадження.

Таким чином, результати аналізу підтверджують доцільність розробки власного програмного забезпечення для обліку паркувальних місць, яке враховуватиме виявлені недоліки існуючих рішень та забезпечуватиме ефективне управління паркувальними ресурсами.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

На сьогоднішній день при створенні програмного забезпечення, спрямованого на автоматизацію обліку місць паркування автомобілів, одним із критичних етапів є вибір мови програмування. Від цього залежать такі аспекти, як функціональність системи, її швидкість роботи, простота подальшого супроводу та потенціал для розширення. Враховуючи поставлені завдання та особливості майбутнього застосунку, було прийнято рішення зупинитися на TypeScript як основній мові програмування [13].

TypeScript являє собою мовне розширення для JavaScript, що базується на ньому. Він вводить концепцію статичної типізації разом із низкою додаткових функцій, які сприяють розробці більш стійких та розширюваних програмних рішень. Незважаючи на свої розширення, TypeScript абсолютно сумісний із загальноприйнятими нормами JavaScript, що забезпечує безшовну інтеграцію в уже існуючі кодові бази. Порівняння мов TypeScript та JavaScript представлено в таблиці 2.1.

Таблиця 2.1 – Порівняння мов TypeScript та JavaScript

Критерій	TypeScript	JavaScript
Типізація	Підтримка статичної типізації	Динамічна типізація
Помилки	Помилки виявляються на етапі компіляції	Помилки виявляються під час виконання
Розширення	Надмножина JavaScript, що забезпечує зворотну сумісність	Базова мова програмування
Інструменти	Хороша підтримка IDE та інших інструментів розробки	Великий вибір інструментів і бібліотек
Сумісність	Можна використовувати JavaScript-код	Можливе використання TypeScript у JavaScript-проектах
Розробка	Дає змогу створювати більш надійний і підтримуваний код	Швидке розроблення та прототипування

Кінець таблиці 2.1

Критерій	TypeScript	JavaScript
Сучасні можливості	Підтримка сучасних стандартів JavaScript	Необмежений доступ до можливостей JavaScript

TypeScript є розширенням JavaScript, що означає, що увесь ваш наявний код на JS можна використовувати без жодних змін. Він запроваджує статичне типізування, яке сприяє виявленню помилок ще на етапі компіляції, що веде до створення більш стійких програм. Проте, варто зауважити, що для опанування особливостей TypeScript потрібен певний час на навчання та пристосування [13].

JavaScript, у свою чергу, слугує фундаментальною мовою для веб-розробки. Він оперує динамічною типізацією, згідно з якою типи змінних визначаються автоматично безпосередньо під час виконання програми. JS пропонує широкий арсенал інструментів та бібліотек, а також дає можливість швидко створювати прототипи та розгортати готові додатки.

Незалежно від того, який вибір буде зроблено, обидві мови можуть похвалитися значною підтримкою з боку інструментарію розробки та мають жваві спільноти програмістів. Це гарантує доступ до навчальних матеріалів, повної документації та оперативної допомоги за потреби.

У TypeScript надається можливість формально визначати типи для змінних, функцій та об'єктів. Ці типи можуть бути як базовими (наприклад, `number`, `string`, `boolean`), так і складнішими.

Застосування класів дозволяє структурувати об'єкти, надаючи їм певний набір властивостей та методів. Класи в TypeScript можуть містити конструктори, методи, поля, а також підтримують концепції успадкування та реалізації інтерфейсів [13].

Інтеграція типів і класів у TypeScript дає змогу створювати код, який є одночасно структурованим і типізованим, що підвищує його надійність та легкість для читання. Компілятор TS ретельно перевіряє узгодженість типів,

допомагаючи усунути помилки на ранніх стадіях проєктування. Класи, зі свого боку, спрощують створення об'єктів з визначеними характеристиками та функціоналом, підтримуючи успадкування та інтерфейси, що робить розробку більш гнучкою.

Завдяки використанню типів і класів, TypeScript забезпечує більший контроль над кодовою базою, сприяючи її впорядкованості та зрозумілості.

Насамперед, статичне типізування дає змогу фіксувати помилки ще на етапі розробки. Це допомагає відсікти значну кількість типових помилок, пов'язаних із невідповідністю типів даних, і суттєво підвищує загальну стійкість коду [13].

TypeScript має відмінну інтеграцію з інструментами розробки, зокрема з інтегрованими середовищами (IDE) та редакторами коду. Статичне типізування дозволяє IDE надавати автодоповнення, підказувати інформацію про типи та пропонувати інші корисні функції, які спрощують процес кодування.

Крім того, TypeScript може похвалитися прекрасною документацією та активною спільнотою розробників. Це означає, що завжди можна швидко знайти допомогу та готові рішення проблем на онлайн-платформах, форумах чи у блогах.

Компілятор TypeScript – це утиліта, яка трансформує код, написаний на TypeScript, у звичайний JavaScript. Він виконує перевірку типів, а після цього генерує відповідний JavaScript-код, придатний для виконання у будь-якому середовищі, сумісному з JavaScript [13].

Компілятор TypeScript також надає можливість налаштовувати численні параметри компіляції, такими як визначення цільової версії JavaScript, рівень суворості перевірки типів та низку інших опцій. Це дозволяє розробникам адаптувати процес компіляції відповідно до специфічних вимог та особливостей їхнього проєкту [13].

Загалом, компілятор є ключовим елементом у циклі розробки на TypeScript. Він відповідає за валідацію типів, конвертацію коду у JavaScript та

тонке налаштування параметрів компіляції, перетворюючи TypeScript у потужний інструмент для створення надійних та масштабованих програмних рішень.

Переваги та недоліки TypeScript представлені в таблиці 2.2.

Таблиця 2.2 – Переваги та недоліки TypeScript

Параметр	Переваги	Недоліки
Статична типізація	Дає змогу виявляти помилки на етапі компіляції	Необхідність додавання типів може потребувати додаткового часу і зусиль
Покращена підтримка інструментів	Хороша інтеграція з IDE та іншими інструментами розробки	Деякі інструменти можуть не повністю підтримувати TypeScript
Сумісність із JavaScript	Можна використовувати наявний JavaScript-код у TypeScript	Може знадобитися час на вивчення синтаксису TypeScript
Підтримка сучасних можливостей JavaScript	Можна використовувати сучасні стандарти JavaScript	Підтримка деяких сучасних можливостей може бути обмеженою
Покращена підтримка великих проєктів	Структурує кодову базу і підвищує її підтримуваність	У деяких випадках може знадобитися більше часу на розробку і підтримку коду
Легке впровадження в існуючий проєкт	Можна поступово додавати TypeScript в наявний JavaScript-код	Необхідно забезпечити навчання і прийняття TypeScript у команді розробників
Активна спільнота та документація	Широка спільнота розробників та навчальні ресурси	Деякі специфічні питання можуть потребувати додаткового дослідження

Використання TypeScript має свої плюси і мінуси, і вибір між TypeScript та іншими мовами програмування залежить від вимог і цілей проєкту.

Таким чином, після аналізу функціональні вимоги, продуктивності, потенціалу для розширення системи, стає очевидним, що TypeScript найкращим вибором для створення програмного забезпечення, призначеного для автоматизації обліку місць паркування автомобілів.

2.2 Вибір фреймворку для створення програмного забезпечення для автоматизації обліку місць паркування автомобілів

При створенні програмного забезпечення інформаційної системи для автоматизації обліку місць паркування автомобілів вирішальну роль відіграє вибір фреймворку. Саме він визначає зручність взаємодії користувача із системою, швидкість доступу до функцій та загальне враження від роботи з продуктом. Зважаючи на функціональні потреби для реалізації програмного продукту в якості фреймворку було обрано React Native [14].

React Native являє собою інструментарій для створення мобільних застосунків, заснований на базі JavaScript-бібліотеки React. Подібно до React, він є проєктом з відкритим кодом, що дає змогу формувати динамічні користувацькі інтерфейси, використовуючи або JavaScript, або TypeScript. Хоча синтаксис і функціональність у нього спільні з React, тут застосовуються "рідні" (нативні) будівельні блоки інтерфейсу (UI) з цільової платформи, а не елементи HTML/CSS чи готові компоненти з бібліотек React. Приклад блок-схеми додатку React Native зображено на рисунку 2.1.

Вбудована сумісність із React Native передбачена для Apple iOS/iPadOS та Android, а його функціональність можна розширити додатковими модулями для застосунків під Windows і MacOS, а також для вебу й Electron за допомогою React Native for Web [14].

React Native виступає як помітна альтернатива застосуванню винятково нативних інструментів, призначених для розробки мобільних застосунків під iOS та Android. Замість того, щоб підтримувати дві незалежні кодові бази (наприклад, Kotlin чи Java для Android, Objective-C чи Swift для пристроїв Apple), розробникам достатньо написати бізнес-логіку та спроектувати інтерфейс засобами React Native. Цей фреймворк автоматично візьме необхідні UI-складові (включно з кнопками, формами введення, списками та іншими інтерактивними елементами) безпосередньо з операційної системи, на якій працює додаток. Це дає змогу мати єдину кодову основу для всього

проєкту, незалежно від платформи, істотно знижуючи витрати на подальший супровід коду.

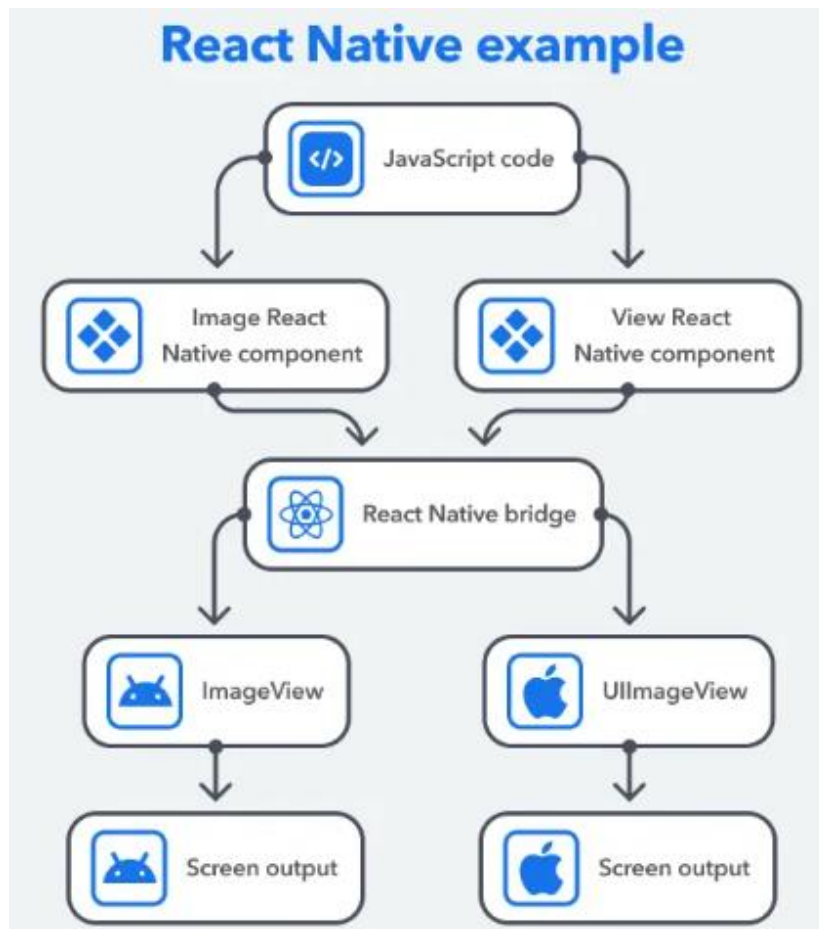


Рисунок 2.1 – Приклад блок-схеми додатку React Native [14]

React Native не обмежується лише наданням доступу до нативних елементів інтерфейсу: тоді як React спеціалізується на візуальному поданні, React Native також пропонує API для взаємодії з хост-ОС. Це допомагає у створенні кросплатформних рішень, які здатні повною мірою використовувати можливості пристрою [15].

React Native спрощує процес створення застосунків, придатних для багатьох платформ, завдяки спільній кодовій базі, усуваючи потребу у створенні власних кастомних UI-компонентів. Це приносить користь як розробникам, так і кінцевим користувачам.

Кінцеві користувачі отримують застосунок, який виглядає та функціонує подібно до інших оригінальних програм, що гарантує звичний та інтуїтивно зрозумілий досвід.

React Native дає змогу легко представити свій продукт новій аудиторії, просто додавши підтримку додаткових платформ у проєкт React Native. Витрати на розробку також зменшуються, оскільки потрібно обслуговувати лише один набір коду.

Фреймворк має потужну спільноту, яка надає бібліотеки, навчальні матеріали та форуми. Це полегшує пошук допомоги та використання чужих напрацювань, що прискорює розробку. Як проєкт з відкритим кодом, React Native також надає свободу використовувати його у будь-який зручний спосіб без фінансових обмежень [15].

Завдяки використанню нативних UI-компонентів, React Native може демонструвати кращу продуктивність, ніж застосунки, які змушені рендерити власний HTML/CSS. Проте, оскільки він не перетворюється в повністю нативний код, він може поступатися у швидкодії абсолютно нативним програмам.

React здобув популярність тому, що він робить розробку на JavaScript значно простішою та результативнішою завдяки модульності та можливості багаторазового використання коду. Наявні знання React можна ефективно застосувати для створення мобільних застосунків через React Native, полегшуючи адаптацію та перенесення навичок між різними проєктами.

Для розробки застосунків на React Native можуть бути використані вже наявні інструменти розробки, зокрема найпоширеніші набори засобів Node.js та JavaScript [15].

Для підтвердження обґрунтованості вибору React Native, було проведено порівняння з фреймворком Flutter. Результати цього порівняння містяться у таблиці 2.3.

Таблиця 2.3 – Обґрунтування вибору фреймворку для створення програмного забезпечення автоматизації обліку місць паркування автомобілів

Функція фреймворку	React Native	Flutter
Кросплатформний	Дозволяє розробникам написати логіку JavaScript один раз та створювати її як для Android, так і для iOS, використовуючи нативні компоненти для узгодженого рендерингу інтерфейсу користувача.	Пропонує справжню кросплатформну розробку, рендерячи все за допомогою власного движка на базі Skia, забезпечуючи майже нативну продуктивність.
Мова програмування	Використовує JavaScript, широко відомий та використовуваний, що полегшує перехід для веб-розробників.	Використовує Dart, менш поширену, але спеціально розроблену мову з такими функціями, як AOT та JIT-компіляція для високої продуктивності.
Рендеринг інтерфейсу користувача	Використовує нативні компоненти платформи, забезпечуючи, щоб програми виглядали та відображалися як нативні програми, але може вимагати об'єднання для розширених функцій.	Використовує власні віджети, що відображаються через Skia, забезпечуючи узгоджений інтерфейс користувача на різних платформах, але вимагаючи налаштування для роботи, подібної до нативної.
Гаряче перезавантаження	Забезпечує гаряче перезавантаження для швидких змін інтерфейсу користувача, але може не відображати зміни стану програми послідовно.	Пропонує гаряче перезавантаження з майже миттєвим зворотним зв'язком щодо змін, включаючи віджети зі збереженням стану, що робить його надійнішим для швидких ітерацій.
Продуктивність	Залежить від JavaScript, який не компілюється в машинний код, що іноді може призвести до збільшення продуктивності.	Компілює код Dart у власний машинний код, забезпечуючи швидшу продуктивність.
Екосистема та бібліотеки	Велика екосистема сторонніх бібліотек завдяки популярності JavaScript, хоча якість бібліотек може відрізнятися.	Менша екосистема, але швидко зростаюча, з офіційною підтримкою багатьох поширених випадків використання та сильним акцентом на якості.
Підтримка громади	Підтримується Meta, з великою та зрілою спільнотою розробників та учасників по всьому світу.	Підтримується Google, має меншу, але швидкозростаючу та дуже активну спільноту.
Рідна інтеграція	Потужна підтримка інтеграції з існуючими нативними програмами, але поєднання складних нативних модулів може вимагати додаткових зусиль.	Обробляє нативні інтеграції за допомогою каналів платформи, пропонуючи хорошу продуктивність, але вимагаючи більше зусиль для складних налаштувань.
Налаштування інтерфейсу користувача	Зменшення налаштування інтерфейсу користувача, оскільки використовуються нативні компоненти для узгодженого вигляду	Повністю настроювані користувацькі інтерфейси.
Крива навчання	Легше для тих, хто вже знайомий з JavaScript та React.	Використовує мову програмування Dart, яка має менший рівень впровадження, але надає багато готових функцій (наприклад, віджети), які ви можете використовувати.

Таким чином, вибір React Native для ПЗ автоматизації обліку місць паркування автомобілів є логічним та виправданим.

2.3 Висновки за розділом 2

В даному розділі було здійснено аналіз наявних підходів до створення програмного забезпечення, призначеного для автоматизації обліку місць паркування автомобілів. Обґрунтовано вибір ключових технологічних рішень. Ці рішення покликані гарантувати функціональність, стійкість та здатність системи до розширення у майбутньому.

Зокрема, у якості мови програмування обрано TypeScript. Це сучасна надбудова над JavaScript, яка поєднує його гнучкість із перевагами, що надає статична типізація. Впровадження TypeScript сприяє підвищенню якості програмного коду через виявлення помилок на ранніх етапах, покращує візуальну структуру та зрозумілість програмних блоків, а також полегшує подальше обслуговування та еволюцію системи. Завдяки повній сумісності з JavaScript та широкому спектру доступних інструментаріїв, ця мова програмування є найкращим варіантом для конструювання надійних програмних рішень.

У процесі дослідження було проведено розгляд актуальних фреймворків. В якості основного інструменту обрано React Native, що дає змогу розробляти мобільні додатки для різних платформ, використовуючи при цьому єдиний набір вихідного коду. Це економить як часові, так і матеріальні ресурси, потрібні для створення та супроводу кінцевого продукту. Порівняльний аналіз із фреймворком Flutter підтвердив, що React Native є виправданим для цього проекту, враховуючи спорідненість технологій, об'ємну бібліотеку готових рішень та активне співтовариство розробників.

Таким чином, обрана технологічна парадигма – TypeScript у зв'язці з React Native – повністю відповідає усім необхідним критеріям для розробки ПЗ, що автоматизує керування парковками, забезпечуючи необхідний

компроміс між швидкістю роботи, зручністю розробки та потенціалом для майбутнього масштабування проєкту.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення

Застосунок складається з статичних елементів та реактивного компоненту `App.tsx`, який слугує точкою входу, де відбувається обробка та відображення інформації. Завдяки обробці даних користувач отримує їх у зручному графічному форматі.

Мобільна розробка базується на шести ключових складових-екранах:

- сторінка логіну;
- сторінка реєстрації;
- головна сторінка з картою де знаходяться парко-місця;
- сторінка з інформацією про паркінг;
- сторінка розрахунку за парко-місце;
- сторінка з qr-code квитком.

Для реалізації додатку був створений API, який знаходиться на створеному віддаленому сервері. Найбільш доцільним API для даного застосунку є Firebase.

Google Firebase — це платформа від Google, що дає змогу створювати застосунки для iOS, Android та вебу. Firebase надає набір інструментів для моніторингу аналітики, формування звітів, усунення помилок у програмах, а також для проведення маркетингових та продуктових експериментів [16].

Система баз даних Firebase пропонує широкий перелік можливостей; у створеному застосунку було інтегровано функціонал автентифікації та бази даних, що працює в режимі реального часу.

Автентифікація Firebase дає змогу розробникам легко впроваджувати надійні схеми верифікації, покращуючи процес входу та адаптації для користувачів. Цей інструмент забезпечує комплексне рішення для ідентифікації, підтримуючи реєстрацію через електронну пошту та пароль, підтвердження через мобільний телефон, а також вхід через акаунти Google, Facebook, GitHub, Twitter та багато інших сервісів [16].

Для розробки веб-інтерфейсу використано API, розташоване на хмарному сховищі, яке оперує моделлю даних, сфокусованою на документах (база даних типу NoSQL).

Бази даних типу NoSQL спеціально проєктуються під конкретні моделі даних та мають гнучкі схеми, що сприяє розробці сучасних програмних рішень.

NoSQL бази даних здобули значну популярність завдяки простоті розробки, високій функціональності та продуктивності незалежно від масштабу проєкту. Вони дозволяють зберігати прості пари "ключ-значення" для швидкого кешування, а також вміщувати неструктуровані масиви даних, якими можна оперувати за допомогою мови структурованих запитів SQL [17].

Документо-орієнтовані NoSQL бази даних користуються великою популярністю у розробників. Ці системи керування базами даних схожі на сховища на основі стовпців, але надають можливість створювати складніші структури [16].

Документи усувають деякі обмеження, властиві сховищам стовпців. Загалом, вони дозволяють конструювати документи з довільною структурою даних будь-якого рівня складності.

Для створення програмного забезпечення, призначеного для автоматизації обліку паркувальних місць, було обрано архітектуру односторінкового застосунку – SPA (Single Page Application). На рисунку 3.1 приведена загальна структурна схема роботи додатку [18].

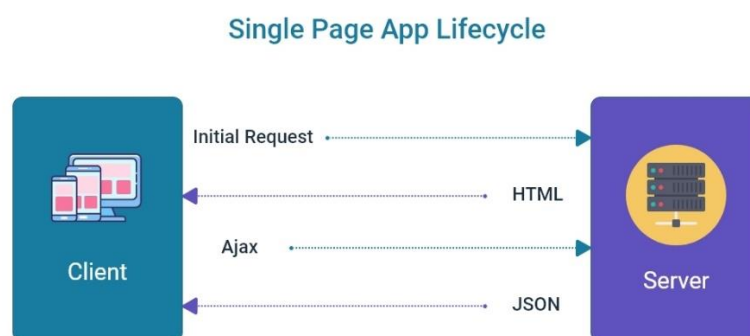


Рисунок 3.1 – Загальна структурна схема роботи додатку [18]

Застосунок типу SPA (Single Page Application) являє собою таку форму веб-додатку, де увесь контент демонструється на одній сторінці, що унеможливорює її повне оновлення, завдяки чому користувачі отримують більш динамічний та швидкий досвід. Архітектуру SPA формує низка ключових складових: клієнтська частина, серверна частина, інтерфейс REST API, сховище даних (база даних), а також механізми обміну даними між цими елементами [18].

Ключова розбіжність між класичними веб-додатками та односторінковими застосунками (SPA) полягає у підході до оновлення та рендерингу вмісту інтерфейсу для користувача.

У традиційних додатках, коли користувач виконує певну дію, веб-сторінка обов'язково перезавантажується, що тягне за собою повну реініціалізацію всього візуального оточення. Для відображення змін серверу необхідно передавати щоразу новий HTML-документ із свіжою інформацією, що може спричинити помітні паузи та навіть призвести до втрати даних, які встиг ввести користувач. Натомість, у SPA вся потрібна інформація завантажується на одну сторінку під час першого запуску програми, а всі наступні маніпуляції користувача опрацьовуються без необхідності повного оновлення сторінки [18].

3.2 Основний алгоритм роботи програмного забезпечення

Алгоритм роботи розробленого програмного забезпечення для обліку місць паркування автомобілів визначає послідовність дій користувача та системи, починаючи з моменту запуску мобільного додатку і завершуючи отриманням підтвердження оплати у вигляді QR-коду. Побудована логіка функціонування забезпечує безперервну взаємодію між інтерфейсом користувача, навігаційною підсистемою та сервісами автентифікації. Основний алгоритм роботи ПЗ зображений на рисунку 3.2.

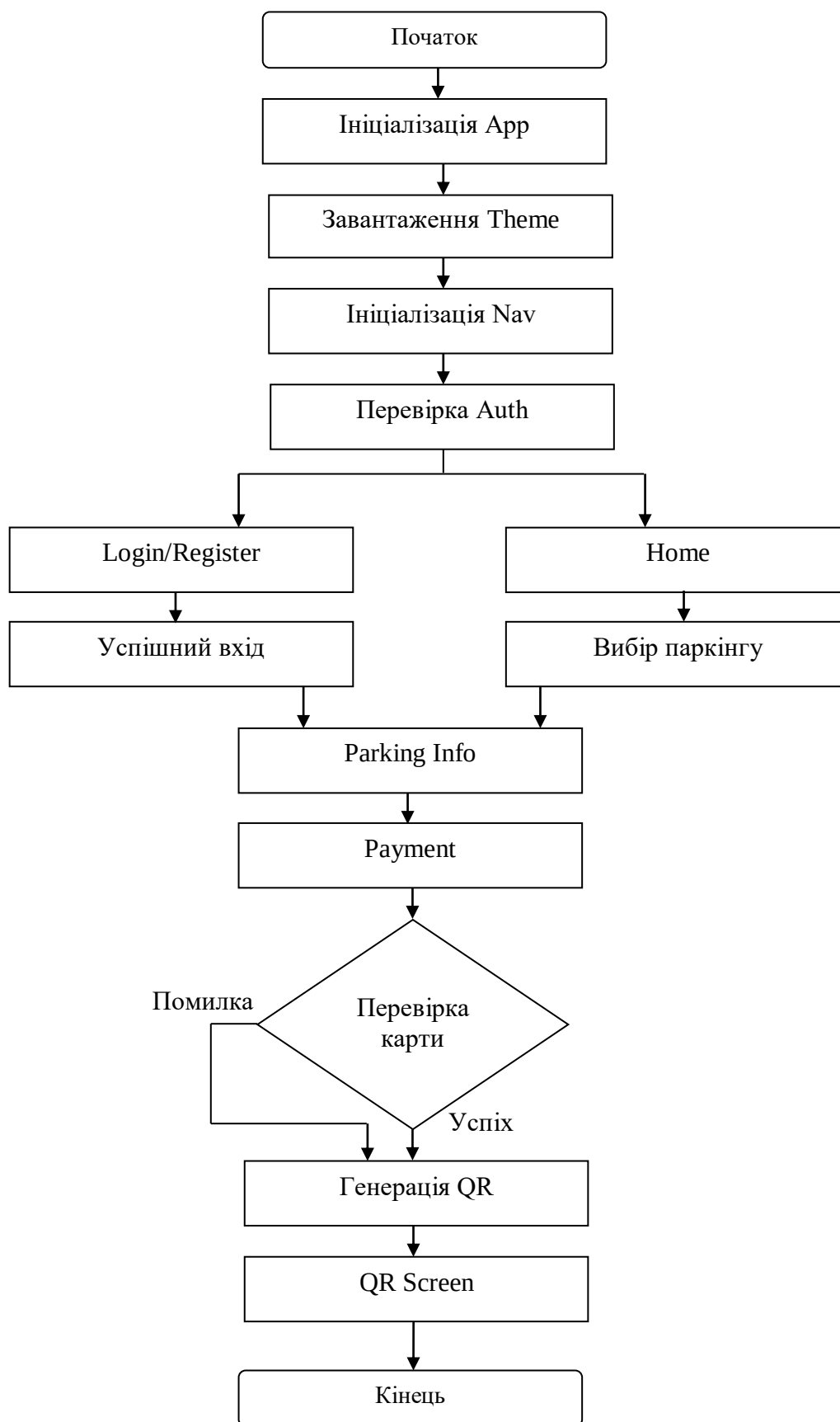


Рисунок 3.2 – Основний алгоритм роботи програмного забезпечення

На початковій стадії відбувається запуск програми. Після ініціалізації головного модуля застосовуються глобальні стилі оформлення, що забезпечує однаковість вигляду усіх елементів керування. Паралельно формується структура навігації, яка визначає допустимі шляхи переміщення між різними вікнами програми.

Критичним моментом є перевірка статусу авторизації особи, яка користується додатком. Для цієї мети використовується механізм, що реагує на зміни стану авторизації. Якщо активна сесія відсутня, програма автоматично перенаправляє користувача на форму входу чи реєстрації. Натомість, якщо користувач уже здійснив вхід, відбувається перехід до основного функціоналу програми.

Процедура авторизації вимагає введення облікових даних (адреси електронної пошти та пароля) та подальшої їхньої передачі до відповідного сервісу ідентифікації. У разі помилкового введення даних користувач отримує сповіщення про помилку. Реєстрація нового користувача містить додаткову валідацію коректності введених паролів, що допомагає запобігти помилкам при створенні профілю.

Після успішного входу відкривається головний екран системи, де відображається динамічна мапа. На цьому етапі за допомогою геопозиційного сервісу визначається поточне місцезнаходження користувача. Доступні для паркування ділянки демонструються на карті спеціальними позначками.

Вибір конкретного паркувального місця ініціюється натисканням на відповідній позначці. Після цього активується вікно з детальною інформацією, яка охоплює назву об'єкта, його вартість та додаткові відомості. Користувачеві надається можливість перейти до оплати наданої послуги.

На стадії транзакції користувач вводить реквізити своєї банківської картки. Система виконує первинну перевірку правильності введених даних, зокрема контролює довжину номера картки. Якщо введені дані не відповідають встановленим критеріям, відображається повідомлення про помилку, і користувачеві пропонується ввести дані повторно.

Якщо перевірка пройдена успішно, процес переходить до етапу підтвердження фінансової операції. Система створює унікальний ідентифікатор замовлення, на основі якого формується QR-код. Цей код слугує електронним підтвердженням оплати та може бути пред'явлений для контролю доступу до паркувальної зони.

На фінальному етапі користувач може повернутися на головний екран, щоб знову обрати паркування або завершити роботу з додатком. Передбачена також функція виходу з облікового запису, що повертає систему до екрана авторизації.

Таким чином, розроблений алгоритм забезпечує прозору та логічно впорядковану взаємодію користувача із системою, охоплюючи всі основні кроки: від ідентифікації до отримання підтвердження оплати. Впроваджена структура сприяє підвищенню зручності користування та надійності функціонування програмного продукту.

3.3 Проєктування інтерфейсу

На сучасному етапі розвитку ПЗ особлива увага приділяється розробці взаємодії кінцевого користувача з програмою. Адже саме від того, наскільки якісним буде інтерфейс, напряму залежить конкурентоспроможність фінального продукту та його прийняття користувачами. Інтерфейс повинен бути легким для розуміння без зайвих пояснень, зручним у щоденній роботі, мати привабливу та збалансовану колірну гаму, а також постійно інформувати користувача про актуальний стан керованих ним систем. При цьому є важливим уникати появи на екрані надмірної кількості елементів візуалізації. Не менш суттєвими є також естетичні аспекти оформлення та неухильне дотримання типових стандартів і правил, які висуваються до побудови інтерфейсів.

Інтерфейсна частина, втілена в програмному комплексі, була створена із залученням ресурсів фреймворку React Native. Архітектурно інтерфейс поділяється на такі ключові розділи:

- сторінка логіну;
- сторінка реєстрації;
- головна сторінка з картою де знаходяться парко-місця;
- сторінка з інформацією про паркінг;
- сторінка розрахунку за парко-місце;
- сторінка з qr-code квитком.

На рисунку 3.3 наведено концепт головного вікна додатку.

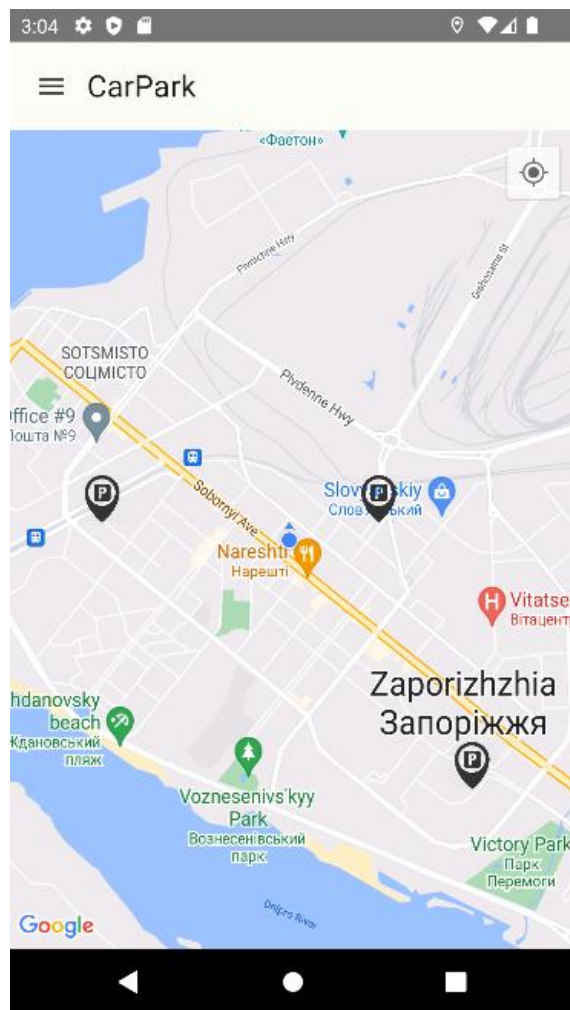


Рисунок 3.3 – Концепт головного вікна додатку

3.4 Висновки за розділом 3

У даному розділі було детально розглянуто особливості побудови та функціонування розробленого програмного забезпечення для обліку паркувальних місць. Проведений аналіз дозволив сформулювати цілісне уявлення про структуру застосунку, його архітектурні рішення та логіку взаємодії між окремими компонентами системи.

Зокрема, визначено, що програмний продукт побудований за принципами односторінкової архітектури, що забезпечує швидке реагування інтерфейсу на дії користувача та мінімізує необхідність повторного завантаження даних.

Описана структура програмного забезпечення, розглянутий алгоритм роботи системи.

Окрема увага була приділена проєктуванню інтерфейсу, який орієнтований на забезпечення зручності та інтуїтивної зрозумілості.

Таким чином, результати, отримані в межах даного розділу, свідчать про те, що розроблене програмне забезпечення відповідає сучасним вимогам до мобільних застосунків, забезпечує стабільну роботу основних функцій та створює зручні умови для користувачів. Запропоновані рішення можуть бути використані як основа для подальшого вдосконалення системи та розширення її функціональних можливостей.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Мобільний додаток призначений для автоматизації обліку місць паркування автомобілів. Користувачі зможуть реєструватися, швидко знайти паркінг та забронювати місце та прокласти до нього маршрут.

Всі дані, які використовувалися для розробки додатка, надає API який знаходиться на сервері, тому всі подальші зміни в даних автоматично відобразяться в додатку.

Веб-додаток виконує наступні функції:

- реєстрація та авторизація користувачів;
- надання зручного користувацького інтерфейсу;
- завантаження парко-місць на карті;
- вибір парко-місця;
- виведення інформації про парко-комплекс;
- реалізація форми оплати банківською картою;
- сторінка з qr-code квитком.

Додаток створений в середовищі програмування Visual Studio Code за допомогою мови програмування TypeScript та фреймворку React Native.

Додаток знаходиться на віддаленому репозиторії. Для того щоб зберегти проєкт локально необхідно завантажити його, виконавши команду “git clone «посилання на репозиторій»”.

Після завантаження проєкту, щоб запустити додаток, в командному рядку треба виконати “npm i”, за допомогою чого будуть встановлені всі модулі необхідні для функціонування додатка. Після цього залишилось виконати “npm run start”, для запуску проєкту.

Вхідними даними до програми є дані про парко-місце, назва, адреса, опис, ціна, а також дані банківської карти.

Вихідними даними ПЗ є візуальна екранна форма із qr-кодом.

4.2 Умови виконання програми

Основною вимогою до технічних засобів використання програми є наявність пристрою на базі операційної системи Android 5.0 (Lollipop) і вище, наявність вільного простору у пам'яті пристрою, 1 Гб оперативної пам'яті для коректної і плавної роботи та наявність стабільного інтернет-з'єднання.

Для відладки програми використовується емулятор Android Studio на базі операційної системи Android 12.0 (API 32), який також володіє додатковим функціоналом для відстежування критичних помилок у коді.

4.3 Виконання програмного забезпечення

Для запуску мобільного додатку в емуляторі потрібно відкрити папку проекту у Microsoft Visual Code та виконати команди `npm start`, `npx react-native run-android`. Після чого відкриється емулятор, на який завантажиться додаток.

Одразу після завантаження додатку відкриється екран авторизації, що зображений на рисунку 4.1. Пропонується увійти в існуючий обліковий запис або створити новий. При цьому відбувається верифікація електронної пошти відповідно до стандартного формату. Під час авторизації та реєстрації передбачена перевірка на помилки при введенні даних.

Після успішної авторизації, користувач потрапляє на головний екран з картою (рисунок 4.2). Одразу після входу, починає відстежуватись геолокація користувача, для того щоб відобразити своє місцезнаходження на карті, треба натиснути у верхньому правому куті на іконку геолокації, після чого на головному екрані відобразиться маркер користувача та можна буде обрати на карті найближче парко-місце. Для того щоб вийти з аккаунту, треба у верхньому лівому куті натиснути на меню, далі натиснути на кнопку «Log Out».

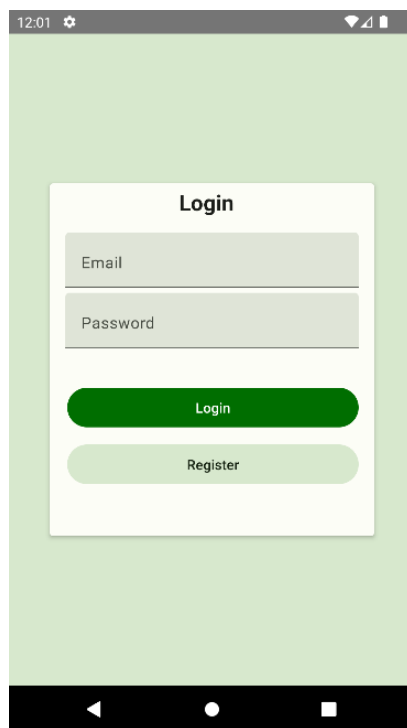


Рисунок 4.1 – Сторінка авторизації

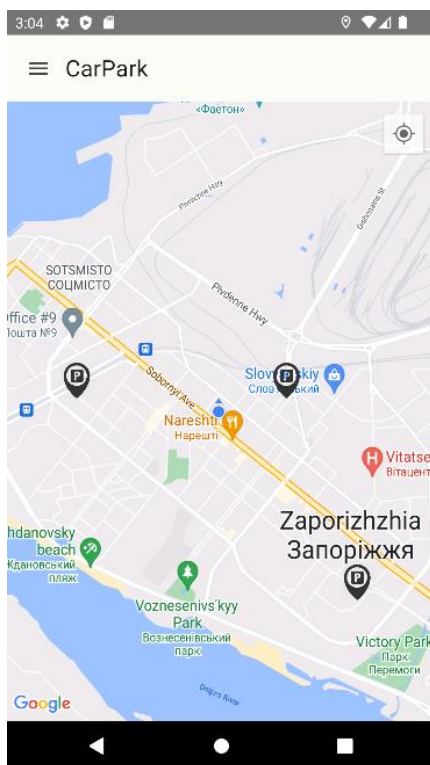


Рисунок 4.2 – Головна сторінка додатку

Коли користувач натисне на маркер парко-місця, відкриється сторінка, де буде відображатися коротка інформація про парко-місце, а саме: фото

паркінгу; назва; рейтинг; адреса де знаходиться парко-місце, стислий опис парко-містя, та ціна парко-містя. Також на цій сторінці присутня кнопка повернення на головну сторінку (рисунок 4.3).



Рисунок 4.3 – Сторінка опису парко-містя

При натисненні на кнопку «Pay» здійснюється перехід на сторінку з оплатою, що зображена на рисунку 4.4. За допомогою кнопки «Cancel Payment» можна повернутися назад до сторінки з описом парко-містя. В програмі передбачена перевірка на випадок введення неправильних даних

карти. Якщо користувач нічого не введе у поля, то про це буде виведено помилку (рисунок 4.5).

5:11

☰ Make Payment

Number Card

2445 6534 2231 2132

MM/YY 12/25 CVV 111

Cancel Payment

Confirm Payment

Рисунок 4.4 – Сторінка оплати

The screenshot shows a mobile application interface for making a payment. At the top, the status bar displays the time 3:06 and various system icons. Below the status bar is a header with a hamburger menu icon and the text 'Make Payment'. The main content area is a light green rounded rectangle containing a form. The form has a label 'Nombor Card' above a single-line text input field. Below this, there are two input fields: 'MM/YY' followed by a two-part date selector (month and year), and 'CVV' followed by a three-digit CVV selector. A red error message 'All fields should be filled' is displayed below the form. At the bottom of the form area, there are two buttons: a light green 'Cancel Payment' button and a dark green 'Confirm Payment' button. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps icons.

Рисунок 4.5 – Приклад помилки оплати

Після того, як користувач правильно введе всі дані карти, та натисне на кнопку «Confirm Payment» він потрапляє на сторінку із згенерованим номером квитку та qr-code (рисунок 4.6), який буде потрібно показати на контрольно-пропускному пункті пакувального комплексу, де співробітник відсканує qr-code з телефону користувача. При натисненні на кнопку «Close» користувач повертається на головну сторінку додатку.

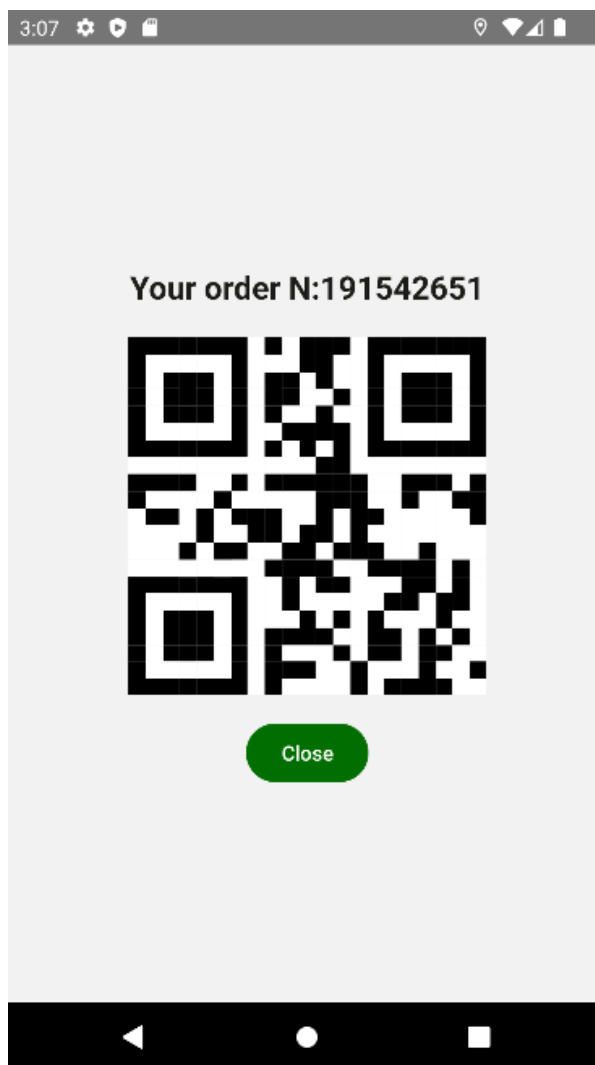


Рисунок 4.6 – Сформований парувальний квиток

4.4 Тестування програмного забезпечення

Тестування розробленого програмного забезпечення є важливим етапом забезпечення його якості, надійності та відповідності функціональним вимогам. Для створеного мобільного застосунку, реалізованого з використанням мови програмування TypeScript та фреймворку React Native, процес перевірки охоплював як окремі компоненти, так і систему в цілому.

На початковому етапі тестування було перевірено правильність функціонування елементів інтерфейсу. Особлива увага була зосереджена на роботі полів для внесення даних, зокрема на сторінках входу та реєстрації. Здійснено перевірку обробки даних, які вносить користувач, як система реагує

на некоректні відомості, а також як виводяться повідомлення про помилки. Цей етап допоміг засвідчитися у безпомилковості реалізації механізмів валідації та зворотного зв'язку з кінцевим користувачем.

Наступним кроком стала перевірка логіки переміщень у додатку (навігації). Було проконтрольовано коректність переходів між різними екранами, зокрема після успішного входу, вибору місця для паркування та завершення транзакції оплати. Окрім того, оцінювалася стійкість роботи механізму навігаційних переходів при поверненні на попередні вікна та при багаторазовому повторенні однакових послідовностей дій [19].

Окрема увага приділялася перевірці взаємодії із зовнішніми сервісами, такими як система автентифікації та база даних. Було протестовано сценарії входу до системи, створення нового облікового запису та виходу користувача. Це дало змогу підтвердити коректність обміну інформацією між клієнтським інтерфейсом та віддаленим сервером.

У рамках функціонального тестування було змодельовано типовий шлях використання застосунку: від моменту входу користувача до отримання ним QR-коду після оплати паркувального місця. Перевірено правильність відображення відомостей про доступні місця, роботу карти, функціональність натискань на маркери та передачу даних між різними екранами.

Також було проведено тестування механізму здійснення платежів, включаючи перевірку правильності введення реквізитів банківської картки та обробку ситуацій, коли виникають помилки. Успішне завершення цієї фази підтверджувалося генерацією унікального QR-коду, який слугує підтвердженням проведеної фінансової операції [19].

Таким чином, результати тестування продемонстрували, що розроблений програмний продукт працює стабільно, належним чином реагує на дії користувача та забезпечує виконання всіх запланованих функцій. Виявлені несуттєві недоліки було усунуто під час подальшого доопрацювання, що позитивно вплинуло на загальний рівень якості та надійності системи.

4.5 Висновки за розділом 4

У межах даного розділу було розглянуто практичні аспекти використання, перевірки працездатності та дослідження розробленого програмного забезпечення для обліку паркувальних місць. Проведений аналіз дозволив оцінити ефективність реалізованих функціональних можливостей, а також визначити умови, за яких система забезпечує стабільну та коректну роботу.

Зокрема, було визначено призначення програмного продукту та окреслено основні сценарії його застосування. Встановлено, що розроблений мобільний додаток забезпечує повний цикл взаємодії користувача із системою. Описані умови експлуатації підтверджують, що ПЗ не потребує складних апаратних ресурсів і може ефективно функціонувати на більшості сучасних мобільних пристроїв за наявності стабільного інтернет-з'єднання.

У процесі дослідження виконання програми було проаналізовано послідовність дій користувача під час роботи з додатком.

Особливу увагу було приділено тестуванню програмного забезпечення, у ході якого перевірено як окремі компоненти системи, так і її роботу в цілому. Результати тестування засвідчили, що застосунок коректно обробляє введені дані, адекватно реагує на помилкові ситуації та забезпечує стабільну взаємодію з зовнішніми сервісами. Проведене функціональне тестування підтвердило правильність реалізації основних сценаріїв використання.

Отже, отримані результати свідчать про те, що розроблене програмне забезпечення є працездатним, надійним та готовим до практичного використання.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було виконано програмну реалізацію застосунку для автоматизації обліку місць паркування автомобілів.

Для цього було виконано ряд завдань:

- розглянуто основні поняття предметної області;
- проведено аналіз способів оплати за користування паркувальним майданчиком;
- виконано опис систем для управління паркувальним майданчиком;
- проведено порівняння існуючого програмного забезпечення для управління паркуванням автомобілів;
- виконано вибір мови програмування та фреймворку для створення програмного забезпечення для автоматизації обліку місць паркування автомобілів;
- сформовано структур програмного забезпечення;
- створено алгоритм функціонування системи;
- виконано проектування інтерфейсу взаємодії користувача з ПЗ;
- розроблено програмне забезпечення для автоматизації обліку місць паркування автомобілів;
- описано процедуру виконання програмного забезпечення;
- проведено тестування додатку.

Розробка програмного забезпечення відбувалася із застосуванням мови програмування TypeScript та фреймворку React Native. Обраний архітектурний підхід односторінкового додатку забезпечує швидке завантаження та динамічне оновлення контенту без потреби повного перезавантаження сторінки, що підвищує загальну якість взаємодії користувача із системою.

Отримані результати тестування розробленого застосунку підтвердили стабільність роботи програми, її відповідність заявленим функціональним вимогам та готовність до застосування на практиці.

Застосунок реалізує такі функціональні вимоги:

- реєстрація та авторизація користувачів;
- надання зручного користувацького інтерфейсу;
- завантаження парко-місць на карті;
- вибір парко-місця;
- виведення інформації про парко-комплекс;
- реалізація форми оплати банківською картою;
- сторінка з qr-code квитком.

В результаті розробки реалізовано систему, що відповідає всім вимогам визначеним на початку розробки у вигляді функціональних вимог.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Parking Reform Will Save the City. *Bloomberg*. URL: <https://www.bloomberg.com/news/articles/2019-09-20/how-to-reform-your-city-s-bad-parking-requirements> (date of access: 06.05.2026).
2. Out, Damned Spot! How D.C.'s Onerous Parking Requirements Slow Development. *Washington Citypaper*. URL: <https://washingtoncitypaper.com/article/377241/out-damned-spot-how-d-c-s-onerous-parking-requirements-slow-development/> (date of access: 09.05.2026).
3. Invisible Bicycles: Tokyo's High-Tech Underground Bike Parking. *Web urbanist*. URL: <https://weburbanist.com/2015/03/26/invisible-bicycles-tokyo-high-tech-underground-bike-parking/> (date of access: 09.05.2026).
4. Parking lot. *Wikipedia*. URL: https://en.wikipedia.org/wiki/Parking_lot (date of access: 09.05.2026).
5. Servant or snoop in the parking garage. *Los Angeles Times*. URL: <https://www.latimes.com/local/la-xpm-2011-jan-23-la-me-santa-monica-parking-20110123-story.html> (date of access: 10.05.2026).
6. Smartphone-Based Real Time Vehicle Tracking in Indoor Parking Structures / R. Gao et al. *IEEE Transactions on mobile computing*. 2017. Vol. 16, no. 7. URL: <https://www.cis.upenn.edu/~mingminz/papers/vetrack-tmc-paper.pdf> (date of access: 10.05.2026).
7. Stormwater Best Management Practice (BMP) Handbooks. *California Stormwater Quality Association*. URL: <http://www.cabmphandbooks.com/> (date of access: 10.05.2026).
8. Green Parking Lot Resource Guide. *U.S. Environmental Protection Agency*. URL: <https://nepis.epa.gov/Exe/ZyPDF.cgi/P100D97A.PDF?Dockey=P100D97A.PDF> (date of access: 10.05.2026).
9. Parking app Pango making tracks in NYC, adding 20 more garages. *Daily News*. URL: <https://www.nydailynews.com/2013/08/06/parking-app-pango->

making-tracks-in-nyc-adding-20-more-garages-over-next-few-weeks/ (date of access: 12.05.2026).

10. Pango Mobile Parking App Coming To Philadelphia. *PR Newswire*. URL: <https://www.prnewswire.com/news-releases/pango-mobile-parking-app-coming-to-philadelphia-300087483.html> (date of access: 12.05.2026).

11. 6 Best Parking Management Software for Offices & Facilities awaio. *Awaio*. URL: <https://awaio.com/best-parking-mg-soft/> (date of access: 14.05.2026).

12. Best Parking Management Software. *Capterra*. URL: <https://www.capterra.com/parking-management-software/> (date of access: 14.05.2026).

13. Що таке Typescript і навіщо він потрібен. *IT School FoxmindEd*. URL: <https://foxminded.ua/typescript/> (дата звернення: 20.05.2026).

14. Flutter vs React Native у 2025 році: що обрати для e-commerce додатку? *Brander*. URL: <https://brander.ua/blog/flutter-vs-react-native-u-2025-rotsi-shcho-obraty-dlya-e-commerce-dodatku> (дата звернення: 20.05.2026).

15. React Native vs. Flutter: Everything you need to know. *Contentful*. URL: <https://www.contentful.com/blog/react-native-vs-flutter/> (date of access: 22.05.2026).

16. Firebase: An all-in-one platform for Web and Mobile application development. *Sngular*. URL: <https://www.sngular.com/insights/313/firebase> (date of access: 22.05.2026).

17. What Is NoSQL? NoSQL Databases Explained. *Mongodb*. URL: <https://www.mongodb.com/resources/basics/databases/nosql-explained> (date of access: 28.05.2026).

18. Single Page Application – definition, examples. *Uniquedevs*. URL: <https://uniquedevs.com/en/blog/single-page-application-definition-examples-use-cases/> (date of access: 01.06.2026).

19. Testing React Native Applications: Best practices. *Medium*. URL: <https://medium.com/pen-with-paper/testing-react-native-applications-best-practices-and-frameworks-d0c44dca78e7> (date of access: 03.06.2026).

ДОДАТОК А
Технічне завдання

Вступ

Даний застосунок присвячений автоматизації обліку місць паркування автомобілів.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Розробка програмного забезпечення для обліку місць паркування автомобілів», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 07 квітня 2026 р.

A.2 Основні вимоги до програми, що розробляється

Розробка ПЗ для створення автоматизації обліку місць паркування автомобілів є актуальною й перспективною темою, яка поєднує в собі технічні виклики та суспільну значущість. Такий інструмент не лише спрощує організацію процесів, але й створює умови для підвищення ефективності роботи, зменшення рутинного навантаження, покращення комунікації та прозорості в роботі.

Отже актуальною задачею залишається програмна реалізація застосунку для автоматизації обліку місць паркування автомобілів.

A.2.1 Вимоги до функціональних характеристик

Система реалізовує наступні функціональні вимоги:

- реєстрація та авторизація користувачів;
- надання зручного користувацького інтерфейсу;
- завантаження парко-місць на карті;
- вибір парко-місця;

- виведення інформації про парко-комплекс;
- реалізація форми оплати банківською картою;
- сторінка з qr-code квитком.

A.2.2 Вимоги до інтерфейсу програми

Інтерфейс системи повинен відрізнятися лаконічністю, інтуїтивною зрозумілістю та зручністю для користувачів. Дизайн має бути адаптивним для зручності роботи на будь-якому дисплеї.

A.2.3 Вимоги до надійності

Забезпечення конфіденційності та безпеки даних забезпечується дотримання відповідних правил та рекомендацій щодо захисту даних користувачів та захисту інформації користувача.

A.2.4 Вимоги до складу та параметрів технічних засобів

Основною вимогою до технічних засобів використання програми є наявність пристрою на базі операційної системи Android 5.0 (Lollipop) і вище, наявність вільного простору у пам'яті пристрою, 1 Гб оперативної пам'яті для коректної і плавної роботи та наявність стабільного інтернет-з'єднання.

Для відладки програми використовується емулятор Android Studio на базі операційної системи Android 12.0 (API 32), який також володіє додатковим функціоналом для відстежування критичних помилок у коді.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Розробка ПЗ для створення автоматизації обліку місць паркування автомобілів є актуальною й перспективною темою, яка поєднує в собі технічні виклики та суспільну значущість. Такий інструмент не лише спрощує організацію процесів, але й створює умови для підвищення ефективності роботи, зменшення рутинного навантаження, покращення комунікації та прозорості в роботі.

Отже актуальною задачею залишається програмна реалізація застосунку для автоматизації обліку місць паркування автомобілів.

Б.2 Функціональне призначення

Система реалізовує наступні функціональні вимоги:

- реєстрація та авторизація користувачів;
- надання зручного користувацького інтерфейсу;
- завантаження парко-місць на карті;
- вибір парко-місця;
- виведення інформації про парко-комплекс;
- реалізація форми оплати банківською картою;
- сторінка з qr-code квитком.

Б.3 Опис логічної структури

Застосунок складається з статичних елементів та реактивного компоненту App.tsx, який слугує точкою входу, де відбувається обробка та відображення інформації. Завдяки обробці даних користувач отримує їх у зручному графічному форматі.

Мобільна розробка базується на шести ключових складових-екранах:

- сторінка логіну;

- сторінка реєстрації;
- головна сторінка з картою де знаходяться парко-місця;
- сторінка з інформацією про паркінг;
- сторінка розрахунку за парко-місце;
- сторінка з qr-code квитком.

Для реалізації додатку був створений API, який знаходиться на створеному віддаленому сервері. Найбільш доцільним API для даного застосунку є Firebase.

Google Firebase — це платформа від Google, що дає змогу створювати застосунки для iOS, Android та вебу. Firebase надає набір інструментів для моніторингу аналітики, формування звітів, усунення помилок у програмах, а також для проведення маркетингових та продуктових експериментів.

Система баз даних Firebase пропонує широкий перелік можливостей; у створеному застосунку було інтегровано функціонал автентифікації та бази даних, що працює в режимі реального часу.

Автентифікація Firebase дає змогу розробникам легко впроваджувати надійні схеми верифікації, покращуючи процес входу та адаптації для користувачів. Цей інструмент забезпечує комплексне рішення для ідентифікації, підтримуючи реєстрацію через електронну пошту та пароль, підтвердження через мобільний телефон, а також вхід через акаунти Google, Facebook, GitHub, Twitter та багато інших сервісів.

Для розробки веб-інтерфейсу використано API, розташоване на хмарному сховищі, яке оперує моделлю даних, сфокусованою на документах (база даних типу NoSQL).

Бази даних типу NoSQL спеціально проєктуються під конкретні моделі даних та мають гнучкі схеми, що сприяє розробці сучасних програмних рішень.

NoSQL бази даних здобули значну популярність завдяки простоті розробки, високій функціональності та продуктивності незалежно від масштабу проєкту. Вони дозволяють зберігати прості пари "ключ-значення"

для швидкого кешування, а також вміщувати неструктуровані масиви даних, якими можна оперувати за допомогою мови структурованих запитів SQL.

Документо-орієнтовані NoSQL бази даних користуються великою популярністю у розробників. Ці системи керування базами даних схожі на сховища на основі стовпців, але надають можливість створювати складніші структури.

Документи усувають деякі обмеження, властиві сховищам стовпців. Загалом, вони дозволяють конструювати документи з довільною структурою даних будь-якого рівня складності.

Для створення програмного забезпечення, призначеного для автоматизації обліку паркувальних місць, було обрано архітектуру односторінкового застосунку — SPA (Single Page Application). На рисунку Б.1 приведена загальна структурна схема роботи додатку.

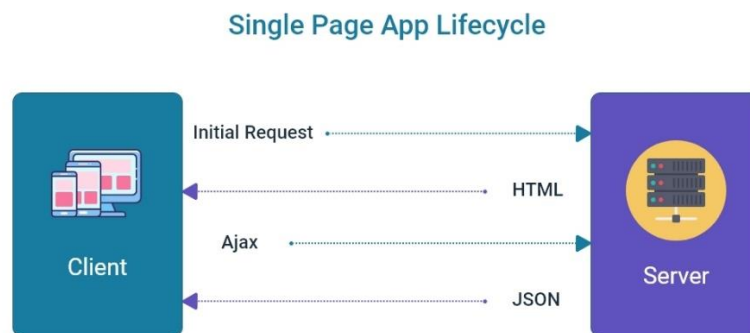


Рисунок Б.1 – Загальна структурна схема роботи додатку [18]

Застосунок типу SPA (Single Page Application) являє собою таку форму веб-додатку, де увесь контент демонструється на одній сторінці, що унеможлиблює її повне оновлення, завдяки чому користувачі отримують більш динамічний та швидкий досвід. Архітектуру SPA формує низка ключових складових: клієнтська частина, серверна частина, інтерфейс REST API, сховище даних (база даних), а також механізми обміну даними між цими елементами.

Ключова розбіжність між класичними веб-додатками та односторінковими застосунками (SPA) полягає у підході до оновлення та рендерингу вмісту інтерфейсу для користувача.

У традиційних додатках, коли користувач виконує певну дію, веб-сторінка обов'язково перезавантажується, що тягне за собою повну реініціалізацію всього візуального оточення. Для відображення змін серверу необхідно передавати щоразу новий HTML-документ із свіжою інформацією, що може спричинити помітні паузи та навіть призвести до втрати даних, які встиг ввести користувач. Натомість, у SPA вся потрібна інформація завантажується на одну сторінку під час першого запуску програми, а всі наступні маніпуляції користувача опрацьовуються без необхідності повного оновлення сторінки.

Б.4 Використані технічні засоби

Основною вимогою до технічних засобів використання програми є наявність пристрою на базі операційної системи Android 5.0 (Lollipop) і вище, наявність вільного простору у пам'яті пристрою, 1 Гб оперативної пам'яті для коректної і плавної роботи та наявність стабільного інтернет-з'єднання.

Для відладки програми використовується емулятор Android Studio на базі операційної системи Android 12.0 (API 32), який також володіє додатковим функціоналом для відстежування критичних помилок у коді.

ДОДАТОК В
Текст програми

1. APP.TSX

```
import React from 'react';
import { Provider as PaperProvider } from 'react-native-paper';
import { globalTheme } from './theme';
import NavigationRoot from './navigation';
```

```
const Application = () => {
  return (
    <PaperProvider theme={globalTheme}>
      <NavigationRoot />
    </PaperProvider>
  );
};
```

```
export default Application;
```

2. THEME.TS

```
import { DefaultTheme } from 'react-native-paper';
```

```
export const globalTheme = {
  ...DefaultTheme,
  colors: {
    ...DefaultTheme.colors,
    primary: '#0a7d0a',
    secondary: '#5b6f54',
    background: '#f9fbf4',
    surface: '#f9fbf4',
    error: '#d32f2f',
  },
};
```

3. NAVIGATION.TSX

```
import React, { useEffect, useState } from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
```

```
import auth from '@react-native-firebase/auth';
```

```
import Login from './screens/Login';
import Register from './screens/Register';
import Home from './screens/Home';
import Details from './screens/Parking';
import Payment from './screens/Payment';
import QR from './screens/QR';
```

```
const Stack = createNativeStackNavigator();
```

```
const NavigationRoot = () => {
  const [currentUser, setCurrentUser] = useState<any>(null);
```

```

const [loading, setLoading] = useState(true);

useEffect(() => {
  const unsubscribe = auth().onAuthStateChanged(user => {
    setCurrentUser(user);
    setLoading(false);
  });

  return unsubscribe;
}, []);

if (loading) return null;

return (
  <NavigationContainer>
    {!currentUser ? (
      <Stack.Navigator screenOptions={{ headerShown: false }}>
        <Stack.Screen name="Login" component={Login} />
        <Stack.Screen name="Register" component={Register} />
      </Stack.Navigator>
    ) : (
      <Stack.Navigator screenOptions={{ headerShown: false }}>
        <Stack.Screen name="Home" component={Home} />
        <Stack.Screen name="Parking" component={Details} />
        <Stack.Screen name="Payment" component={Payment} />
        <Stack.Screen name="QR" component={QR} />
      </Stack.Navigator>
    )}
  </NavigationContainer>
);
};

export default NavigationRoot;

```

4. COMPONENTS/HEADER.TSX

```

import React, { useState } from 'react';
import { AppBar, Menu } from 'react-native-paper';
import auth from '@react-native-firebase/auth';

export const HeaderBar = ({ title, navigation, back }: any) => {
  const [open, setOpen] = useState(false);

  return (
    <AppBar.Header>
      {back ? (
        <AppBar.BackAction onPress={() => navigation.goBack()} />
      ) : (
        <Menu
          visible={open}
          onDismiss={() => setOpen(false)}
          anchor={

```

```

        <AppBar.Action icon="menu" onPress={() => setOpen(true)} />
      }
    >
    <Menu.Item title="Exit" onPress={() => auth().signOut()} />
  </Menu>
)}
<AppBar.Content title={title} />
</AppBar.Header>
);
};

```

5. SCREENS/LOGIN.TSX

```

import React, { useState } from 'react';
import { SafeAreaView } from 'react-native';
import { Button, TextInput, Text } from 'react-native-paper';
import auth from '@react-native-firebase/auth';

const Login = ({ navigation }: any) => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [msg, setMsg] = useState("");

  const handleLogin = async () => {
    try {
      await auth().signInWithEmailAndPassword(email, password);
    } catch {
      setMsg('Invalid login data');
    }
  };

  return (
    <SafeAreaView>
      <TextInput label="Email" onChangeText={setEmail} />
      <TextInput label="Password" secureTextEntry onChangeText={setPassword} />

      <Button onPress={handleLogin}>Sign In</Button>
      <Button onPress={() => navigation.navigate('Register')}>
        Create Account
      </Button>

      <Text>{msg}</Text>
    </SafeAreaView>
  );
};

export default Login;

```

6. SCREENS/REGISTER.TSX

```

import React, { useState } from 'react';
import { SafeAreaView } from 'react-native';

```

```

import { Button, TextInput, Text } from 'react-native-paper';
import auth from '@react-native-firebase/auth';

const Register = () => {
  const [email, setEmail] = useState("");
  const [pass, setPass] = useState("");
  const [repeat, setRepeat] = useState("");
  const [error, setError] = useState("");

  const handleRegister = async () => {
    if (pass !== repeat) {
      setError('Passwords mismatch');
      return;
    }

    try {
      await auth().createUserWithEmailAndPassword(email, pass);
    } catch {
      setError('Registration error');
    }
  };

  return (
    <SafeAreaView>
      <TextInput label="Email" onChangeText={setEmail} />
      <TextInput label="Password" secureTextEntry onChangeText={setPass} />
      <TextInput label="Repeat Password" secureTextEntry onChangeText={setRepeat} />

      <Button onPress={handleRegister}>Register</Button>

      <Text>{error}</Text>
    </SafeAreaView>
  );
};

export default Register;

```

7. SCREENS/HOME.TSX

```

import React, { useEffect } from 'react';
import { SafeAreaView, Image } from 'react-native';
import MapView, { Marker } from 'react-native-maps';
import Geolocation from '@react-native-community/geolocation';
import { HeaderBar } from '../components/Header';

const points = [
  { id: 1, lat: 47.82, lng: 35.17, title: 'Parking A' },
  { id: 2, lat: 47.85, lng: 35.13, title: 'Parking B' },
];

const Home = ({ navigation }: any) => {
  useEffect(() => {

```

```

    Geolocation.getCurrentPosition(pos => {
      console.log(pos.coords);
    });
  }, []);

  return (
    <SafeAreaView style={{ flex: 1 }}>
      <HeaderBar title="Parking Map" />

      <MapView style={{ flex: 1 }} showsUserLocation>
        {points.map(p => (
          <Marker
            key={p.id}
            coordinate={{ latitude: p.lat, longitude: p.lng }}
            onPress={() => navigation.navigate('Parking', p)}
          >
            <Image
              source={require('../assets/marker.png')}
              style={{ width: 30, height: 30 }}
            />
          </Marker>
        ))}
      </MapView>
    </SafeAreaView>
  );
};

export default Home;

```

8. SCREENS/PARKING.TSX

```

import React from 'react';
import { SafeAreaView, Image, View } from 'react-native';
import { Text, Button } from 'react-native-paper';
import { HeaderBar } from '../components/Header';

const Parking = ({ route, navigation }: any) => {
  const { title } = route.params;

  return (
    <SafeAreaView>
      <HeaderBar title={title} back navigation={navigation} />

      <Image
        source={{ uri: 'https://via.placeholder.com/400' }}
        style={{ height: 250 }}
      />

      <View>
        <Text>{title}</Text>
        <Text>Price: 2$</Text>
      </View>
    </SafeAreaView>
  );
};

```

```

        <Button onPress={() => navigation.navigate('Payment')}>
          Pay
        </Button>
      </View>
    </SafeAreaView>
  );
};

```

```
export default Parking;
```

9. SCREENS/PAYMENT.TSX

```

import React, { useState } from 'react';
import { SafeAreaView } from 'react-native';
import { Button, TextInput, Text } from 'react-native-paper';
import { HeaderBar } from '../components/Header';

const Payment = ({ navigation }: any) => {
  const [card, setCard] = useState("");
  const [error, setError] = useState("");

  const handlePay = () => {
    if (card.length < 16) {
      setError('Invalid card');
      return;
    }
    navigation.navigate('QR');
  };

  return (
    <SafeAreaView>
      <HeaderBar title="Payment" />

      <TextInput label="Card Number" onChangeText={setCard} />

      <Button onPress={handlePay}>Confirm</Button>

      <Text>{error}</Text>
    </SafeAreaView>
  );
};

export default Payment;

```

10. SCREENS/QR.TSX

```

import React, { useState } from 'react';
import { SafeAreaView } from 'react-native';
import QRCode from 'react-qr-code';
import { Button, Text } from 'react-native-paper';

const QR = ({ navigation }: any) => {

```

```
const [code] = useState(  
  Math.floor(1000000000 + Math.random() * 9000000000)  
);  
  
return (  
  <SafeAreaView style={{ alignItems: 'center', justifyContent: 'center', flex: 1 }}>  
    <Text>Order #{code}</Text>  
  
    <QRCode value={code.toString()} />  
  
    <Button onPress={() => navigation.navigate('Home')}>  
      Close  
    </Button>  
  </SafeAreaView>  
);  
};  
  
export default QR;
```

ДОДАТОК Г
Слайди презентації



Міністерство освіти та науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Розробка програмного забезпечення для обліку місць паркування автомобілів

Виконав: Максим ВАСИЛЬЄВ

ст. групи КНТ-122

Керівник: Михайло ПОЛЯКОВ

д.т.н., професор

Рисунок Г.1 – Слайд 1



Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес розробки програмного забезпечення для автоматизації обліку місць паркування автомобілів.

Предмет дослідження – програмні засоби для автоматизації обліку місць паркування автомобілів.

Мета роботи – збільшення швидкості обробки даних про зайнятість паркувальних майданчиків та зменшення часу пошуку вільного місця за рахунок автоматизації обліку

2

Рисунок Г.2 – Слайд 2

Постановка завдання

Розробка ПЗ для створення автоматизації обліку місць паркування автомобілів є актуальною й перспективною темою, яка поєднує в собі технічні виклики та суспільну значущість. Такий інструмент не лише спрощує організацію процесів, але й створює умови для підвищення ефективності роботи, зменшення рутинного навантаження, покращення комунікації та прозорості в роботі.

Отже актуальною задачею залишається програмна реалізація застосунку для автоматизації обліку місць паркування автомобілів.

3

Рисунок Г.3 – Слайд 3

Порівняння ПЗ для створення тестів

Хар-ка	Awaio	Parkade	Ronspot	ParkingPass	ParkHub	Parkable
Особливість	Спільне використання та управління ресурсами на гібридних робочих місцях.	Підходить для багатоквартирних будинків.	Безкоштовний пробний період.	Управління громадськими парковками.	Для університетів та розваг, або спортивних заходів.	Призначений для комерційних будівель та робочих місць.
Відмінність від аналогів	Динамічний обмін для обмеженої кількості місць.	Динамічне ціноутворення Цілодобова підтримка.	Розподіл простору за допомогою системи бронювання.	Цифровий дозвіл на паркування.	Інтегрується з основними платформами продажу квитків.	Алгоритм Fairchance, який розподіляє місця між користувачами.
Вимоги до обладнання	Немає	Так	Так	Так	Так	Так
Ціноутворення	Модульне та гнучке.	Недоступно для громадськості.	Наявні безкоштовні та платні тарифні плани	Недоступно для громадськості	Недоступно для громадськості.	Недоступно для громадськості
Аналіз та звітність	Так	Так	Так	Так	Так	Так

4

Рисунок Г.4 – Слайд 4

Вибір мови програмування

Параметр	TypeScript	JavaScript
Типізація	Підтримка статичної типізації	Динамічна типізація
Помилки	Помилки виявляються на етапі компіляції	Помилки виявляються під час виконання
Розширення	Надмножина JavaScript, що забезпечує зворотну сумісність	Базова мова програмування
Інструменти	Хороша підтримка IDE та інших інструментів розробки	Великий вибір інструментів і бібліотек
Сумісність	Можна використовувати JavaScript-код	Можливе використання TypeScript у JavaScript-проектах
Розробка	Дає змогу створювати більш надійний і підтримуваний код	Швидке розроблення та прототипування
Сучасні можливості	Підтримка сучасних стандартів JavaScript	Необмежений доступ до можливостей JS

5

Рисунок Г.5 – Слайд 5

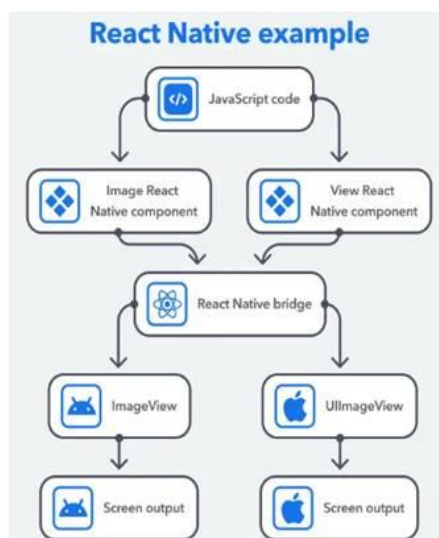
Вибір фреймворку для створення ПЗ

Функція фреймворку	React Native	Flutter
Кросплатформний	Дозволяє розробникам написати логіку JavaScript один раз та створювати її як для Android, так і для iOS, використовуючи нативні компоненти для узгодженого рендерингу інтерфейсу користувача.	Пропонує справжню кросплатформну розробку, рендерячи все за допомогою власного движка на базі Skia, забезпечуючи майже нативну продуктивність.
Мова програмування	Використовує JavaScript, широко відомий та використовуваний, що полегшує перехід для веб-розробників.	Використовує Dart, менш поширену, але спеціально розроблену мову з такими функціями, як AOT та JIT-компіляція для високої продуктивності.
Рендеринг інтерфейсу користувача	Використовує нативні компоненти платформи, забезпечуючи, щоб програми виглядали та відображалися як нативні користувацькі інтерфейси на різних платформах, але програми, але може вимагати об'єднання розширених функцій.	Використовує власні віджети, що відображаються через Skia, забезпечуючи узгоджений інтерфейс користувацької роботи, подібної до нативної.
Продуктивність	Залежить від JS, який не компілюється в машинний код, що іноді може призвести до збільшення продуктивності.	Компілює код Dart у власний машинний код, забезпечуючи швидшу продуктивність.
Екосистема та бібліотеки	Велика екосистема сторонніх бібліотек завдяки популярності JavaScript, хоча якість бібліотек може відрізнятися.	Менша екосистема, з офіційною підтримкою багатьох поширених випадків використання та сильним акцентом на якості.
Підтримка громади	Підтримується Meta, з великою спільнотою розробників та учасників по всьому світу.	Підтримується Google, має меншу, але швидкозростаючу та дуже активну спільноту.

6

Рисунок Г.6 – Слайд 6

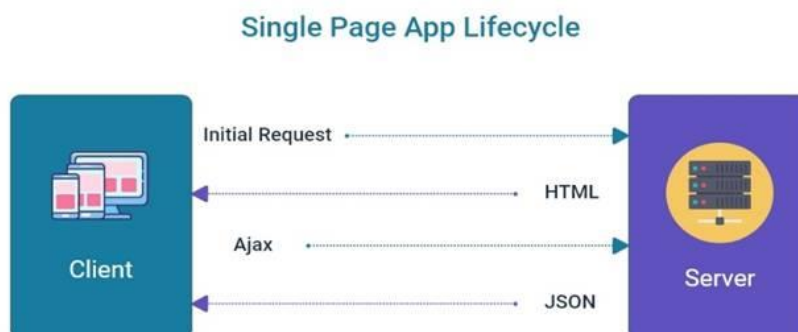
Схема роботи додатку з використанням React Native



7

Рисунок Г.7 – Слайд 7

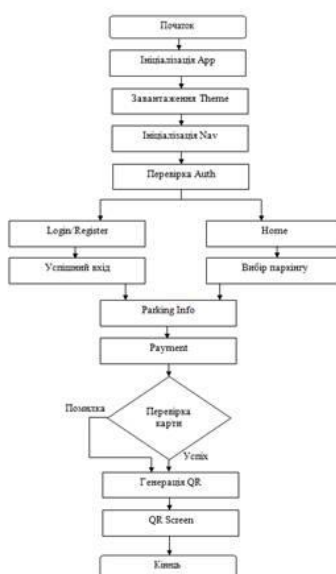
Загальна структурна схема системи



8

Рисунок Г.8 – Слайд 8

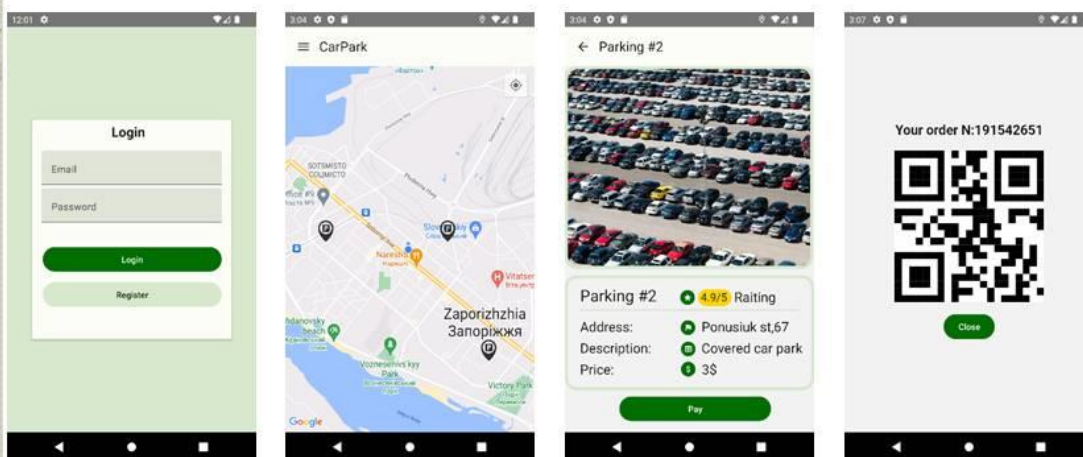
Основний алгоритм роботи програмного забезпечення



9

Рисунок Г.9 – Слайд 9

Інтерфейс застосунку



10

Рисунок Г.10 – Слайд 10

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було виконано програмну реалізацію застосунку для автоматизації обліку місць паркування автомобілів.

Для цього було виконано ряд завдань:

- розглянуто основні поняття предметної області;
- проведено аналіз способів оплати за користування паркувальним майданчиком;
- виконано опис систем для управління паркувальним майданчиком;
- проведено порівняння існуючого програмного забезпечення для управління паркуванням автомобілів;
- виконано вибір мови програмування та фреймворку для створення програмного забезпечення для автоматизації обліку місць паркування автомобілів;
- сформовано структур програмного забезпечення;
- створено алгоритм функціонування системи;
- виконано проектування інтерфейсу взаємодії користувача з ПЗ;
- розроблено програмне забезпечення для автоматизації обліку місць паркування автомобілів;
- описано процедуру виконання програмного забезпечення;
- проведено тестування додатку.

В результаті розробки реалізовано систему, що відповідає всім вимогам визначеним на початку розробки у вигляді функціональних вимог.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

11

Рисунок Г.11 – Слайд 11