Додаток А

Звіт з результатами перевірки на унікальність тексту в базі ХНУРЕ

StrikePlagiarism.com

Дата звіту 6/6/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_Б_ПІ_ПЗПІ-21-10_Радько_М_М_скорочений
Автор
Радько Марк Максимович
Науковий керівник / Експерт
Євген Кардаш
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

0.61% 0.61% КП 1

0.41% 0.41% КЦ

25
Довжина фрази для коефіцієнта подібності 2

15053
Кількість слів

126138
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		0
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)		6

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка платформи для проведення голосувань з елементами блокчейн-технологій [redacted] 5/21/2025 Volodymyr Vynnychenko Central Ukrainian State Pedagogical University (кафедра інформатики, програмування, штучного інтелекту та технологічної освіти)	19 0.13 %
2	https://mskelton.dev/bytes/encrypting-data-with-node-js	18 0.12 %
3	https://github.com/NurePershynaAnastasiia/apzkr-pzpi-21-1-pershyna-anastasiia/blob/main/apzkr-pzpi-21-1-pershyna-anastasiia.txt	13 0.09 %

4	Бакалаврська_робота_Янок_М.В._н.кер.Переверзєв А.М. [redacted] 5/6/2025 European University (European University)	11 0.07 %
5	https://mskelton.dev/bytes/encrypting-data-with-node-js	10 0.07 %
6	https://github.com/NurePershynaAnastasiia/apzkr-pzpi-21-1-pershyna-anastasiia/blob/main/apzkr-pzpi-21-1-pershyna-anastasiia.txt	10 0.07 %
7	https://gist.github.com/kbhokray/34a76061fb8696b3e37d7de740271df0	6 0.04 %
8	https://mskelton.dev/bytes/encrypting-data-with-node-js	5 0.03 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з програми обміну базами даних (0.20 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка платформи для проведення голосувань з елементами блокчейн-технологій [redacted] 5/21/2025 Volodymyr Vynnychenko Central Ukrainian State Pedagogical University (кафедра інформатики, програмування, штучного інтелекту та технологічної освіти)	19 (1) 0.13 %
2	Бакалаврська_робота_Янок_М.В._н.кер.Переверзєв А.М. [redacted] 5/6/2025 European University (European University)	11 (1) 0.07 %

з Інтернету (0.41 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://mskelton.dev/bytes/encrypting-data-with-node-js	33 (3) 0.22 %
2	https://github.com/NurePershynaAnastasiia/apzkr-pzpi-21-1-pershyna-anastasiia/blob/main/apzkr-pzpi-21-1-pershyna-anastasiia.txt	23 (2) 0.15 %
3	https://gist.github.com/kbhokray/34a76061fb8696b3e37d7de740271df0	6 (1) 0.04 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

Додаток Б

Слайди презентації

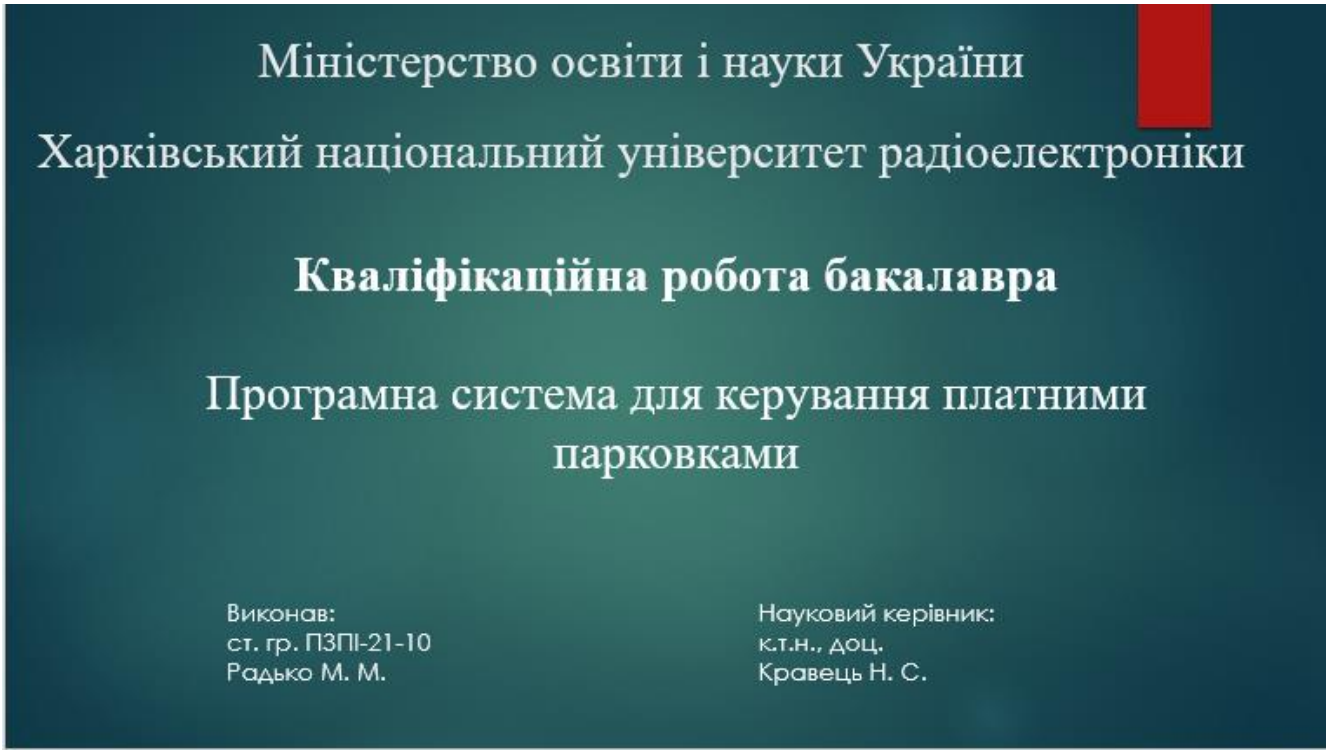

Рисунок Б.1 – Слайд №1 (рисунок виконаний самостійно)

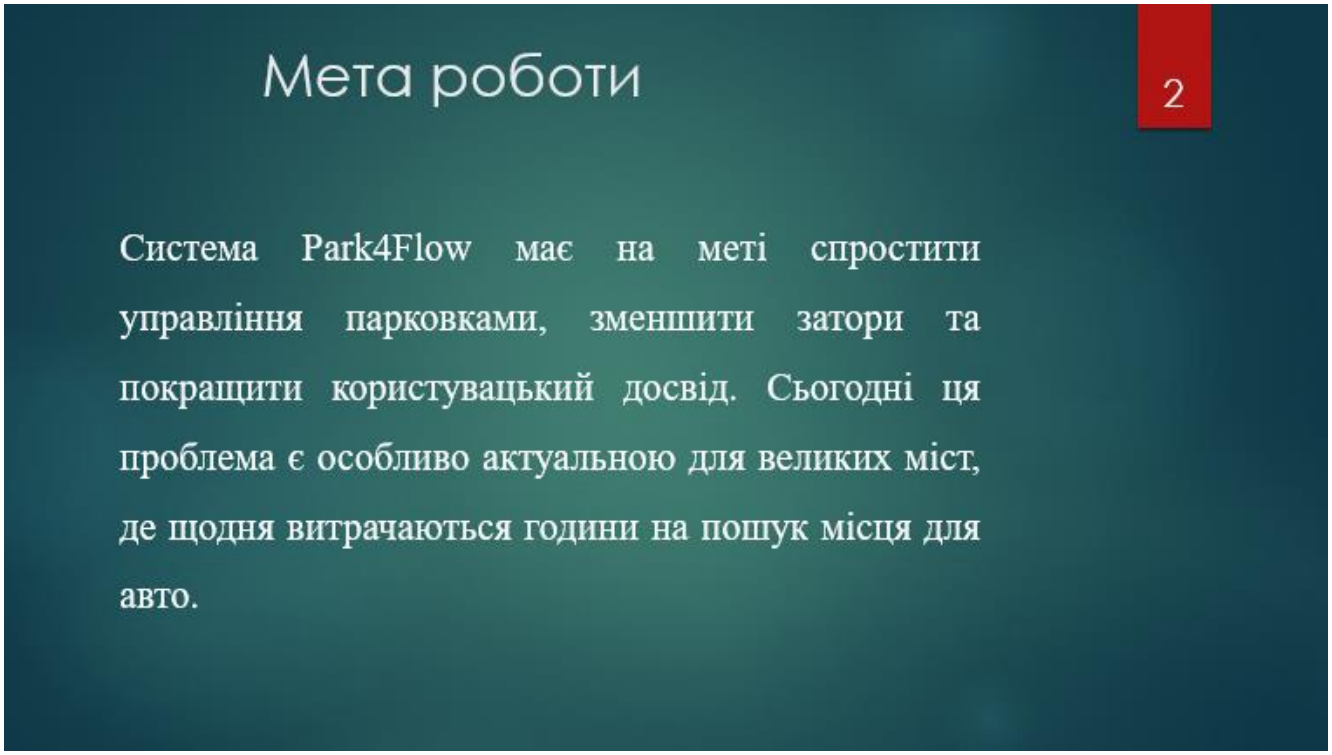

Рисунок Б.2 – Слайд №2 (рисунок виконаний самостійно)

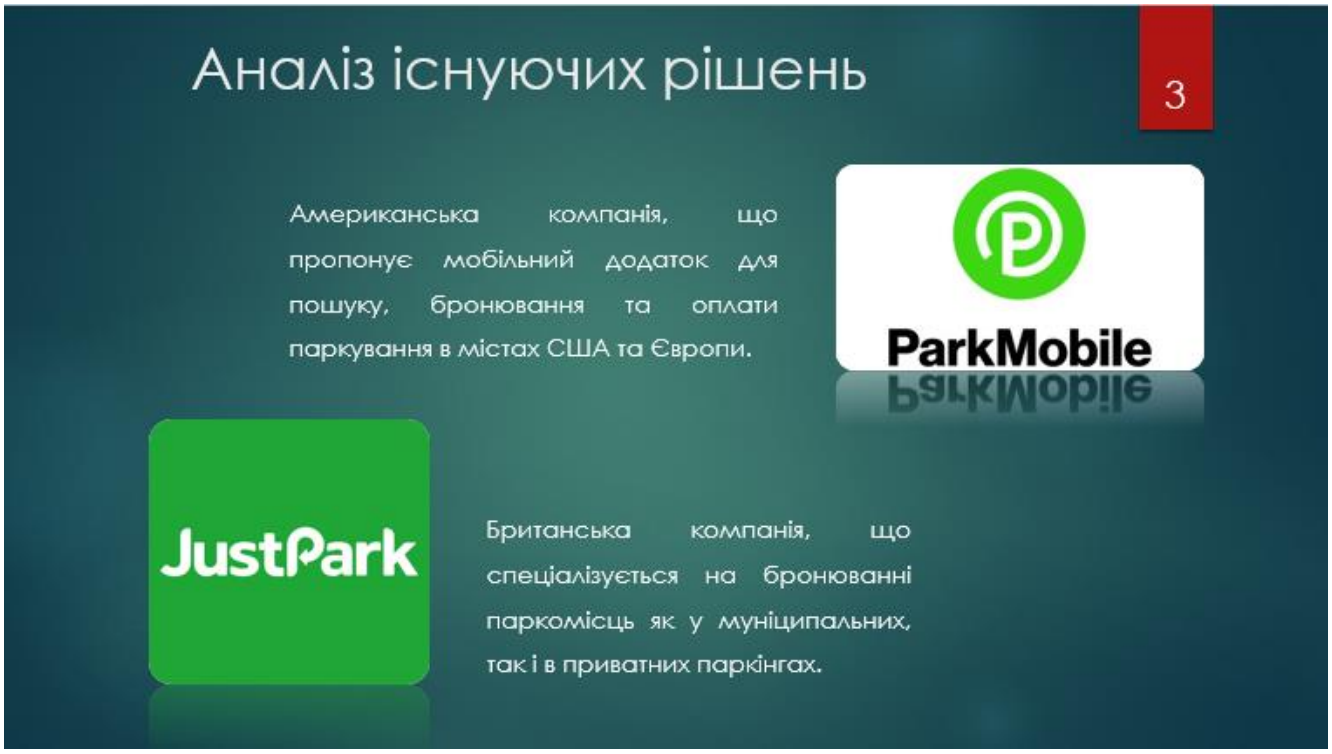

Рисунок Б.3 - Слайд №3 (рисунок виконаний самостійно)

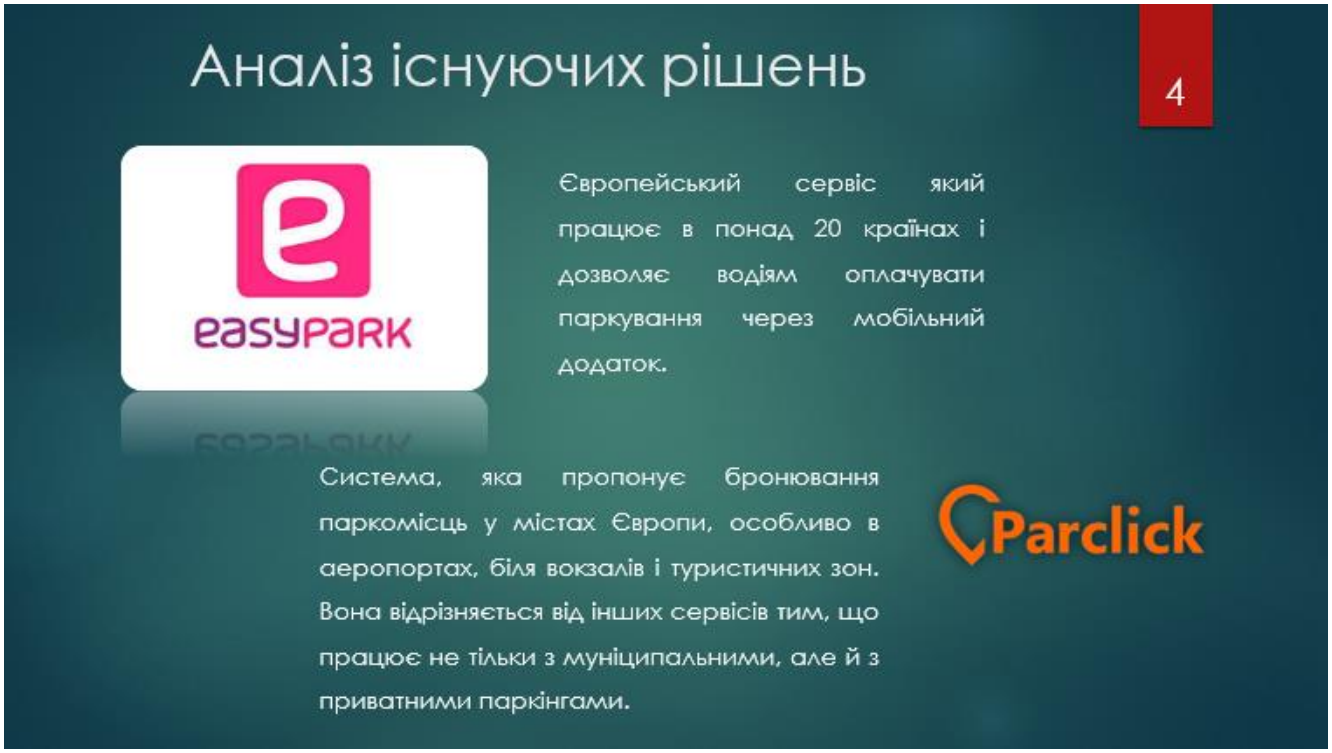

Рисунок Б.4 – Слайд №4 (рисунок виконаний самостійно)

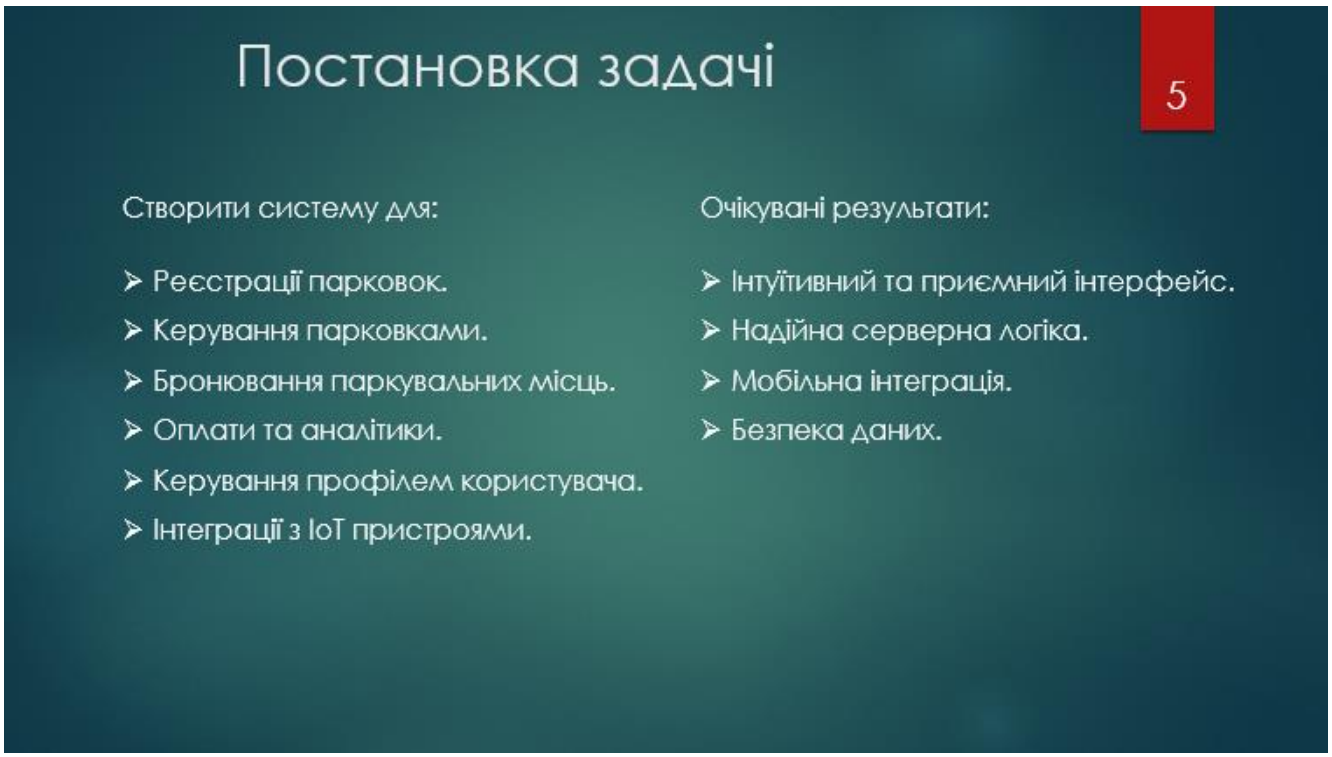

Рисунок Б.5 – Слайд №5 (рисунок виконаний самостійно)

Рисунок Б.6 – Слайд №6 (рисунок виконаний самостійно)

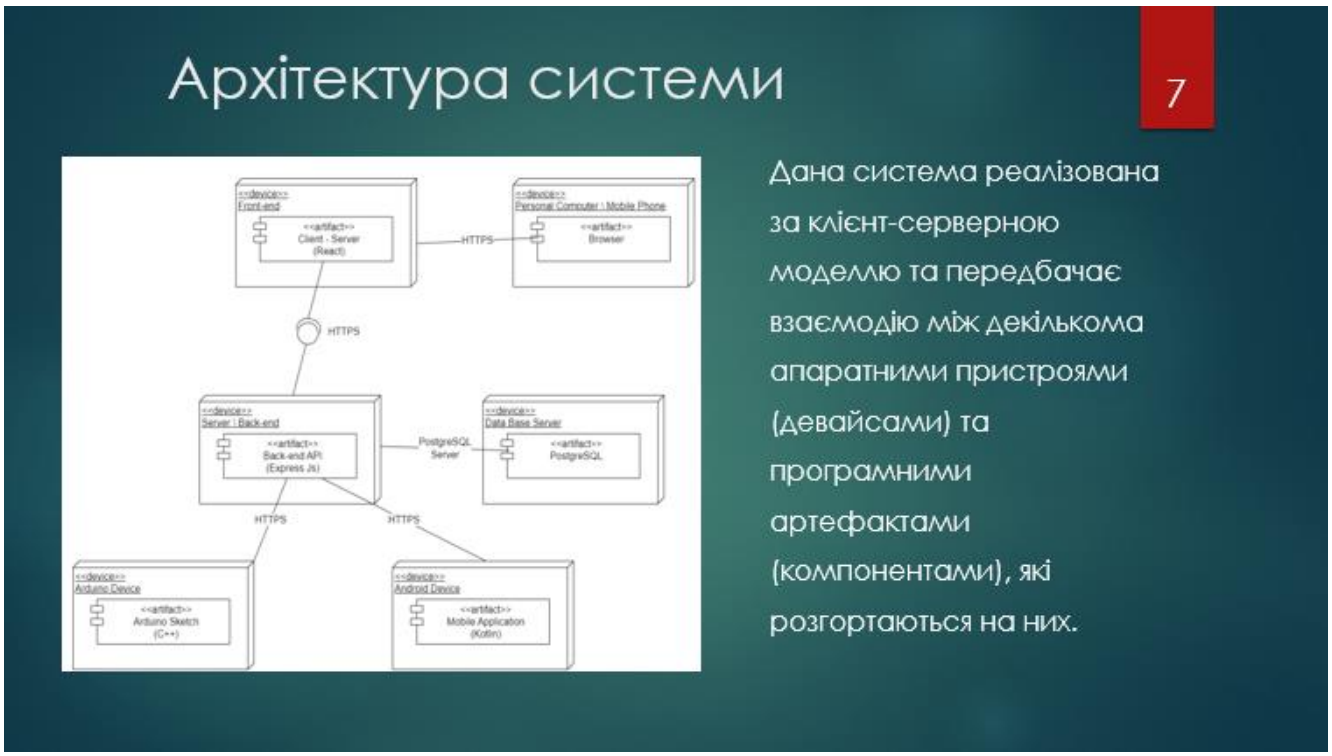

Рисунок Б.7 – Слайд №7 (рисунок виконаний самостійно)

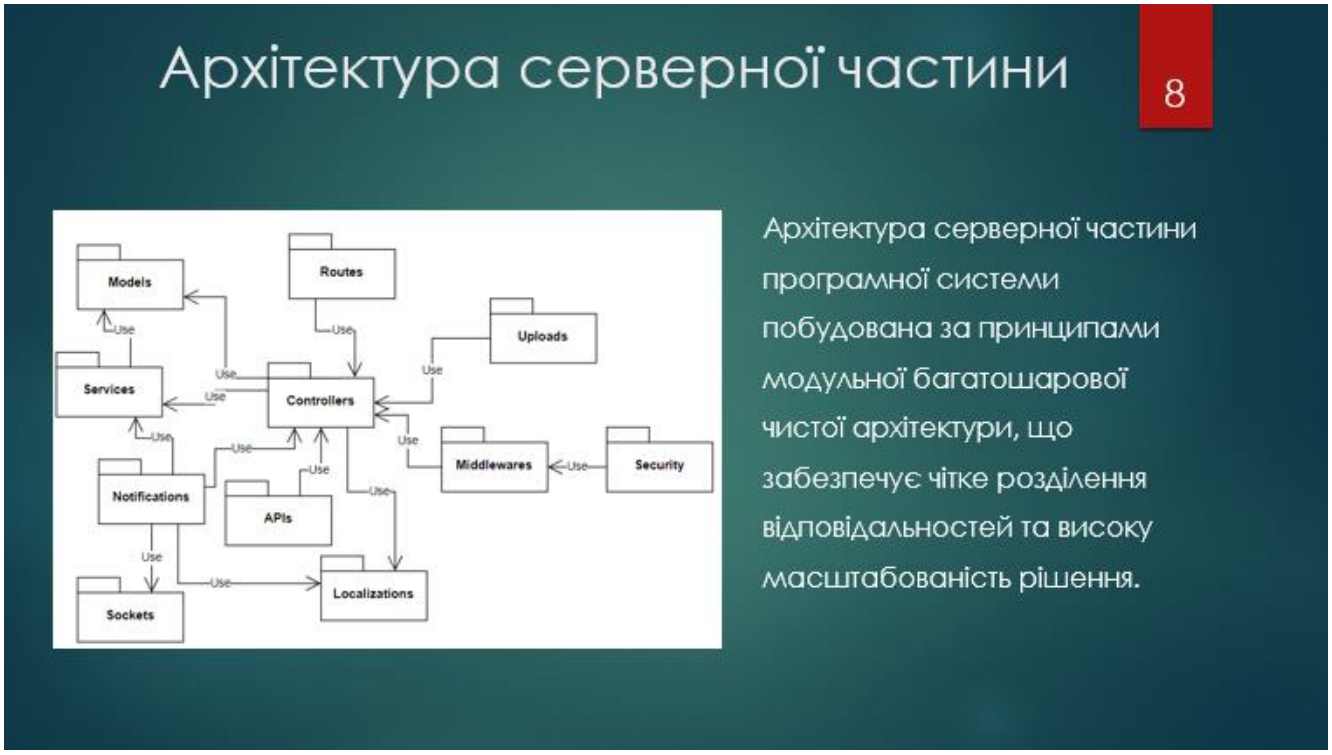

Рисунок Б.8 – Слайд №8 (рисунок виконаний самостійно)

Рисунок Б.9 – Слайд №9 (рисунок виконаний самостійно)

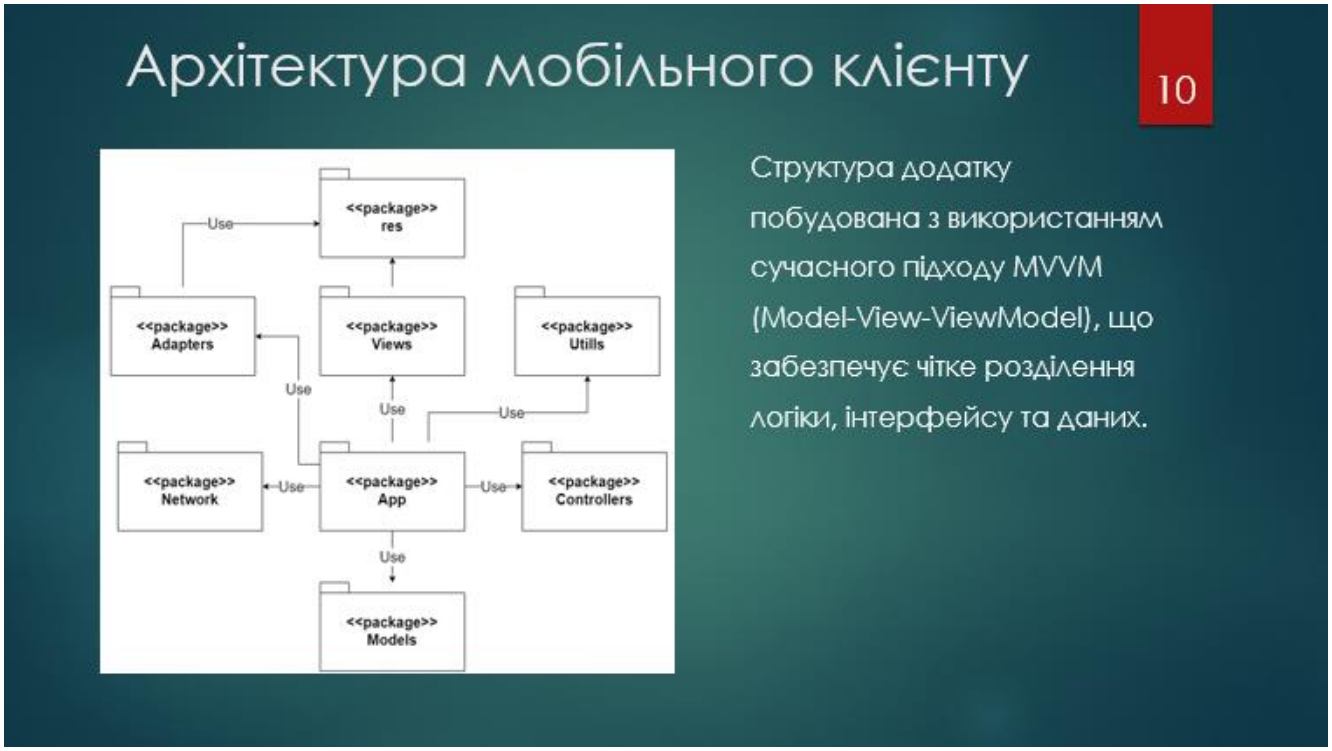

Рисунок Б.10 – Слайд №10 (рисунок виконаний самостійно)

Рисунок Б.11 – Слайд №11 (рисунок виконаний самостійно)

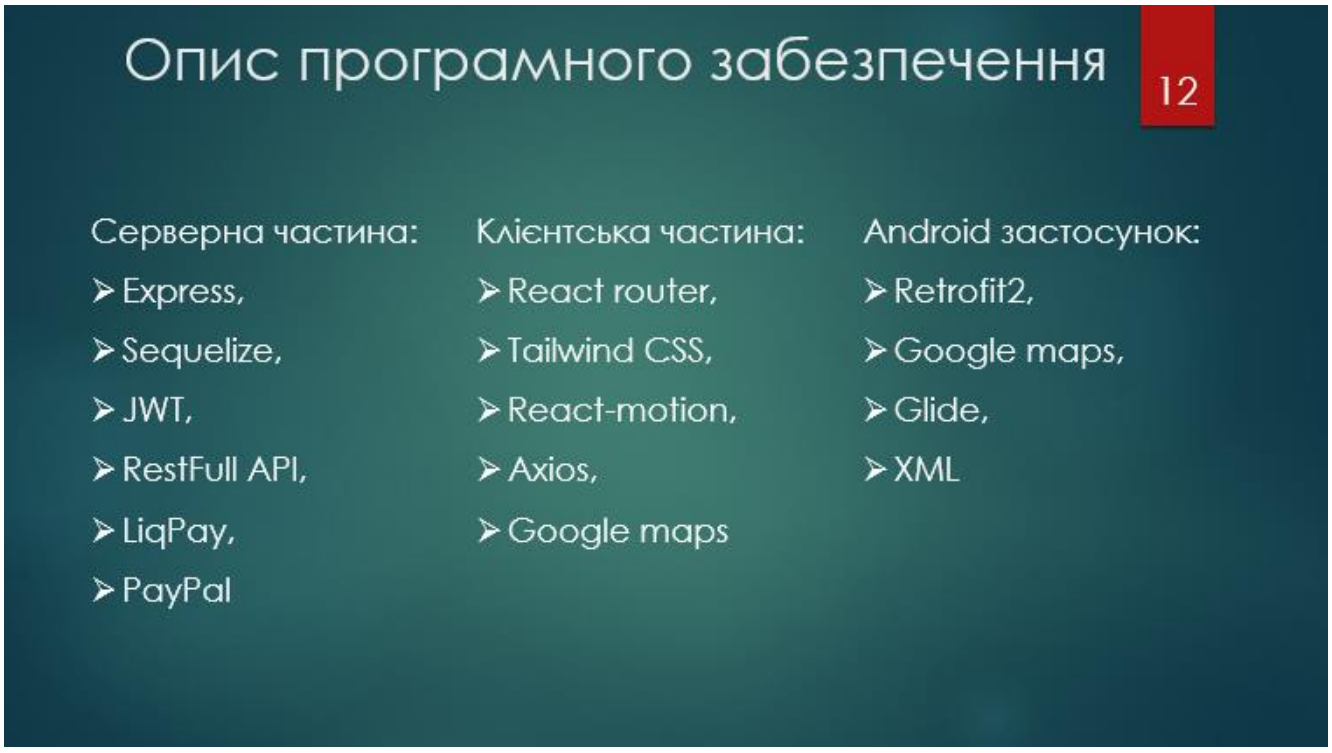

Рисунок Б.12 – Слайд №12 (рисунок виконаний самостійно)

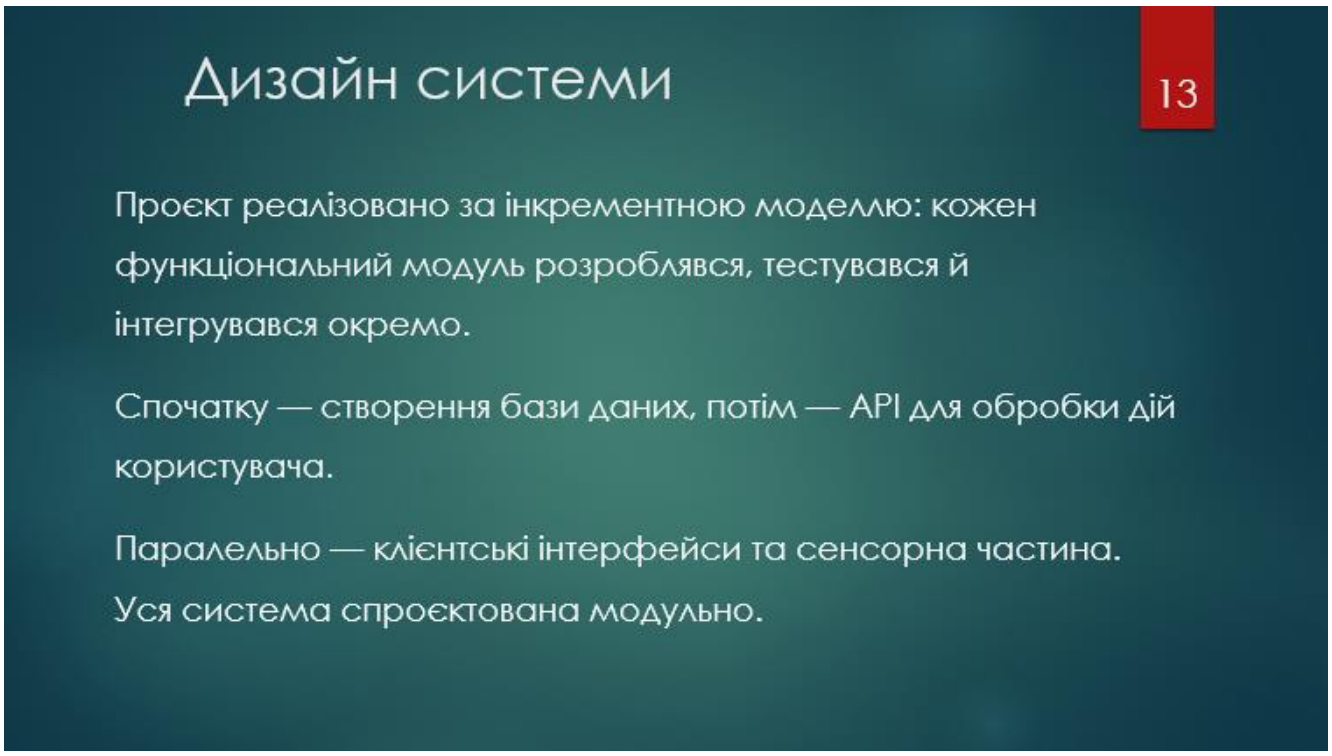

Рисунок Б.13 – Слайд №13 (рисунок виконаний самостійно)

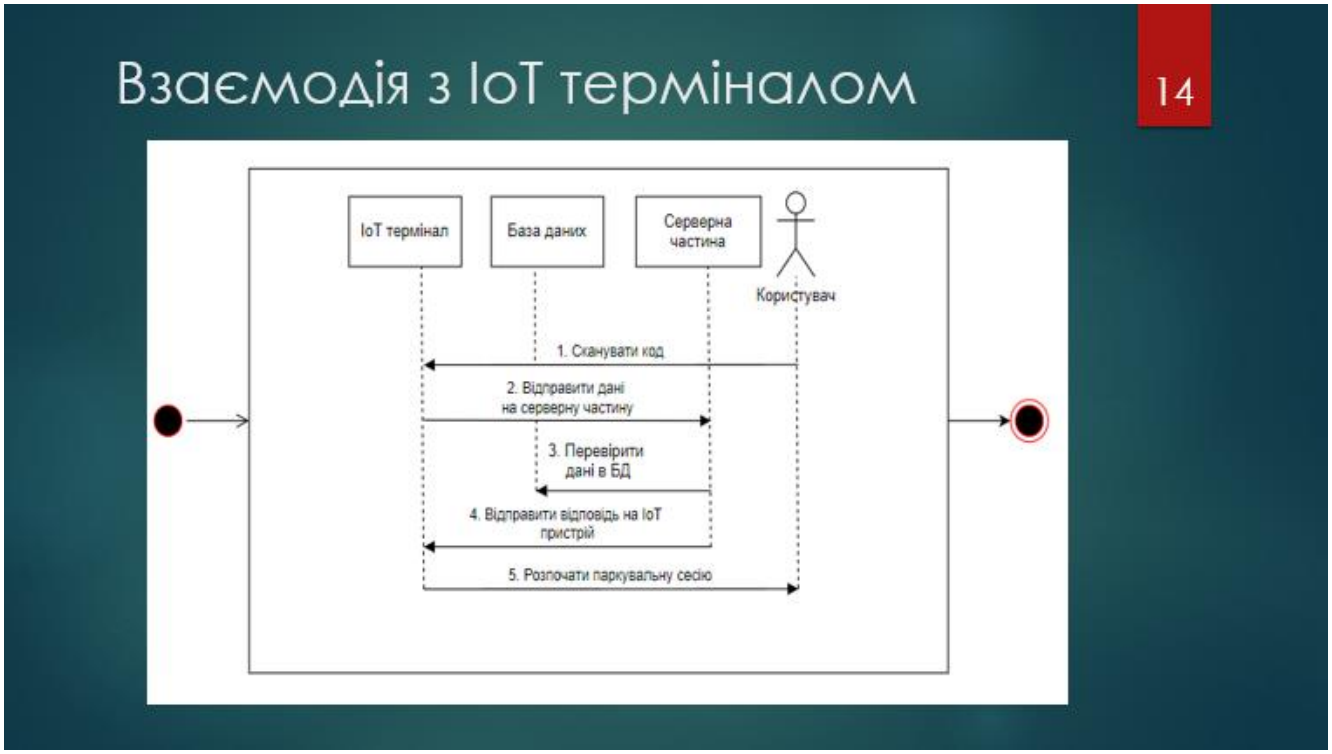

Рисунок Б.14 – Слайд №14 (рисунок виконаний самостійно)

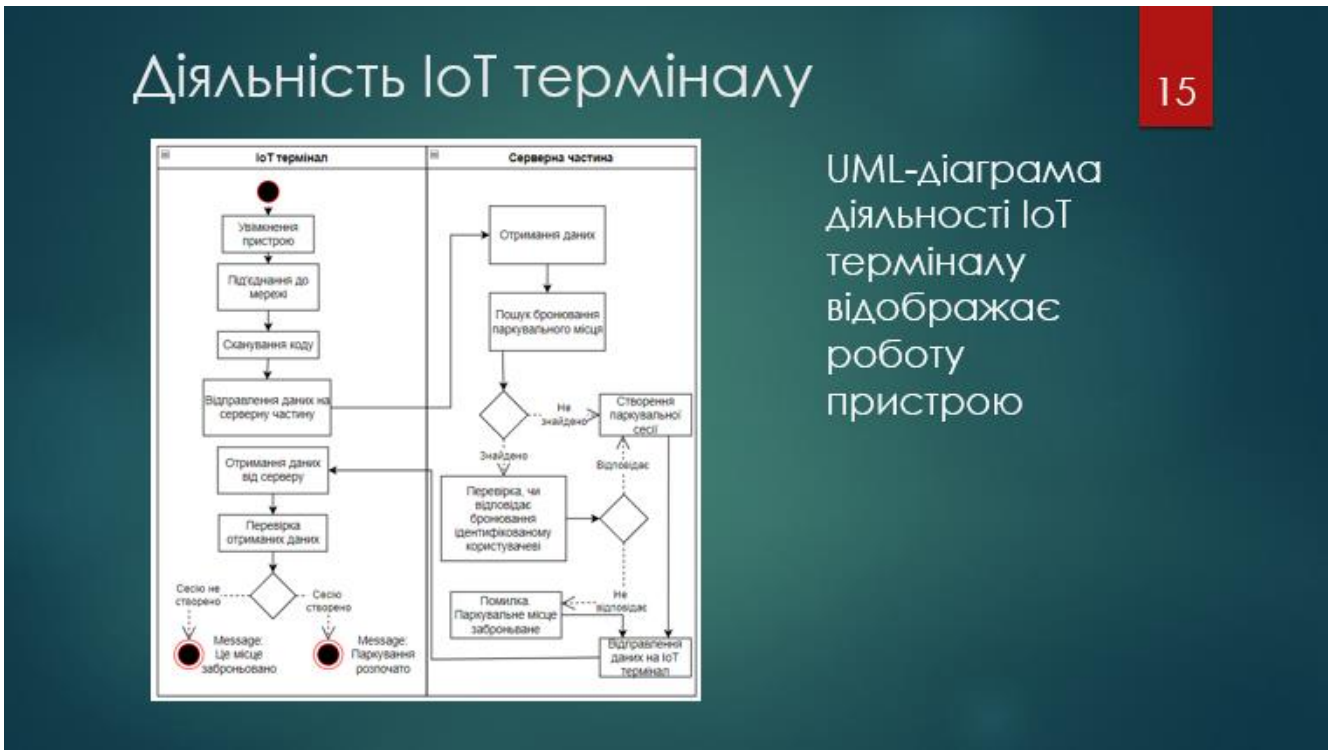

Рисунок Б.15 – Слайд №15 (рисунок виконаний самостійно)

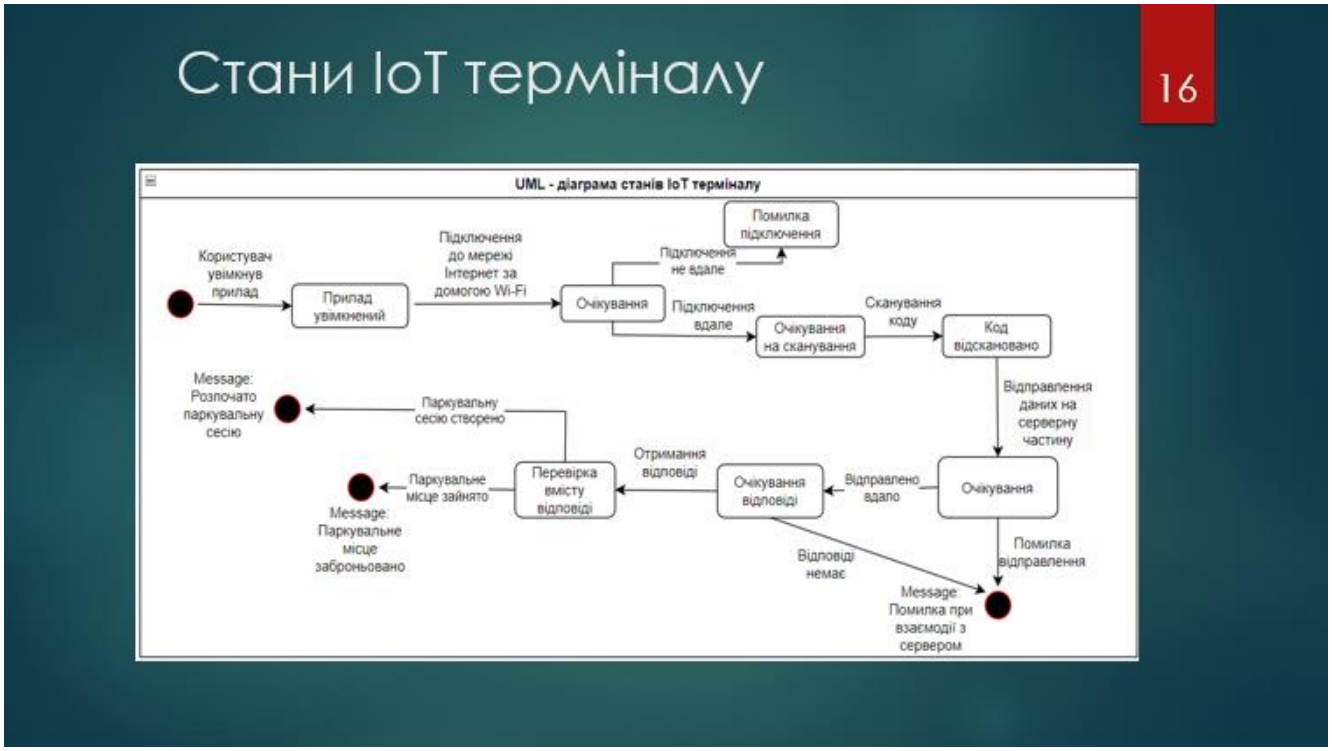

Рисунок Б.16 – Слайд №16 (рисунок виконаний самостійно)

Рисунок Б.17 – Слайд №17 (рисунок виконаний самостійно)

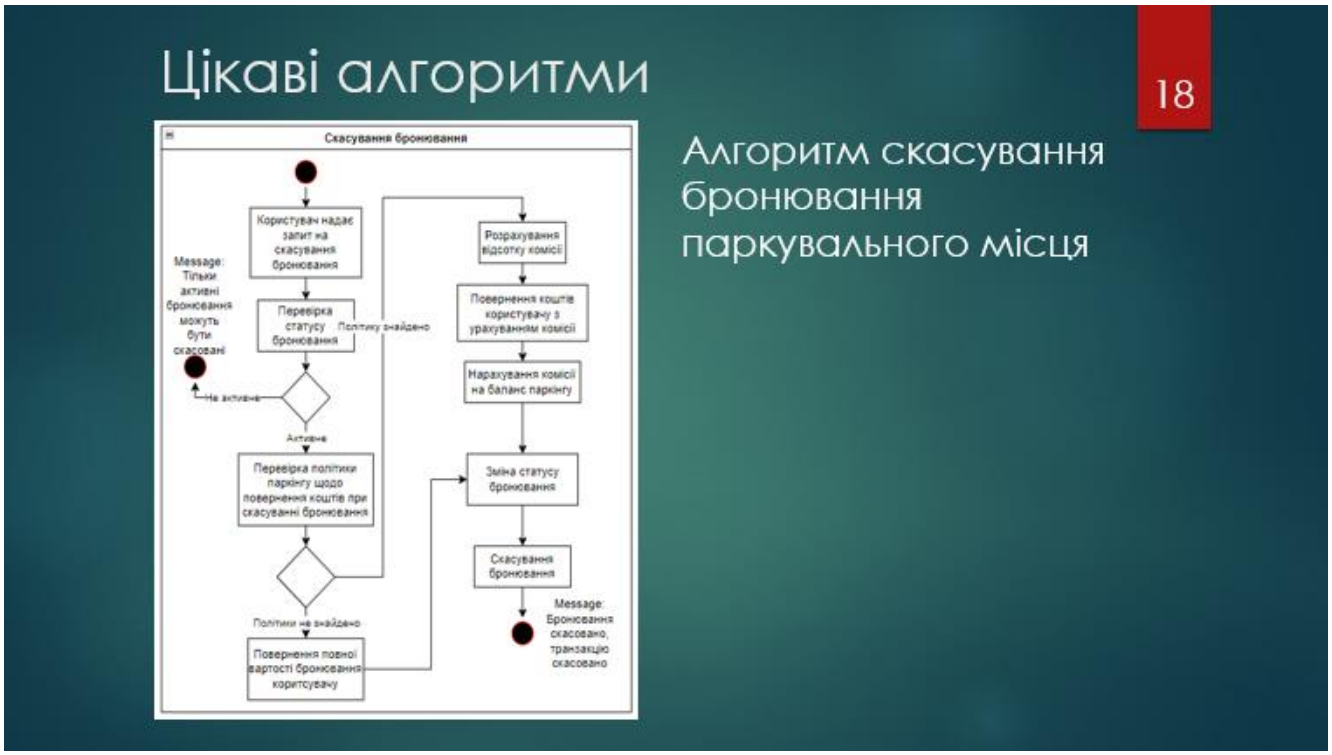

Рисунок Б.18 – Слайд №18 (рисунок виконаний самостійно)

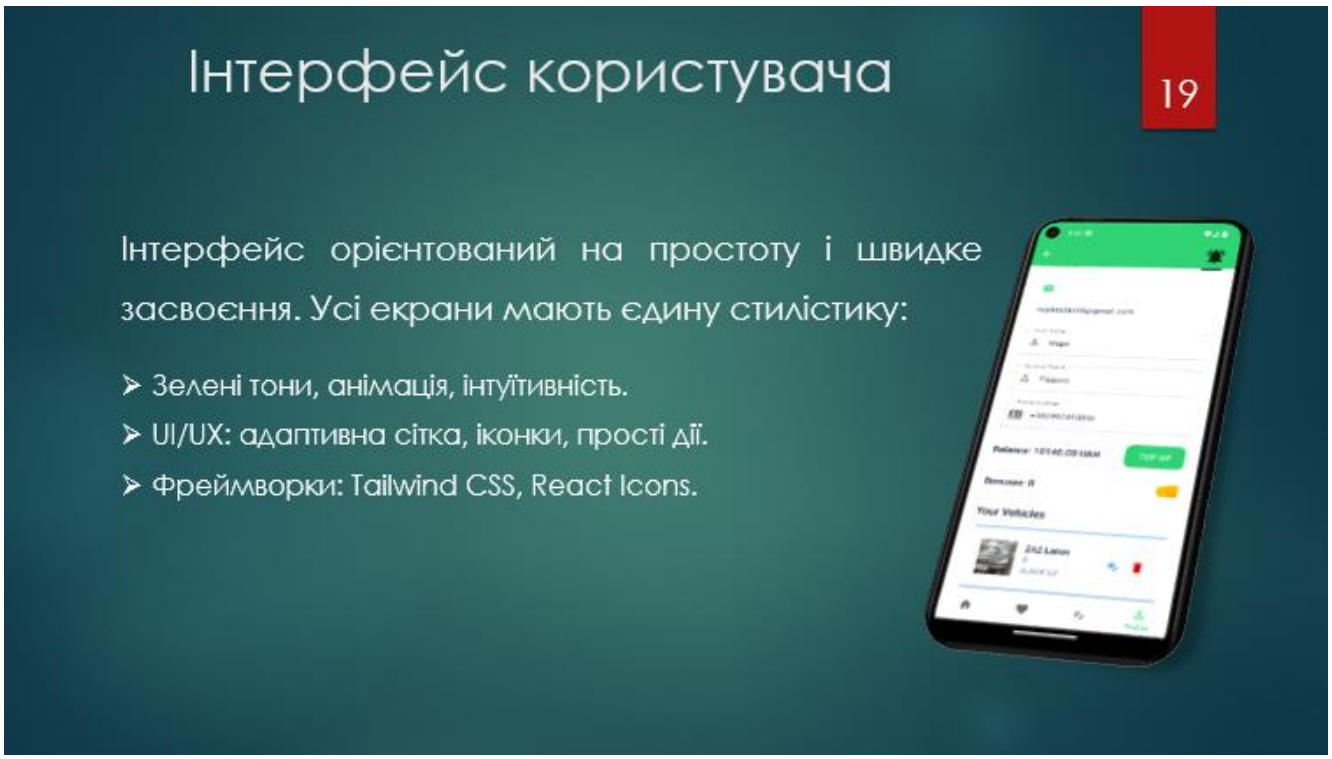

Рисунок Б.19 – Слайд №19 (рисунок виконаний самостійно)

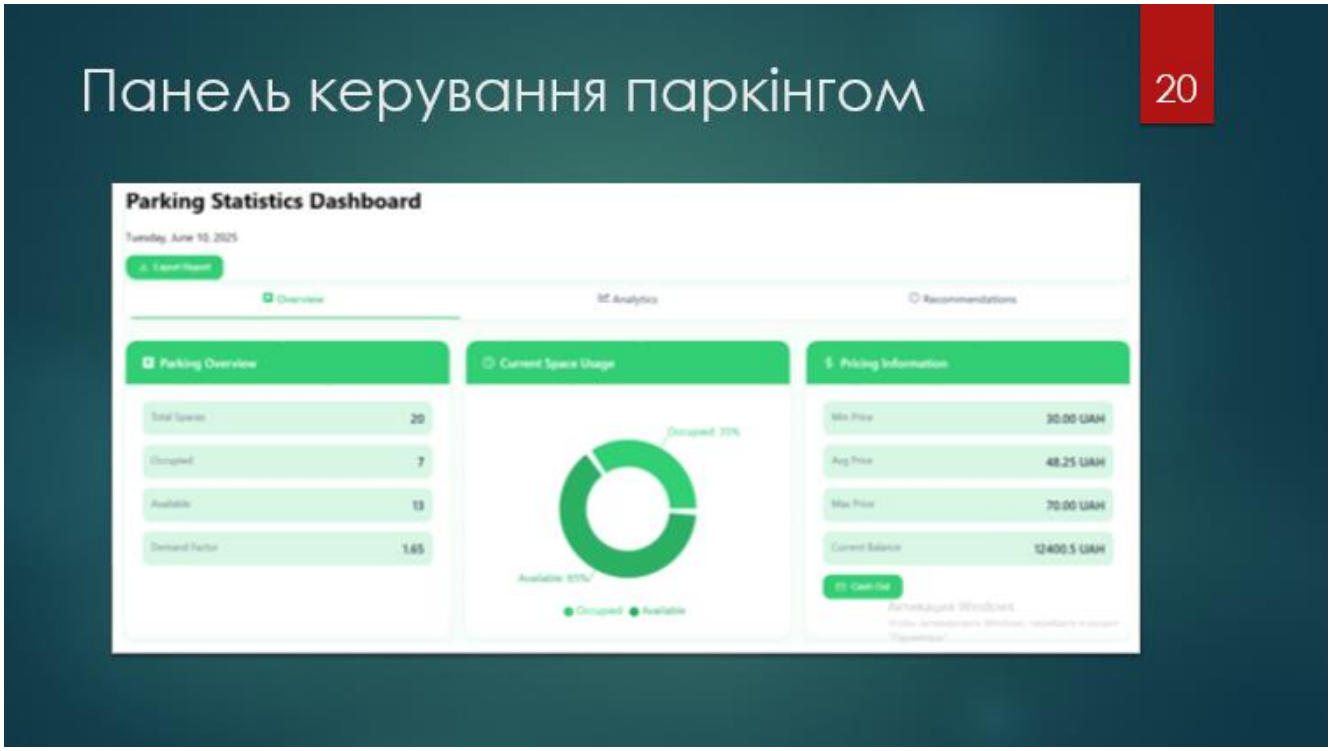

Рисунок Б.20 – Слайд №20 (рисунок виконаний самостійно)

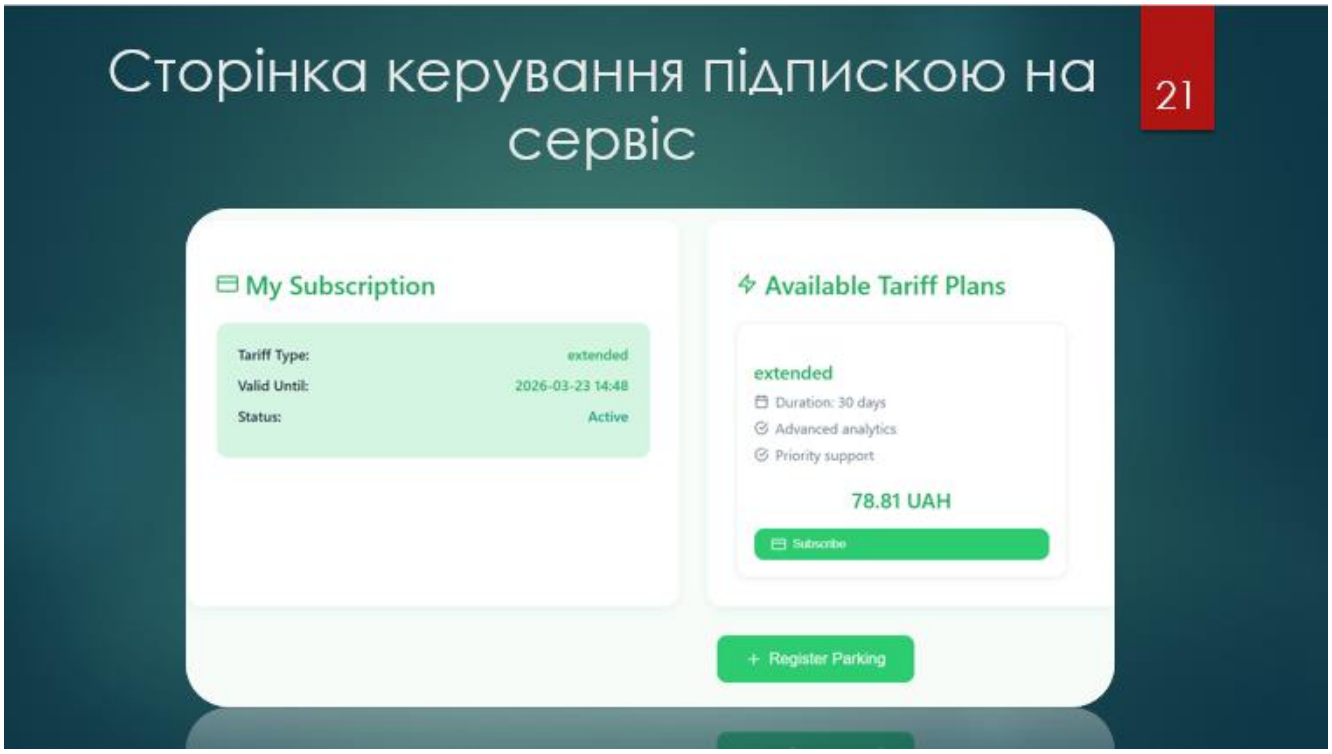

Рисунок Б.21 – Слайд №21 (рисунок виконаний самостійно)

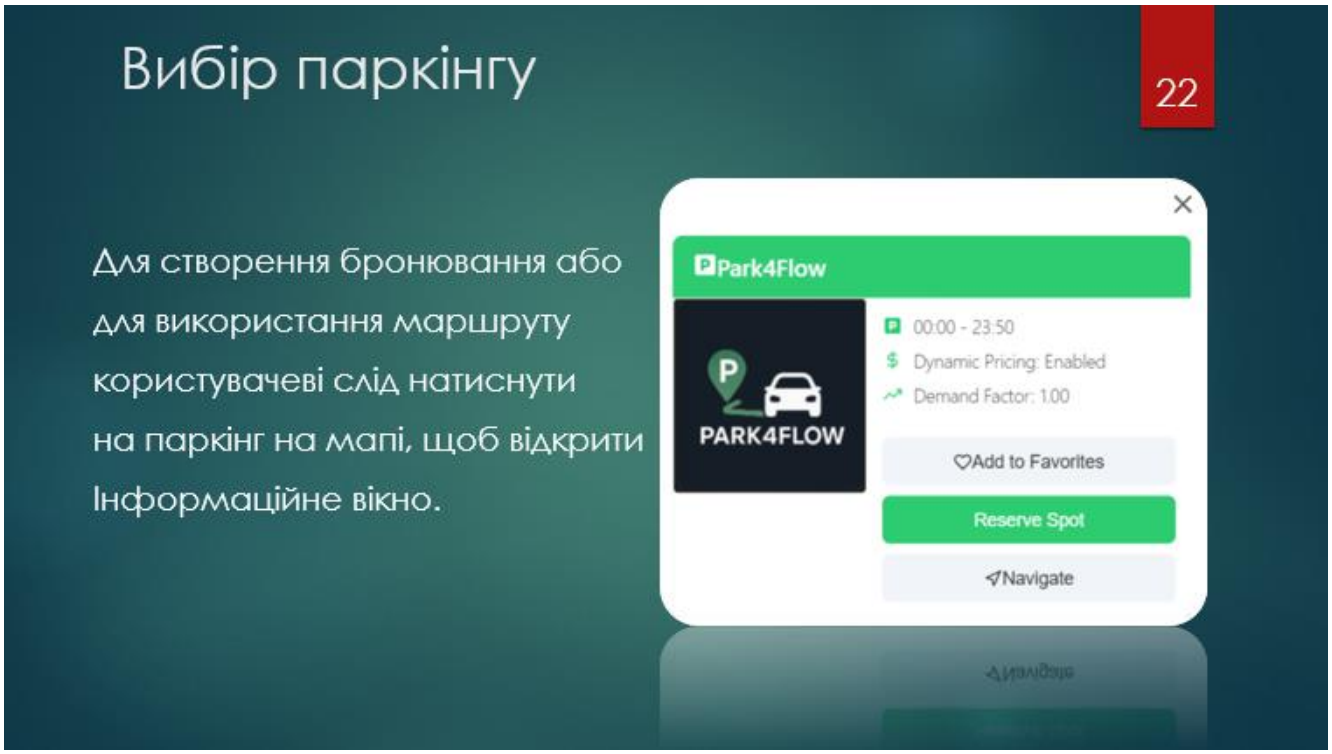

Рисунок Б.22 – Слайд №22 (рисунок виконаний самостійно)

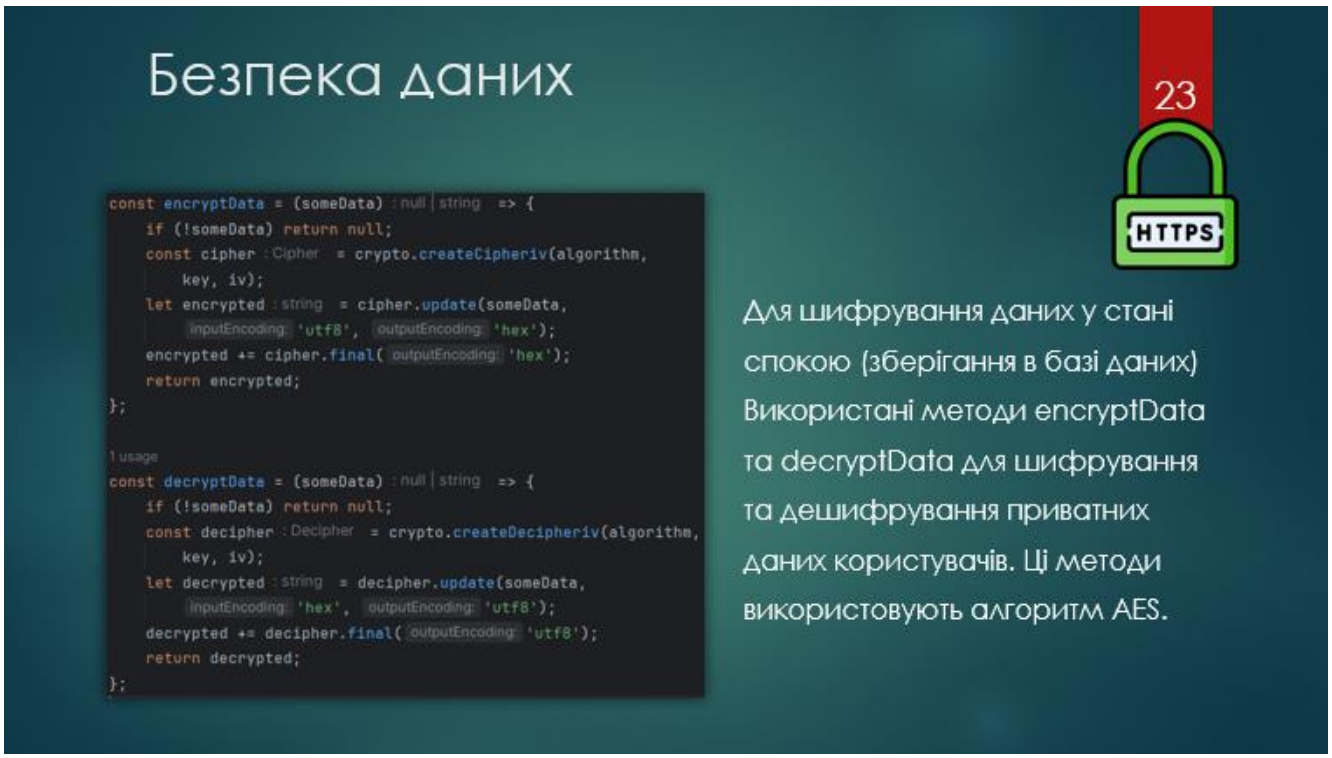

Рисунок Б.23 – Слайд №23 (рисунок виконаний самостійно)

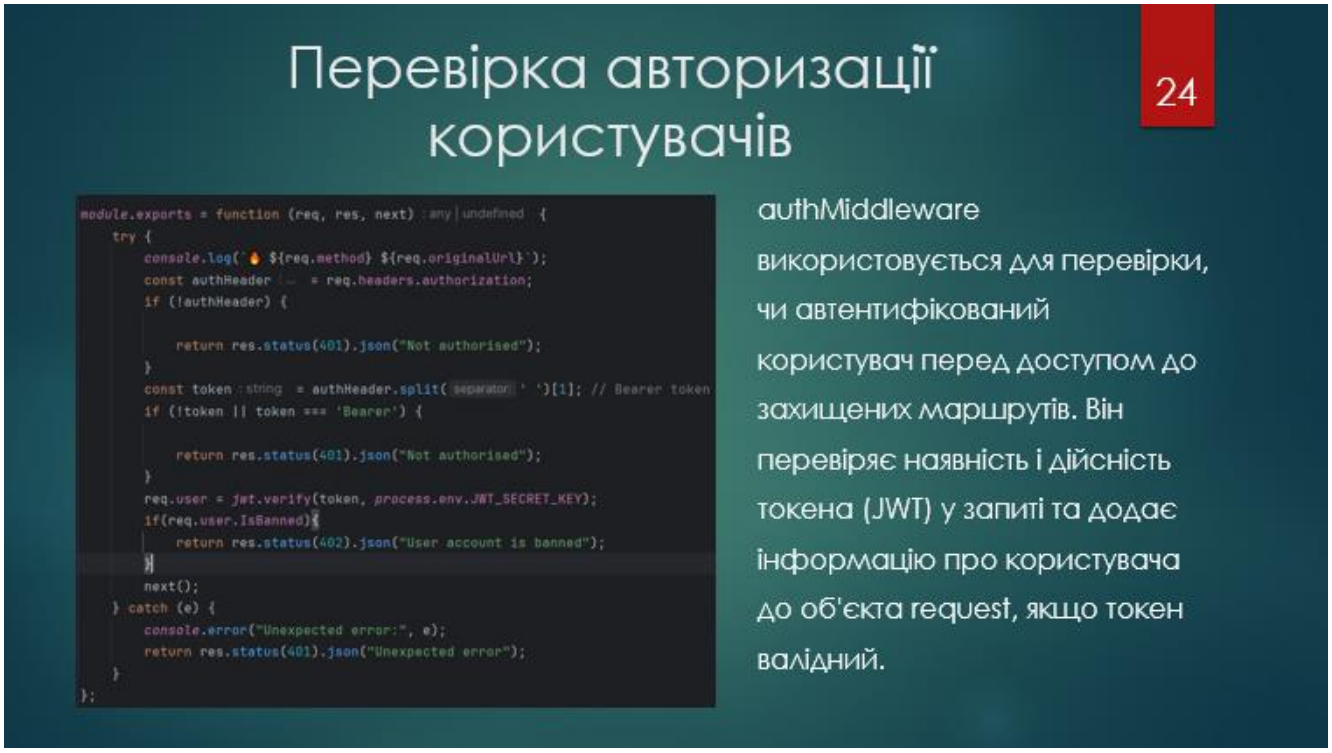

Рисунок Б.24 – Слайд №24 (рисунок виконаний самостійно)

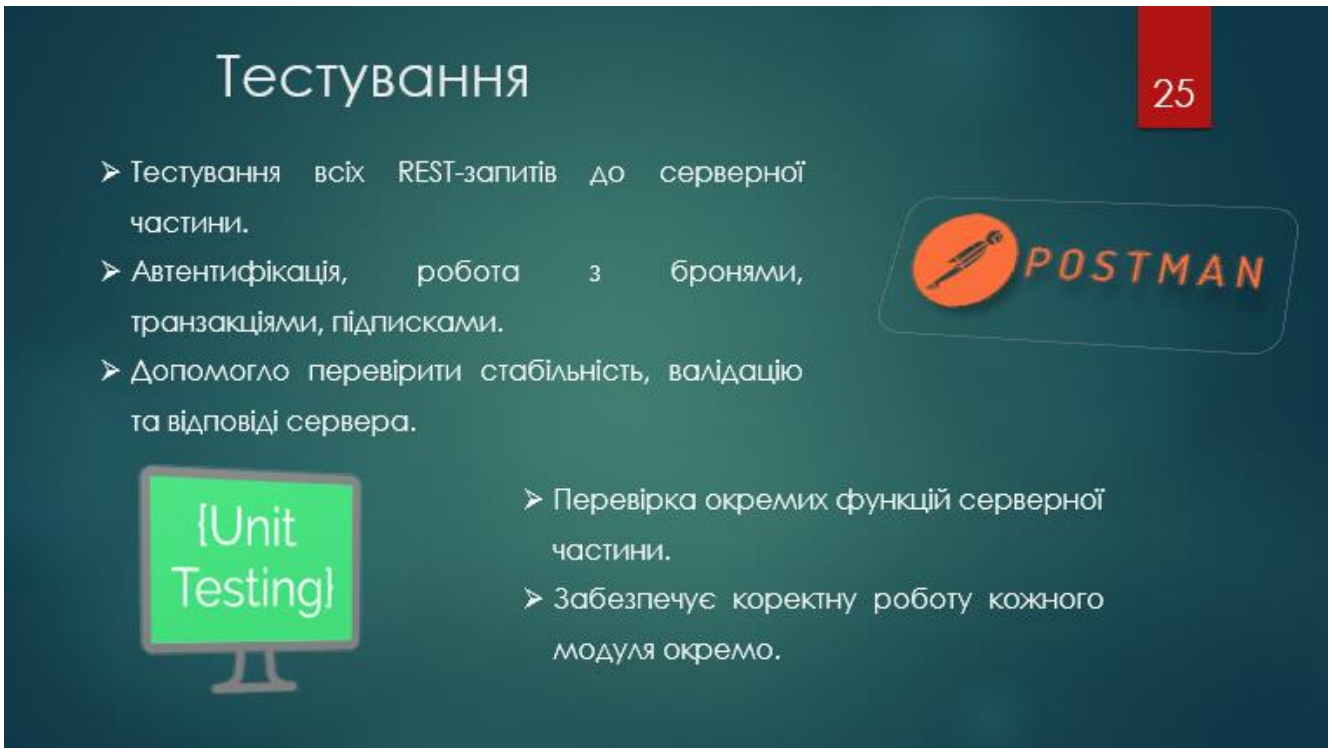

Рисунок Б.25 – Слайд №25 (рисунок виконаний самостійно)

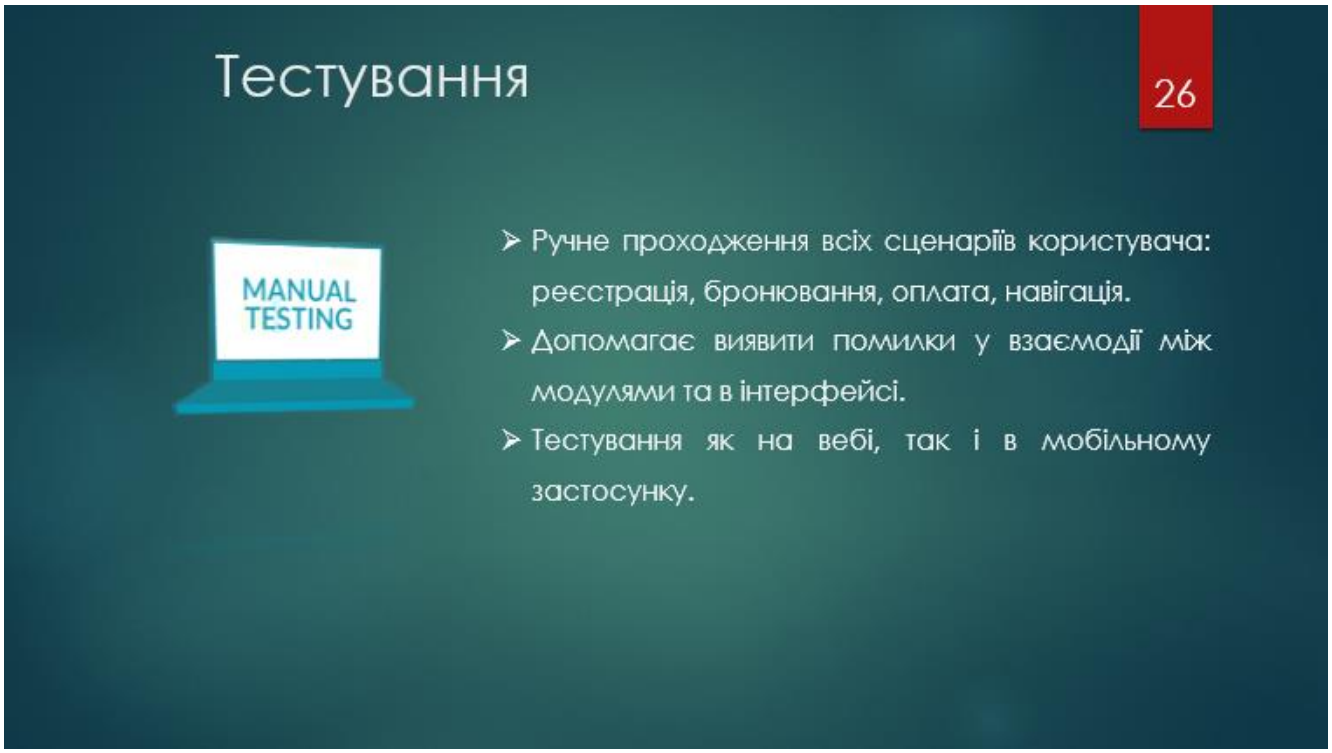

Рисунок Б.26 – Слайд №26 (рисунок виконаний самостійно)

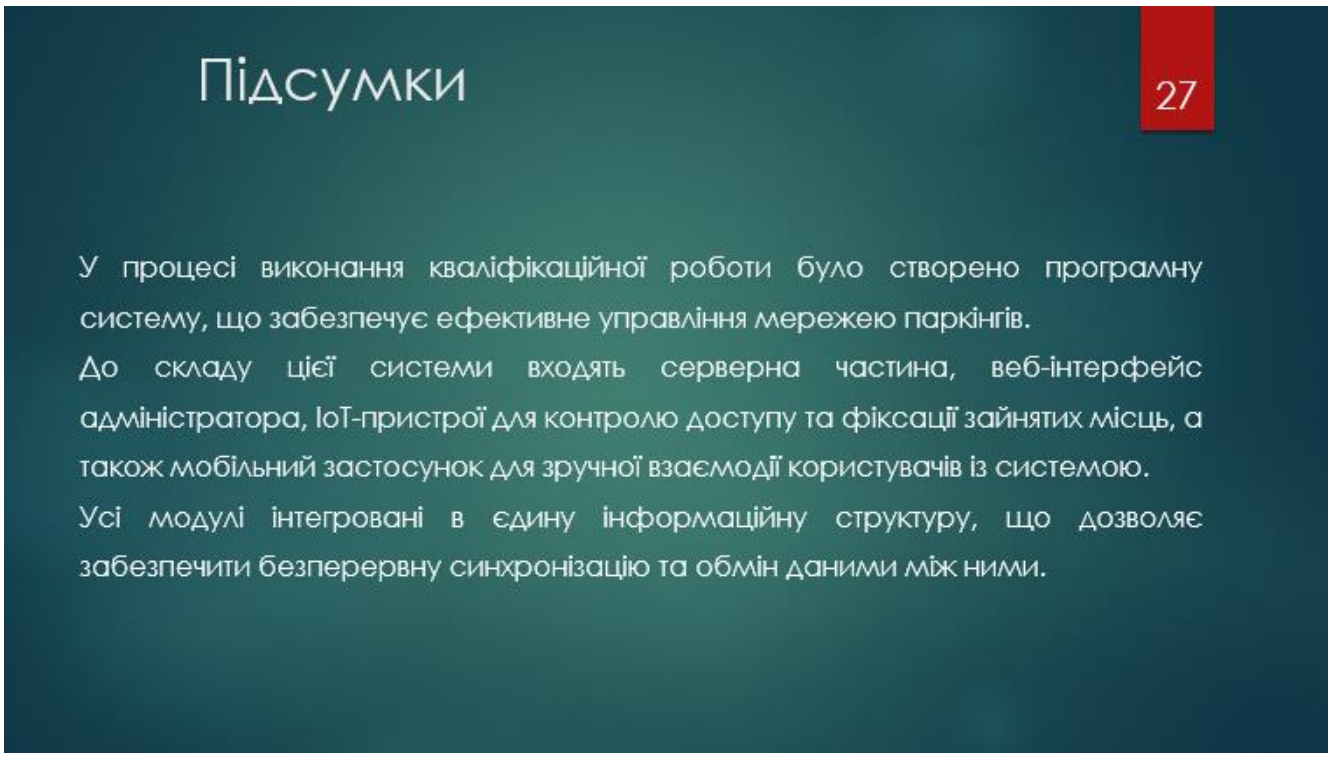

Рисунок Б.27 – Слайд №27 (рисунок виконаний самостійно)

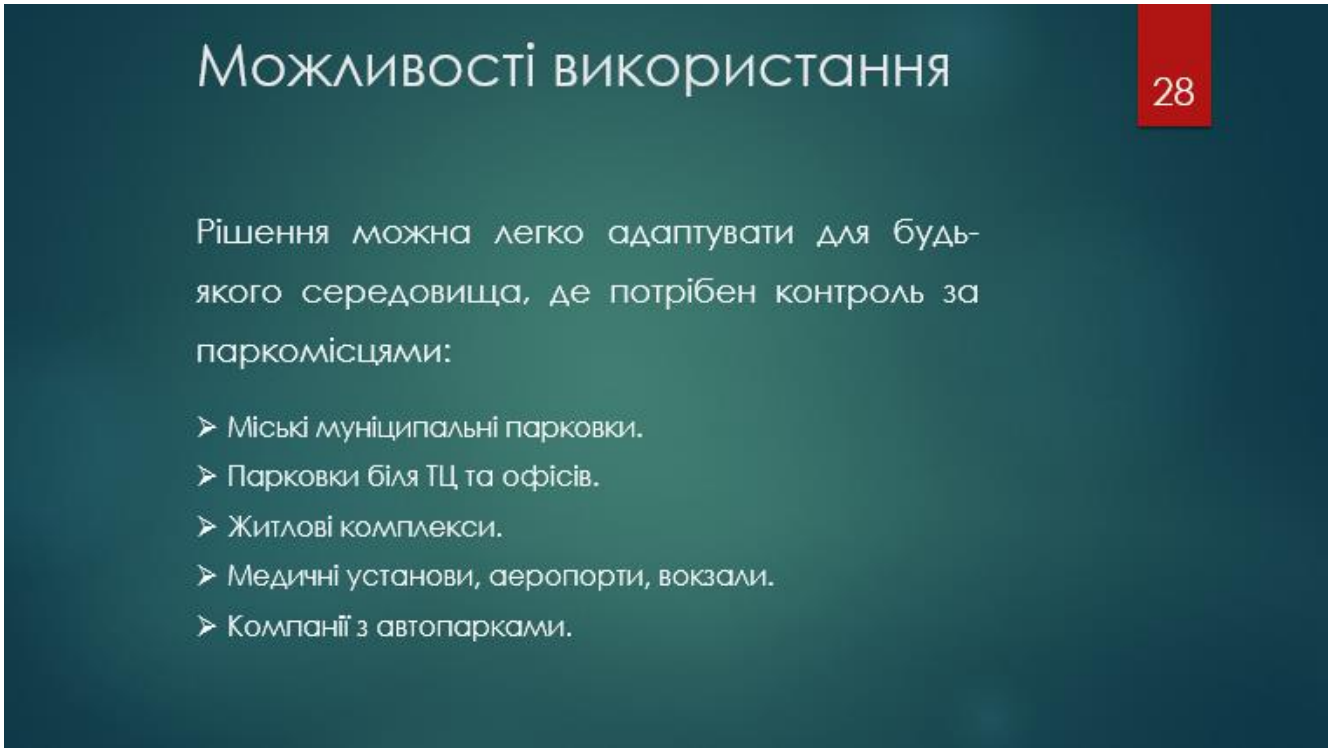

Рисунок Б.28 – Слайд №28 (рисунок виконаний самостійно)

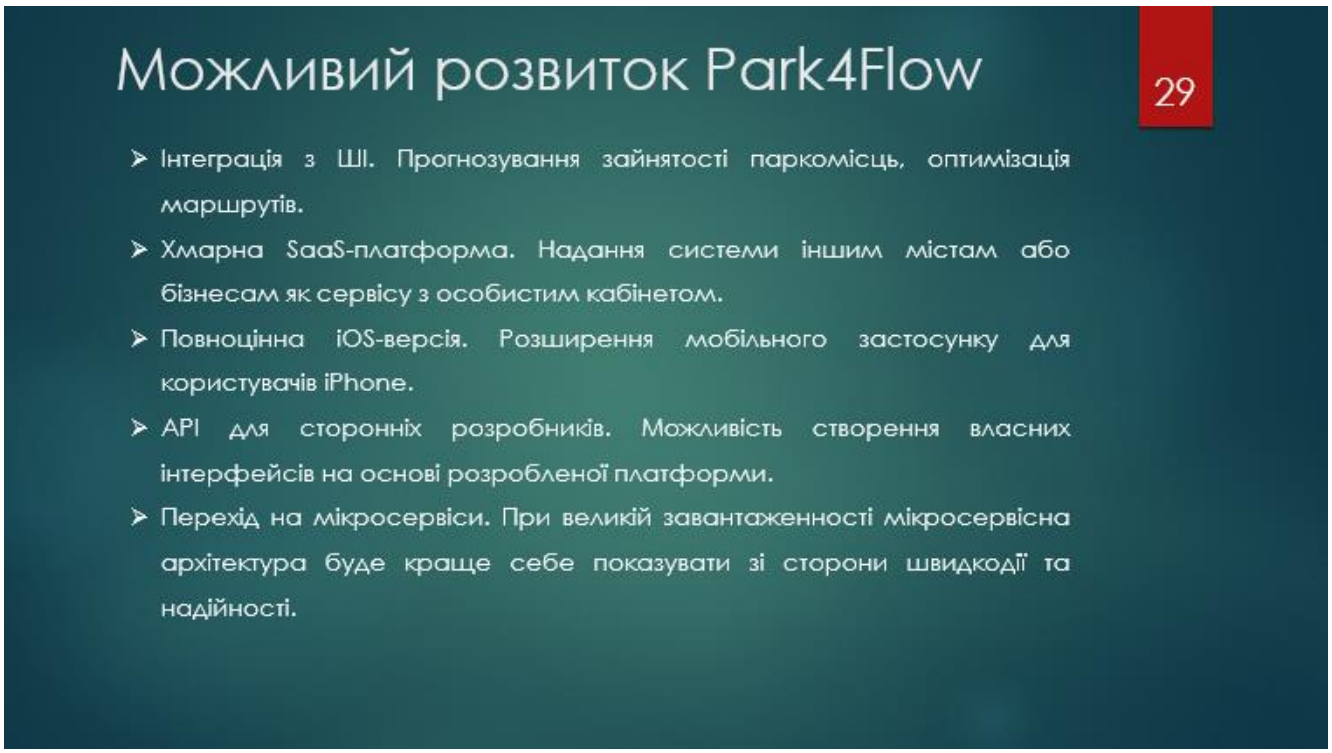

Рисунок Б.29 – Слайд №29 (рисунок виконаний самостійно)

Додаток В

Специфікація ПЗ

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук

Кафедра програмної інженерії

СПЕЦИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

програмної системи для керування платними парковками

Студент гр. ПЗПІ-21-10_____ Радько М. М.

Харків 2025

ЗМІСТ

1 ВСТУП

1.1 Огляд продукту

Програмне забезпечення, що розробляється, є комплексною системою для автоматизованого управління паркувальними майданчиками в умовах міської інфраструктури. Продукт спрямований на вирішення актуальних проблем, пов’язаних із дефіцитом паркомісць, неефективним використанням паркувального простору та відсутністю централізованого контролю. Система забезпечує облік, бронювання, моніторинг і оплату послуг паркування з використанням сучасних інформаційних та апаратних технологій.

Програмний продукт передбачає централізовану обробку даних про паркінги, транспортні засоби та користувачів, забезпечуючи безперервну взаємодію між усіма компонентами інфраструктури в реальному часі. Архітектура рішення охоплює серверну частину з бізнес-логікою, адміністративний веб-інтерфейс, мобільний клієнт для користувачів, а також IoT-пристрої для автоматичного обліку вільних місць і контролю доступу.

Система розробляється з використанням сучасного технологічного стеку, зокрема: PostgreSQL як СУБД, Node.js з Express для серверної частини, React для веб-інтерфейсу адміністратора, Kotlin для створення мобільного додатку, а також мікроконтролерів ESP32 з відповідними сенсорами для апаратної частини. Для реалізації функціоналу онлайн-оплати інтегровано підтримку популярних платіжних сервісів — PayPal і LiqPay, що дозволяє користувачам швидко й безпечно здійснювати оплату послуг безпосередньо через мобільний застосунок.

1.2 Мета

Метою створення програмного забезпечення є розробка інтегрованої інформаційної системи, призначеної для ефективного управління паркувальними майданчиками в умовах сучасної міської інфраструктури. Рішення передбачає централізоване зберігання та обробку даних, що стосуються об’єктів паркування, паркомісць, транспортних засобів, бронювань та зареєстрованих користувачів.

Система забезпечує взаємодію усіх складових паркувальної інфраструктури в режимі реального часу, що дозволяє здійснювати координацію процесів, моніторинг поточних станів та оптимізацію операційної діяльності. Програмний продукт включає серверну частину з реалізованою бізнес-логікою, адміністративну веб-панель для управління паркінгом, мобільний застосунок для кінцевих користувачів, а також апаратні засоби на базі IoT-пристроїв для автоматизованого обліку вільних місць та контролю доступу.

1.3 Межі

Програмне забезпечення має визначені межі функціональності та доступу, що залежать від ролей користувачів і специфіки реалізованої архітектури. Система передбачає поділ прав доступу між адміністраторами, операторами паркінгів та звичайними користувачами. Адміністративна частина надає розширені можливості для управління структурою паркінгів, конфігурацією обладнання, моніторингу активності та перегляду статистики, тоді як мобільний застосунок орієнтований на кінцевих користувачів і забезпечує лише функції пошуку, бронювання та оплати паркомісць.

Доступ до окремих функцій системи обмежується автентифікацією користувачів та перевіркою їхніх повноважень. Кожен запит до серверної частини

супроводжується авторизаційною перевіркою, що унеможливлює виконання критично важливих операцій неавторизованими особами.

Система не передбачає використання у відсутності стабільного інтернет-з’єднання, оскільки обробка даних та взаємодія між компонентами здійснюється через хмарну інфраструктуру. Також не підтримується інтеграція з паркувальними майданчиками, які не обладнані відповідними IoT-пристроями або не відповідають встановленим технічним вимогам. Підтримка веб-інтерфейсу реалізована для сучасних браузерів, мобільний додаток — виключно для операційної системи Android.

Обробка платежів обмежена підтримуваними платіжними сервісами (PayPal та LiqPay), що також визначає регіональні обмеження щодо використання функцій онлайн-оплати.

1.4 Посилання

Під час розробки програмного забезпечення використовуються зовнішні та внутрішні джерела, що визначають технічні, нормативні та функціональні основи системи. Основу проектування складає технічне завдання, в якому сформульовані вимоги до функціональності, архітектури та призначення системи. В процесі реалізації враховано офіційну документацію до використаних технологій, зокрема для середовищ Node.js, Express, React, PostgreSQL, Kotlin, Android SDK, а також апаратної платформи ESP32 з відповідними сенсорами та модулями.

Інтеграція платіжних засобів реалізована на основі специфікацій та API-документації платіжних сервісів PayPal і LiqPay, що визначають формат запитів, вимоги до безпеки, обробку транзакцій та правила взаємодії з клієнтськими додатками. Крім того, при роботі з персональними даними враховано положення чинного законодавства України, зокрема Закону «Про захист персональних даних»,

а також міжнародні стандарти якості програмного забезпечення, такі як ISO/IEC 25010:2011, що регламентує вимоги до надійності, безпеки, зручності використання та підтримуваності систем.

Зазначені джерела слугують базою для прийняття технічних рішень, побудови архітектури системи, реалізації функціональних модулів, а також для оцінки відповідності програмного продукту сучасним вимогам і практикам у сфері розробки програмного забезпечення.

1.5 Означення та абревіатури

СИСТЕМА — програмно-апаратний комплекс, що включає серверну частину, веб-інтерфейс адміністратора, мобільний додаток користувача та IoT-пристрої для автоматизованого управління паркуванням.

КОРИСТУВАЧ — фізична особа, що взаємодіє із системою через мобільний додаток або веб-інтерфейс з метою отримання паркувальних послуг (звичайний користувач або адміністратор).

ПАРКОМІСЦЕ — одиниця паркувального простору, інформація про яку зберігається у системі, включає статус (вільне/зайняте), координати та належність до конкретного паркінгу.

IOT (IoT) — Internet of Things, інтернет речей; сукупність пристроїв, підключених до мережі, що здійснюють автоматичний облік паркомісць, контроль доступу та передають дані до системи.

API — Application Programming Interface, інтерфейс прикладного програмування; набір визначень і протоколів для взаємодії компонентів системи між собою та з зовнішніми сервісами.

БД (БАЗА ДАНИХ) — централізоване сховище структурованих даних, що використовується для зберігання інформації про користувачів, паркомісця, бронювання та інші сутності системи.

UI — User Interface, інтерфейс користувача; графічне або текстове середовище, через яке користувач взаємодіє із програмною системою.

HTTP — HyperText Transfer Protocol, протокол прикладного рівня для передавання гіпертексту, який використовується для комунікації між клієнтами та сервером.

ESP32 — мікроконтролер із підтримкою Wi-Fi та Bluetooth, що використовується для реалізації апаратної частини системи обліку паркомісць та контролю доступу.

PAYPAL — міжнародна система електронних платежів, яка використовується у складі мобільного додатку для здійснення безпечної оплати паркувальних послуг.

LIQPAY — українська платіжна система, що забезпечує електронну обробку транзакцій та інтегрується з мобільним застосунком як альтернативний спосіб оплати.

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Розроблена програмна система управління паркувальними майданчиками має високий потенціал для подальшого розвитку та масштабування. З урахуванням сучасних тенденцій урбанізації, зростання кількості транспортних засобів та популяризації концепції «розумного міста», потреба в автоматизованих рішеннях для організації паркувального простору є надзвичайно актуальною. Запропоноване програмне забезпечення може бути адаптоване для використання в різних умовах — від муніципальних паркінгів до закритих стоянок житлових комплексів, торговельних центрів, аеропортів чи бізнес-центрів.

Завдяки модульній архітектурі система легко розширюється новими функціональними можливостями, зокрема інтеграцією з додатковими платіжними сервісами, впровадженням аналітичних інструментів для прогнозування завантаженості, додаванням підтримки відеонагляду або систем розпізнавання номерних знаків. Передбачена підтримка масштабування дозволяє використовувати систему як у межах одного об’єкта, так і на рівні міста чи мережі паркінгів різних операторів.

Інтерфейси та API системи відкривають можливості для інтеграції з іншими сервісами, зокрема навігаційними системами, мобільними додатками для транспорту, або муніципальними інформаційними платформами. Наявність мобільного додатку, веб-інтерфейсу та IoT-компонентів формує комплексне рішення, яке може бути комерціалізоване або використане в рамках соціально значущих ініціатив. У перспективі система може стати основою для створення повноцінної міської платформи паркування, орієнтованої на зручність користувачів, ефективне використання ресурсів та зниження навантаження на дорожню інфраструктуру.

2.2 Функції продукту

Програмна система управління паркувальними майданчиками реалізує широкий спектр функцій, які забезпечують ефективне, зручне та безпечне використання паркувального простору як для кінцевих користувачів, так і для адміністраторів. Однією з ключових функцій є реєстрація та аутентифікація користувачів із різними рівнями доступу. Це дозволяє розмежувати права та можливості між звичайними користувачами, які здійснюють бронювання паркомісць, та адміністраторами, які управляють роботою паркінгу, контролюють стан інфраструктури та проводять аналітику.

Для кінцевих користувачів система надає зручний мобільний додаток з інтуїтивно зрозумілим інтерфейсом. Користувачі можуть резервувати місця в реальному часі, отримувати підтвердження бронювання та переглядати історію своїх поїздок і платежів. Вбудовані повідомлення інформують користувачів про зміни статусу бронювання, наближення терміну оплати або інші важливі події.

Адміністративна панель системи надає розширені можливості управління паркувальними майданчиками. Адміністратори можуть додавати або змінювати інформацію про паркомісця, керувати бронюваннями, здійснювати моніторинг завантаженості майданчиків у режимі реального часу, отримувати звіти про використання ресурсів та фінансові операції. Система підтримує налаштування правил доступу та тарифів, що дозволяє гнучко адаптувати послуги під різні категорії користувачів і типи об’єктів.

Інтеграція з IoT-пристроями є важливою складовою системи. Завдяки використанню датчиків, мікроконтролерів ESP32 та інших апаратних рішень автоматично визначається статус паркомісць — зайняте чи вільне. Також реалізовано контроль доступу транспортних засобів за допомогою систем ідентифікації, що підвищує рівень безпеки та зменшує ризик несанкціонованого проникнення. Дані з IoT-пристроїв передаються у центральну систему в режимі

реального часу, що дозволяє підтримувати актуальність інформації та швидко реагувати на зміни.

Щодо фінансової складової, система підтримує інтеграцію з платіжними сервісами PayPal та LiqPay, що забезпечує надійність, безпеку та зручність проведення платежів. Користувачі можуть обирати комфортний для себе спосіб оплати послуг паркування безпосередньо через мобільний додаток. Реалізовано механізми підтвердження платежів, обробки транзакцій, а також ведення історії фінансових операцій із можливістю формування звітності.

Захист персональних даних та інформаційна безпека є пріоритетними у функціоналі системи. Використовуються сучасні методи шифрування, аутентифікації та авторизації, забезпечується безпечне зберігання та передавання даних між клієнтськими додатками, сервером та IoT-пристроями. Система підтримує резервне копіювання інформації та відновлення у разі виникнення збоїв.

Додатково, система реалізує функції моніторингу та аналітики, що дозволяє адміністраторам отримувати детальну статистику щодо використання паркомісць, поведінки користувачів, доходів від послуг та ефективності роботи обладнання. Це сприяє прийняттю обґрунтованих управлінських рішень і подальшому вдосконаленню сервісу.

2.3 Характеристики користувачів

Користувачі програмної системи поділяються на кілька основних категорій, кожна з яких має свої особливості, рівень доступу та функціональні потреби. До кінцевих користувачів належать водії, які користуються паркувальними послугами. Вони мають змогу реєструватися в системі, шукати та бронювати паркомісця, здійснювати оплату через інтегровані платіжні сервіси, а також переглядати

історію власних бронювань і платежів. Для зручності користувачів передбачено інтуїтивно зрозумілий мобільний додаток, що дозволяє виконувати всі необхідні операції у режимі реального часу.

Адміністратори паркувальних майданчиків виконують роль управлінців системою. Вони мають розширені права доступу, що дозволяють управляти інформацією про паркомісця, контролювати статус бронювань, моніторити завантаженість паркінгу, переглядати звіти та аналітику, а також налаштовувати параметри системи відповідно до потреб організації. Адміністратори також відповідають за підтримку працездатності IoT-пристроїв, які забезпечують автоматичний облік паркомісць і контроль доступу.

Користувачі з роллю технічного персоналу відповідають за обслуговування та підтримку апаратної частини системи, зокрема IoT-пристроїв, контролерів та мережевого обладнання. Вони мають доступ до спеціалізованих інструментів діагностики та оновлення системного програмного забезпечення.

Усі категорії користувачів взаємодіють із системою через відповідні інтерфейси — мобільний додаток для кінцевих користувачів, веб-інтерфейс для адміністраторів та технічного персоналу. Рівень доступу та функціональні можливості суворо регламентовані з метою забезпечення безпеки та ефективності роботи системи. Кожен користувач повинен проходити процедуру аутентифікації та отримувати відповідні права відповідно до своєї ролі.

2.4 Загальні обмеження

Програмна система Park4Flow має комплекс технічних, організаційних, правових та функціональних обмежень, які необхідно враховувати при її впровадженні, експлуатації та масштабуванні. Водночас існують певні

виключення, що визначають сфери, в яких відповідальність постачальника програмного забезпечення обмежена. До технічних обмежень належить залежність системи від стабільного інтернет-з’єднання, оскільки без нього неможливі обробка платежів, синхронізація бронювань, надсилання push-сповіщень та отримання аналітики. У регіонах з нестабільним мобільним покриттям можливі затримки або втрата даних. Обмежена апаратна сумісність означає, що система підтримує роботу тільки з рекомендованими моделями камер розпізнавання номерних знаків, датчиків заповненості та шлагбаумів, використання стороннього обладнання вимагає додаткової адаптації або заміни.

Фінансові та операційні обмеження включають обмежений перелік платіжних провайдерів, зокрема інтеграцію лише з LiqPay та PayPal, що може ускладнити використання системи у юрисдикціях, де ці сервіси обмежені чи заборонені. Регулярні оновлення програмного забезпечення є обов’язковими для підтримки стабільності та безпеки, однак вони можуть призводити до тимчасової недоступності деяких сервісів. Також впровадження і обслуговування системи потребує фінансових затрат на придбання обладнання та абонентську плату, а ефективне використання можливе лише за умови навчання персоналу, відсутність якого знижує якість сервісу.

З правових обмежень слід враховувати суворе дотримання вимог до обробки персональних даних відповідно до GDPR, CCPA та локального законодавства, а також можливу необхідність отримання ліцензій чи погоджень для легального використання системи у певних регіонах.

Функціональні обмеження передбачають, що динамічне ціноутворення можливе лише за наявності відповідно налаштованих алгоритмів, а інтеграція з картографічними сервісами базується на Google Maps з обмеженою підтримкою інших платформ без додаткової розробки. Зі сфери виключень варто зазначити, що система не підтримує роботу офлайн — відсутність інтернет-з’єднання робить неможливим оплату, бронювання та отримання сповіщень. Постачальник не несе відповідальності за неполадки зовнішніх платіжних та картографічних сервісів,

несумісність із несертифікованим обладнанням, а також обмеження щодо підтримки альтернативних методів оплати, таких як криптовалюти чи бартер.

Щодо безпеки, хоча система застосовує сучасні протоколи шифрування, вона не гарантує повний захист від усіх кіберзагроз, включаючи цільові атаки або витоки через сторонні пристрої. Відповідальність за помилки користувачів, наприклад, при введенні номерного знака, також несе користувач.

У разі змін законодавства адаптація системи може займати час та потребувати додаткових ресурсів, при цьому постачальник не відповідає за можливі втрати через затримки у відповідності. Крім того, система може тимчасово не працювати під час планових технічних перерв або обслуговування, про що користувачі попереджаються заздалегідь. У форс-мажорних ситуаціях, таких як природні катастрофи, масштабні кібератаки, відключення електропостачання чи банкрутство оператора паркінгу, стабільність роботи системи не гарантується.

2.5 Припущення й залежності

Для успішної розробки, впровадження та подальшої експлуатації програмної системи Park4Flow робляться певні припущення, які забезпечують належне функціонування та ефективність системи. По-перше, передбачається наявність стабільного та високошвидкісного інтернет-з’єднання на території паркінгів, що інтегрують систему. Це є необхідною умовою для безперебійної роботи функцій обробки оплат, синхронізації даних у реальному часі та взаємодії із хмарними сервісами. Крім того, оператори паркінгів повинні мати сучасне технічне обладнання, таке як комп’ютери, POS-термінали, планшети або мобільні пристрої з необхідними характеристиками, які забезпечують комфортну та повноцінну роботу з системою.

Важливою передумовою є також наявність базових навичок у користувачів системи. Очікується, що персонал паркінгів і водії, пройшовши навчання у вигляді відеоінструкцій, документації та інтерактивних тренінгів, володітимуть усіма необхідними знаннями для ефективного використання Park4Flow. Впровадження системи передбачає активну участь адміністрацій паркінгів, які повинні забезпечити доступ до навчальних матеріалів, організацію навчання персоналу, а також надавати зворотний зв’язок для подальшого вдосконалення системи.

Також до важливих припущень належить дотримання чинного законодавства. Система буде розроблена та адаптована відповідно до актуальних нормативно-правових вимог, зокрема в частині обробки персональних даних, захисту інформації та проведення електронних фінансових операцій. Перед запуском проекту передбачається отримання всіх необхідних ліцензій і дозволів, що дозволить використовувати Park4Flow у законодавчому полі конкретних країн чи регіонів.

Фінансування проекту має бути забезпечене повністю на всіх етапах життєвого циклу системи — від розробки до тестування, впровадження та подальшої технічної підтримки. До бюджету проекту також включаються витрати на навчання персоналу, супровід користувачів, оновлення програмного забезпечення та супутню документацію. Очікується, що кінцеві користувачі — оператори паркінгів і водії — будуть зацікавлені у застосуванні системи завдяки її зручному інтерфейсу, широкому функціоналу та можливості оптимізувати операційні витрати.

Крім того, важливою умовою є активна співпраця адміністрацій паркінгів з командою розробників для оперативного вирішення технічних питань, внесення необхідних змін і покращень функціональності системи.

Ефективність роботи Park4Flow залежить також від зовнішніх та внутрішніх факторів. Система повинна безперебійно інтегруватися з різноманітними зовнішніми сервісами, такими як платіжні шлюзи, картографічні API (наприклад, Google Maps), служби сповіщень та аналітичні інструменти. Надійність цих

технологічних рішень є критичною, адже стабільність роботи хмарних платформ, банківських процесингових центрів і зовнішніх API безпосередньо впливає на функціонування системи.

Для підтримки актуальності, безпеки та відповідності ринку Park4Flow передбачає регулярне оновлення програмного забезпечення. Важливою складовою є також наявність кваліфікованої технічної підтримки користувачів, що включає в себе чат-боти, базу знань та підтримку у режимі реального часу, що сприяє стабільному та комфортному користуванню системою.

Система має відповідати чинним і майбутнім законодавчим вимогам у сфері обробки даних, кібербезпеки та електронної комерції. Її архітектура повинна бути гнучкою для оперативної адаптації до змін у законодавстві, безпекових стандартах або потребах ринку. Важливою складовою розвитку є збір і аналіз зворотного зв’язку від користувачів, що дозволить цілеспрямовано вдосконалювати інтерфейс, логіку роботи та функціональність системи.

Також Park4Flow повинна мати достатню продуктивність і масштабованість для обслуговування зростаючої кількості користувачів без втрати якості обслуговування. Ефективне просування системи на ринку забезпечується комплексними маркетинговими заходами, спрямованими на залучення нових користувачів. Для підтримки життєздатності продукту необхідна стабільна професійна команда розробників, здатна оперативно вирішувати технічні завдання, розвивати функціонал і забезпечувати відповідність системи сучасним вимогам.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Користувацький інтерфейс реалізовано з використанням бібліотеки React, яка є однією з найпопулярніших технологій для розробки односторінкових додатків (SPA). Вибір React зумовлений високою швидкістю роботи, гнучкістю компонентного підходу та активною спільнотою розробників.

Інтерфейс спроектовано відповідно до принципів UX/UI-дизайну. Кожна функціональність, така як реєстрація, вхід, бронювання, перегляд паркомісць, історія транзакцій чи налаштування профілю, представлена у вигляді окремого компонента або модуля. Завдяки цьому досягається висока модульність, повторне використання коду та зручність підтримки.

Валідація користувацького вводу реалізована як на клієнтській стороні, так і на сервері для забезпечення гнучкого опису правил перевірки, зокрема перевірки обов’язковості полів, формату електронної пошти, мінімальної довжини пароля тощо. При виявленні помилки валідації користувач отримує зрозуміле повідомлення, яке дозволяє швидко виправити некоректні дані.

Крім того, передбачено обробку помилок при взаємодії з API. Якщо сервер повертає помилку (наприклад, неправильні облікові дані при вході або відсутність доступу до ресурсу), на інтерфейсі виводиться відповідне повідомлення з поясненням проблеми. У випадку втрати підключення до інтернету або недоступності сервера користувач також інформується через інтерфейс.

Уся навігація між сторінками здійснюється без повного перезавантаження завдяки використанню бібліотеки React Router, що забезпечує плавний користувацький досвід. Інтерфейс адаптований під мобільні пристрої за

допомогою CSS Grid, що дозволяє комфортно користуватися додатком з будь-якого пристрою — смартфона, планшета чи настільного комп’ютера.

Загалом, інтерфейс користувача орієнтований на простоту, швидкодію та зручність використання, з акцентом на доступність та мінімізацію дій, необхідних для досягнення цілі.

3.1.2 Програмний інтерфейс

Програмний інтерфейс системи реалізовано у вигляді набору RESTful API-ендпоінтів, які надають функціональність для клієнтських додатків. Інтерфейс побудований відповідно до принципів REST, що забезпечує логічну організацію доступу до ресурсів та зручність розширення системи в майбутньому.

Кожен ресурс системи, такий як користувачі, транспортні засоби, паркомісця, підписки, платежі, бронювання тощо, має окремі маршрути, які обробляють відповідні HTTP-запити.

API розроблено з урахуванням принципів модульності та розмежування відповідальності: логіка обробки запитів винесена в окремі контролери, а робота з базою даних реалізована через рівень репозиторіїв. Це дозволяє легко тестувати, масштабувати та підтримувати систему.

На стороні клієнта (веб- або мобільного додатку) здійснюється виклик відповідних ендпоінтів для взаємодії з бекендом. Усі відповіді мають стандартну структуру: код стану HTTP, повідомлення про успішність або помилку, та, за необхідності, тіло з даними.

Особливу увагу приділено обробці помилок: API завжди повертає інформативні повідомлення у випадках неправильного запиту, відсутності прав доступу, помилок валідації або внутрішніх збоїв сервера.

3.1.3 Комунікаційний протокол

Для обміну даними між клієнтськими застосунками (веб-інтерфейсом, мобільним застосунком) і серверною частиною системи використовується комунікаційний протокол HTTPS, використання якого забезпечує шифрування переданих даних за допомогою SSL/TLS, що гарантує конфіденційність, цілісність та аутентичність.

Усі запити до REST API виконуються виключно через HTTPS-з’єднання за визначеним IP-адресом та портом (https://192.168.31.250:5192). Доступ до захищених ресурсів здійснюється тільки після проходження автентифікації.

Авторизація користувача реалізована за допомогою токенів формату JWT, які передаються в заголовку Authorization кожного запиту. Це дозволяє чітко контролювати доступ до функціональності в залежності від ролі користувача (адміністратор, звичайний користувач, тощо).

Запити та відповіді формуються у форматі JSON, що дозволяє легко обробляти дані як на фронтенді, так і в мобільному застосунку.

Таким чином, комунікація в системі є безпечною, надійною та відповідною сучасним стандартам передачі даних в Інтернеті.

3.1.4 Обмеження пам’яті

Система має працювати ефективно в умовах обмежених обсягів оперативної пам’яті, особливо на мобільних пристроях користувачів та серверному хостингу із фіксованими ресурсами.

Для клієнтської частини (веб та мобільного застосунку) обмеження пам’яті враховуються шляхом:

– використання оптимізованих компонентів інтерфейсу користувача (уникається зберігання великих об'єктів у стані);

– очищення зайвих даних з оперативної пам’яті після завершення дій;

– мінімального використання великих зображень та медіафайлів у додатку;

– обмеження кешу локального сховища (до 5 МБ для вебзастосунку).

Для серверної частини система має працювати стабільно на сервері з мінімальною конфігурацією:

– ОЗП: від 1 ГБ — з урахуванням обмеженого числа одночасних підключень;

– процесор: від 1 vCPU — достатньо для обробки запитів REST API без високонавантажених обчислень;

– всі тимчасові дані (у тому числі зображення та файли) зберігаються у файловій системі лише на час обробки, після чого автоматично видаляються;

– застосовується потокова обробка файлів без повного завантаження в пам’ять (наприклад, при завантаженні фото).

Крім того, при проектуванні моделі бази даних враховано обмеження на розмір записів, уникається зберігання великих JSON-структур або файлів безпосередньо в БД.

Таким чином, система залишається стабільною і продуктивною навіть в умовах обмежених апаратних ресурсів.

3.1.5 Операції

Зовнішні інтерфейси системи надають набір операцій, доступних через клієнтські застосунки (веб та мобільний), а також за допомогою програмного інтерфейсу (REST API). Ці операції реалізують ключову функціональність системи

керування паркуванням та забезпечують взаємодію користувачів з усіма компонентами.

Основні групи операцій включають:

– Реєстрація та автентифікація користувачів – створення акаунтів, вхід у систему, перевірка прав доступу.

– Управління користувачами – оновлення профілю, блокування, скидання паролю.

– Робота з транспортними засобами – додавання, оновлення, видалення та перегляд власного автопарку.

– Управління тарифами та підписками – перегляд актуальних тарифів, створення підписок через PayPal або LiqPay.

– Бронювання місць для паркування – створення, перегляд, скасування та пропуск бронювань.

– Платежі та поповнення балансу – ініціалізація та підтвердження оплат різними платіжними сервісами.

– Виконання паркувальних дій – старт та зупинка паркування вручну або за допомогою IoT-модулів.

– Робота з політиками (знижки, скасування) – додавання, оновлення або видалення правил та політик.

– Пошук та аналітика – доступ до інформації про паркувальні місця, історію дій, улюблені паркінги, аналіз доступності.

Усі операції доступні через захищений канал зв’язку (HTTPS) та потребують належної автентифікації за допомогою токена доступу (JWT).

Інтерфейс розроблений таким чином, щоб користувач мав змогу інтуїтивно виконувати необхідні дії за кілька кроків. При цьому на рівні API підтримується модульність, логічна структурованість запитів та чітке розділення обов’язків між ендпоінтами.

3.1.6 Функції продукту

Зовнішні інтерфейси програмної системи повинні забезпечувати доступ до всіх основних функцій продукту згідно з їх призначенням. Вимоги до реалізації цих функцій через інтерфейси охоплюють як функціональність для кінцевих користувачів, так і для адміністраторів системи.

Для кінцевих користувачів через мобільний додаток та веб-інтерфейс мають бути доступні такі функції:

– Реєстрація, автентифікація, авторизація користувача (через API із захищеною передачею даних).

– Перегляд доступних паркувальних майданчиків у режимі реального часу.

– Бронювання паркомісця та отримання підтвердження.

– Відображення стану активних бронювань і повідомлень про зміни.

– Проведення оплат за допомогою інтегрованих сервісів PayPal та LiqPay.

– Перегляд історії транзакцій, поїздок, бронювань.

– Отримання push- або системних повідомлень щодо статусу бронювання, нагадувань, завершення часу тощо.

– Зміна налаштувань профілю, видалення акаунту.

Для адміністратора через веб-інтерфейс доступні:

– Управління користувачами, їх ролями та правами доступу.

– Додавання, редагування або видалення паркувальних майданчиків та окремих місць.

– Перегляд аналітики використання паркінгів, завантаженості, фінансової статистики.

– Контроль платежів, знижок, тарифів та правил надання послуг.

– Моніторинг даних з IoT-пристроїв (ESP32, сенсори) щодо стану паркомісць.

– Реагування на нештатні ситуації, перегляд журналів подій.

– Експорт звітів по бронюваннях, доходах, роботі обладнання.

Вимоги до реалізації:

– Усі функції повинні бути доступні через стандартизовані REST API з чітким описом параметрів і відповідей.

– Мобільний застосунок і веб-інтерфейс мають бути реалізовані відповідно до принципів UX-дизайну для забезпечення зручності користувачів.

– Доступ до критичних функцій (фінанси, управління доступом, адміністрування) повинен бути захищений та обмежений роллю користувача.

– Обробка помилок та повідомлення про них мають бути інтегровані як у UI, так і в API (через відповідні статус-коди та повідомлення).

Усі функції повинні працювати стабільно незалежно від платформи (мобільний додаток чи веб), а дані – синхронізуватися через бекенд у режимі реального часу.

3.1.7 Припущення й залежності

Успішна робота зовнішніх інтерфейсів системи Park4Flow (мобільного застосунку, веб-інтерфейсу, REST API та IoT-взаємодії) базується на низці припущень і залежностей, які необхідно враховувати під час розробки, впровадження та експлуатації системи.

Припущення:

– Припускається, що всі користувачі, які взаємодіють із зовнішніми інтерфейсами (водії, оператори, адміністратори), мають стабільне підключення до Інтернету, що забезпечує доступ до функцій у режимі реального часу.

– Очікується, що кінцеві користувачі мають сучасні пристрої (Android/iOS смартфони або комп’ютери з сучасними браузерами), які сумісні з клієнтськими додатками.

– Адміністрація паркінгів забезпечить використання сумісного обладнання для підключення до IoT-пристроїв та веб-інтерфейсу (наприклад, ПК з підтримкою HTTPS і стабільним з’єднанням).

– Користувачі та оператори пройдуть початкове навчання або ознайомлення з інтерфейсами системи (доступ через інтуїтивно зрозумілий UI, інструкції, довідка).

– Ключові функції зовнішніх інтерфейсів (реєстрація, оплата, бронювання, перегляд паркінгів) не потребуватимуть додаткових дій з боку користувача поза межами стандартного функціоналу.

Залежності:

– Зовнішні інтерфейси напряму залежать від стабільної роботи сторонніх API (Google Maps, платіжні шлюзи LiqPay/PayPal, служби аутентифікації), які забезпечують ключову функціональність.

– Безперебійна робота REST API для мобільного додатку, веб-інтерфейсу та IoT-пристроїв залежить від хмарної інфраструктури (сервери, CDN, бази даних).

– Усі зовнішні інтерфейси повинні регулярно оновлюватися згідно з новими вимогами безпеки, змінами в протоколах, оновленнями платформ (Android/iOS/браузери).

– Сумісність мобільного додатку з останніми версіями ОС Android та iOS залежить від дотримання вимог відповідних маркетів (Play Market, App Store).

– Інтерфейси IoT-компонентів (ESP32 тощо) залежать від стабільності передачі даних через Wi-Fi та коректної обробки пакетів REST/WebSocket-запитів.

– Взаємодія між клієнтськими інтерфейсами та бекендом має бути захищена сучасними методами шифрування (HTTPS, JWT, OAuth).

– Надійна технічна підтримка необхідна для вирішення проблем із доступом або помилками в роботі інтерфейсів у режимі реального часу.

– Масштабування кількості запитів до API та інтерфейсів прямо залежить від продуктивності серверної частини системи та кешування запитів.

3.2 Атрибути програмного продукту

Програмний продукт Park4Flow створюється для надійного, зручного та безпечного управління паркувальними майданчиками. Для успішного функціонування та задоволення потреб користувачів система повинна відповідати ряду ключових властивостей, які визначають якість та ефективність продукту.

3.2.1 Надійність

Надійність системи є критично важливою для забезпечення безперервної роботи сервісу, особливо у випадку управління паркувальними майданчиками, де від доступності системи залежить комфорт користувачів і дохід операторів. Park4Flow має бути здатним працювати без збоїв і швидко відновлюватися після можливих помилок.

– Система повинна забезпечувати стабільну роботу 24/7 без збоїв.

– Критичні зовнішні інтерфейси (API, мобільний додаток, веб-інтерфейс) повинні мати постійний доступ до мережі та електроенергії.

– Передбачено механізми обробки помилок і аварійне відновлення (наприклад, повторні запити, резервне копіювання даних).

3.2.2 Доступність

Доступність системи має важливе значення, адже користувачі повинні мати можливість отримувати сервіс у будь-який час, без обмежень за часом або місцем, особливо у випадках, коли паркування є критичною послугою.

– Система повинна забезпечувати високий рівень доступності (не менше 99.5%) і підтримувати роботу в режимі 24/7.

– Передбачена можливість роботи мобільного додатку в офлайн-режимі з подальшою синхронізацією даних.

– Серверна інфраструктура має бути резервована для мінімізації часу простою.

3.2.3 Безпека

Забезпечення безпеки є критично важливою вимогою до програмної системи Park4Flow, оскільки вона обробляє чутливу інформацію користувачів — включно з персональними даними, платіжними реквізитами та історією паркувань. Надійний захист цих даних є основою довіри користувачів до сервісу, а також необхідною умовою відповідності міжнародним стандартам і законодавству про захист персональної інформації.

У системі обробляються кілька категорій конфіденційних даних, зокрема ім’я, прізвище, контактна інформація, дані транспортного засобу, а також фінансова інформація — баланс рахунку та історія платежів. Дані банківських карток не зберігаються на сервері, а всі транзакції здійснюються через сертифіковані платіжні шлюзи з високим рівнем захисту. Також система працює з геолокаційними даними — координатами припаркованих авто, часом паркування

та історією бронювань, — що потребує особливої уваги до приватності користувача.

Щоб запобігти витоку або несанкціонованому доступу, у Park4Flow реалізовано низку технічних і організаційних заходів. Усі дані шифруються — як у спокої (з використанням AES-256), так і під час передачі (через TLS 1.3). Щодня створюються резервні копії, які зберігаються у захищеному середовищі з обмеженим доступом. При роботі з користувачами застосовується двофакторна аутентифікація, а також політика доступу на основі ролей, яка дозволяє обмежити доступ до критичної інформації лише уповноваженим співробітникам. У разі підозрілої активності користувач отримує автоматичне сповіщення, що дозволяє швидко реагувати на потенційні інциденти безпеки.

Окрему увагу приділено відповідності нормативним вимогам. Park4Flow відповідає положенням GDPR: користувачі мають повний контроль над своїми персональними даними — вони можуть переглядати, редагувати або видаляти інформацію у власному кабінеті. Також система дотримується принципів прозорості та отримання згоди на обробку даних. З боку інфраструктури впроваджено стандарти інформаційної безпеки ISO/IEC 27001, які регулюють контроль доступу, захист на рівні серверів і управління ризиками.

Користувачі мають змогу в будь-який момент ознайомитися з актуальною політикою конфіденційності та умовами використання. При оновленні цих документів система повідомляє користувача автоматично. Усі запити на видалення або редагування персональних даних обробляються відповідно до чинного законодавства та внутрішньої політики обробки інформації.

Таким чином, система Park4Flow забезпечує комплексний підхід до захисту даних — як технічно, так і нормативно — і створює безпечне середовище для зберігання, обробки й використання персональної інформації користувачів.

3.2.4 Продуктивність

Для користувачів дуже важливо, щоб система реагувала швидко, особливо під час здійснення бронювання чи оплати. Висока продуктивність гарантує комфортний досвід та запобігає втраті клієнтів через затримки або зависання.

– Відгук зовнішніх інтерфейсів (UI/API) не повинен перевищувати 1–2 секунд при нормальному навантаженні.

– API повинно обробляти щонайменше 100 запитів на секунду без втрати стабільності.

– IoT-інтерфейси повинні передавати оновлення про статус паркомісць з інтервалом не більше 5 секунд.

3.2.5 Супроводжуваність

Легка підтримка та оновлення системи забезпечує довготривалу ефективність та швидку адаптацію до нових вимог ринку.

– Кодова база повинна мати зрозумілу структуру, логування, коментарі та відповідати принципам SOLID.

– Передбачено автоматизоване тестування (unit, integration, UI), CI/CD та документацію API (наприклад, Swagger/OpenAPI).

– Адміністративна панель має функціональність для оновлення тарифів, керування паркінгами та генерації звітів без потреби залучати розробників.

3.2.6 Зручність використання

Зручність і простота інтерфейсів є запорукою широкого прийняття системи серед користувачів різного рівня технічної підготовки. Park4Flow орієнтований на максимальне спрощення взаємодії з сервісом.

– Інтерфейси для водіїв (мобільний застосунок) повинні бути інтуїтивно зрозумілими, із мінімальною кількістю кроків до здійснення бронювання чи оплати.

– Для операторів передбачена панель адміністрування з фільтрами, аналітикою та візуалізацією даних.

– Передбачена локалізація UI мінімум на дві мови: українську та англійську.

3.3 Вимоги бази даних

Програмна система Park4Flow потребує надійної, масштабованої та безпечної бази даних для зберігання ключової інформації: користувацьких профілів, транспортних засобів, паркувальних сесій, транзакцій, геолокаційних координат, журналів подій та технічних логів. До бази даних висуваються кілька критично важливих вимог: підтримка транзакційності, забезпечення цілісності даних, швидка обробка запитів, можливість роботи з геоданими, масштабованість, безпечне зберігання конфіденційної інформації та підтримка резервного копіювання.

У контексті цих вимог оптимальним вибором є PostgreSQL — потужна об’єктно-реляційна система управління базами даних, що відповідає стандарту ACID і підтримує складні структури даних та індексацію. По-перше, PostgreSQL забезпечує повну транзакційну підтримку: будь-яка дія в системі, пов’язана з

оплатою, бронюванням або зміною даних користувача, виконується у межах транзакції, що гарантує узгодженість навіть у разі помилки або збою.

Крім того, PostgreSQL підтримує зовнішні ключі, унікальні обмеження, перевірки значень (CHECK), що дозволяє реалізувати сувору модель цілісності даних — наприклад, бронювання не може існувати без відповідного автомобіля та користувача. Це особливо важливо для динамічної структури системи, де багато сутностей взаємопов’язані.

Ще однією критичною вимогою є робота з геолокаційними даними. PostgreSQL разом з розширенням PostGIS дозволяє зберігати координати, виконувати просторові запити (наприклад, пошук найближчих паркінгів) і оптимізувати ці операції за допомогою спеціалізованих індексів. Це значно підвищує продуктивність та точність пошуку паркувальних місць на основі місцезнаходження.

Безпека даних — ще один ключовий аспект. PostgreSQL підтримує шифрування з’єднань (SSL/TLS), налаштування доступу за ролями, розмежування прав користувачів та аудит змін. Це дозволяє реалізувати надійний контроль доступу до конфіденційної інформації та відповідати вимогам GDPR та ISO 27001.

У контексті масштабованості та надійності PostgreSQL забезпечує підтримку реплікації, резервного копіювання та кластеризації, що дозволяє системі зростати разом із кількістю користувачів без втрати продуктивності. Також система підтримує індексацію великих таблиць, розбиття (partitioning) та асинхронну реплікацію, що є критичним для збереження високої доступності даних.

Усі ці властивості роблять PostgreSQL ідеальним кандидатом для використання в системі Park4Flow, де важлива надійність, масштабованість, безпека та розширена підтримка географічних і фінансових даних.

Додаток Г

Функція зі створення бронювання

```
1. /**
2.  * Create a reservation for a given parking and assign an optimal
park place.
3.  */
4. async createReservation(req, res) {
5.     try {
6.         const { ParkingID, VehicleID, StartTime, EndTime } =
req.body;
7.
8.         if (!ParkingID || !VehicleID || !StartTime || !EndTime) {
9.             return res.status(400).json({ message: 'ParkingID,
VehicleID, StartTime and EndTime are required.' });
10.         }
11.
12.         // UTC
13.         const startUTC = new Date(StartTime).toISOString(); //
ISO 8601 format in UTC
14.         const endUTC = new Date(EndTime).toISOString();
15.
16.         if (isNaN(Date.parse(startUTC)) ||
isNaN(Date.parse(endUTC))) {
17.             return res.status(400).json({ message: 'Invalid date
format for StartTime or EndTime.' });
18.         }
19.
20.         const vehicle = await Vehicle.findByPk(VehicleID);
21.         if (!vehicle) {
22.             return res.status(404).json({ message: 'Vehicle not
found.' });
23.         }
24.
25.         const vehicleCategory = vehicle.VehicleCategory; // A, B,
C, D
26.
27.         const parkPlaces = await ParkPlace.findAll({
28.             where: {
29.                 ParkingParkingID: ParkingID,
30.                 IsTaken: false,
31.                 VehicleCategory: vehicleCategory
32.             }
33.         });
34.
35.         if (parkPlaces.length === 0) {
36.             return res.status(404).json({ message: 'No compatible
park places available for your vehicle category.' });
37.         }
38.
39.         const reservations = await Reservation.findAll({
40.             include: [{
41.                 model: ParkPlace,
42.                 where: {
43.                     ParkingParkingID: ParkingID,
```

```
                    VehicleCategory: vehicleCategory
                }
            }],
            where: {
                [Op.and]: [
                    {
                        [Op.not]: {
                            [Op.or]: [
                                { StartTime: { [Op.gte]: endUTC } },
                                { EndTime: { [Op.lte]: startUTC } }
                            ]
                        }
                    }
                ]
            }
        });

        const reservedPlaceIds = reservations.map(r => r.ParkPlaceParkPlaceID);
        const availablePlaces = parkPlaces.filter(p => !reservedPlaceIds.includes(p.ParkPlaceID));

        if (availablePlaces.length === 0) {
            return res.status(409).json({ message: 'No free park places available for the selected time range.' });
        }

        const selectedPlace = availablePlaces[0];

        const start = new Date(startUTC);
        const end = new Date(endUTC);
        const durationMs = end - start;

        if (durationMs <= 0) {
            return res.status(400).json({ message: 'EndTime must be after StartTime.' });
        }

        // PriceTimeDuration (HH:mm:ss)
        const [hours, minutes, seconds] = selectedPlace.PriceTimeDuration.split(':').map(Number);
        const unitMs = ((hours * 60 + minutes) * 60 + seconds) * 1000;

        const units = Math.ceil(durationMs / unitMs);

        const currentPrice = new Decimal(selectedPlace.CurrentPrice);
        const totalPrice = currentPrice.mul(units); // decimal.js

        const reservation = await Reservation.create({
            DateAndTime: new Date().toISOString(),
            StartTime: startUTC,
            EndTime: endUTC,
            Status: 'active',
            VehicleVehicleID: VehicleID,
```

```
93.             ParkPlaceParkPlaceID: selectedPlace.ParkPlaceID
94.         });
95.
96.
97.         return res.status(201).json({
98.             message: 'Reservation created successfully.',
99.             reservationID: reservation.ReservationID,
100.             price: totalPrice.toFixed(2),
101.             currency: selectedPlace.Currency
102.         });
103.
104.     } catch (error) {
105.         console.error('Error creating reservation:', error);
106.         return res.status(500).json({ message: 'Internal server
error.' });
107.     }
108. }
```

Додаток Д

Функція зі скасування бронювання

```
1. /**
2.  * Skip reservation and handle refund with cancellation policy.
3.  */
4. async skipReservation(req, res) {
5.     try {
6.         const { reservationID } = req.params;
7.
8.         if (!reservationID) {
9.             return res.status(400).json({ message: 'ReservationID
parameter is required.' });
10.         }
11.
12.         // Fetch reservation
13.         const reservation = await
Reservation.findByPk(reservationID);
14.         if (!reservation) {
15.             return res.status(404).json({ message: 'Reservation
not found.' });
16.         }
17.
18.         if (reservation.Status !== 'active') {
19.             return res.status(409).json({ message: 'Only active
reservations can be skipped.' });
20.         }
21.
22.         // Fetch related park place and parking
23.         const parkPlace = await
ParkPlace.findByPk(reservation.ParkPlaceParkPlaceID);
24.         if (!parkPlace) {
25.             return res.status(404).json({ message: 'Associated
park place not found.' });
26.         }
27.
28.         const parking = await
Parking.findByPk(parkPlace.ParkingParkingID);
29.         if (!parking) {
30.             return res.status(404).json({ message: 'Associated
parking not found.' });
31.         }
32.
33.         // Get user ID through vehicle
34.         const vehicle = await
Vehicle.findByPk(reservation.VehicleVehicleID);
35.         if (!vehicle) {
36.             return res.status(404).json({ message: 'Vehicle
associated with reservation not found.' });
37.         }
38.
39.         const userId = vehicle.UserUserID;
40.
41.         // Time before reservation starts in hours
42.         const now = new Date();
```

```
43.         const timeToStart = (new Date(reservation.StartTime) -
now) / 3600000;
44.
45.         // Fetch original successful payment transaction
46.         const transaction = await Transaction.findOne({
47.             where: {
48.                 ReservationReservationID: reservationID,
49.                 UserUserID: userId,
50.                 Type: 'payment',
51.                 Status: 'PAID'
52.             }
53.         });
54.
55.         if (!transaction) {
56.             return res.status(404).json({ message: 'Original
payment transaction not found.' });
57.         }
58.
59.
60.         // Calculate refund and fee
61.         const originalAmount = new Decimal(transaction.Amount);
62.
63.         // Get cancellation policy
64.         const policy = await
cancellationPoliciesService.getWhereParkingAndTime(parking.ParkingID,
timeToStart);
65.         if (!policy) {
66.             await UserBalanceService.refundUser(userId,
originalAmount.toNumber());
67.
68.             // Mark reservation as skipped
69.             reservation.Status = 'skipped';
70.             await reservation.save();
71.             return res.status(200).json({
72.                 message: 'Reservation successfully skipped.
Refund processed.',
73.                 refundAmount: originalAmount.toString(),
74.                 feeAmount: 0.00
75.             });
76.         }
77.
78.         const cancelPercent = new
Decimal(policy.CancellationFeePercent);
79.
80.         const fee =
originalAmount.mul(cancelPercent.div(100)).toDecimalPlaces(2);
81.         const refund =
originalAmount.minus(fee).toDecimalPlaces(2);
82.
83.         // Refund to user via UserBalanceService
84.         await UserBalanceService.refundUser(userId,
refund.toNumber());
85.
86.         // Mark reservation as skipped
87.         reservation.Status = 'skipped';
88.         await reservation.save();
89.
90.         // Add cancellation fee to parking manager's balance
```

```
91.         const parkingManager = await ParkingManager.findOne({
92.             where: { ParkingParkingID: parking.ParkingID,
93.                      Role: 'owner'}
94.         });
95.         if (parkingManager) {
96.             const adminUserID = parkingManager.UserUserID;
97.
98.             const revenueTransaction = await
TransactionService.create(
99.                 fee.toNumber(),
100.                 transaction.Currency,
101.                 adminUserID,
102.                 'revenue',
103.                 'system',
104.                 'PAID',
105.                 `Cancellation fee from reservation
${reservationID}`
106.             );
107.         }
108.
109.         return res.status(200).json({
110.             message: 'Reservation successfully skipped. Refund
processed.',
111.             refundAmount: refund.toString(),
112.             feeAmount: fee.toString()
113.         });
114.
115.     } catch (error) {
116.         console.error('Error skipping reservation:', error);
117.         return res.status(500).json({ message: 'Internal server
error.' });
118.     }
119. }
```

Додаток Е

LiqPay API

```
1. // Function to generate LiqPay signature
2. function generateSignature(data) {
3.     return crypto.createHash('sha1')
4.         .update(LIQPAY_PRIVATE_KEY + data + LIQPAY_PRIVATE_KEY)
5.         .digest('base64');
6. }
7.
8. // Create a payment order
9. const createLiqPayOrder = async (currency, amount, description,
orderId) => {
10.     try {
11.         console.log('Creating link in LiqPay');
12.
13.         const requestData = {
14.             public_key: LIQPAY_PUBLIC_KEY,
15.             version: 3,
16.             action: 'pay',
17.             amount: amount,
18.             currency: currency,
19.             description: description,
20.             order_id: orderId,
21.             language: "ru",
22.             result_url: 'https://localhost:5192/complete-order',
23.             server_url: 'https://ba8e-141-105-139-174.ngrok-
free.app/webhook',
24.             sandbox: 1 // Remove for production
25.         };
26.
27.         const data =
Buffer.from(JSON.stringify(requestData)).toString('base64');
28.         const signature = generateSignature(data);
29.
30.         return
`https://www.liqpay.ua/api/3/checkout?data=${encodeURIComponent(data)
}&signature=${encodeURIComponent(signature)}`;
31.     } catch (error) {
32.         console.error('LiqPay API Error:', error);
33.         throw error;
34.     }
35. };
36.
37. // Confirm Webhook by LiqPay
38. const handleLiqPayWebhook = async (req, res) => {
39.     try {
40.         if (!req.body.data || !req.body.signature) {
41.             return res.status(400).json({ error: 'Missing data or
signature in request' });
42.         }
43.         const { data, signature } = req.body;
44.         const expectedSignature = generateSignature(data);
45.
46.         if (signature !== expectedSignature) {
```

```
47.             return res.status(400).json({ error: 'Invalid
signature' });
48.         }
49.
50.         const responseData = JSON.parse(Buffer.from(data,
'base64').toString('utf-8'));
51.
52.         if (responseData.status === 'success' ||
responseData.status === 'sandbox') {
53.             await Transaction.update(
54.                 { Status: 'success' },
55.                 { where: { TransactionID: responseData.order_id }
}
56.             );
57.             return res.status(200).json({ message: 'Payment
successful' });
58.         }
59.         return res.status(400).json({ error: 'Payment failed' });
60.
61.     } catch (error) {
62.         console.error('❌ Error Webhook:', error);
63.         return res.status(500).json({ error: 'Internal server
error' });
64.     }
65. };
66.
67.
68. module.exports = { createLiqPayOrder, handleLiqPayWebhook };
```

Додаток Ж

PayPal API

```
async function generateAccessToken() {
    const response = await axios({
        url: `${process.env.PAYPAL_BASE_URL}/v1/oauth2/token`,
        method: 'post',
        data: 'grant_type=client_credentials',
        auth: {
            username: process.env.PAYPAL_CLIENT_ID,
            password: process.env.PAYPAL_SECRET_KEY
        }
    })
    console.log(response)
    return response.data.access_token
}

const createOrder = async (currency, amount, description) => {
    try {
        const accessToken = await generateAccessToken()

        const response = await axios({
            url:
`${process.env.PAYPAL_BASE_URL}/v2/checkout/orders`,
            method: 'post',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${accessToken}`
            },
            data: {
                intent: 'CAPTURE',
                purchase_units: [{
                    items: [{
                        name: 'Bachelor project parking system',
                        description: description,
                        quantity: 1,
                        unit_amount: {
                            currency_code: currency,
                            value: amount
                        }
                    }],
                    amount: {
                        currency_code: currency,
                        value: amount,
                        breakdown: {
                            item_total: {
                                currency_code: currency,
                                value: amount
                            }
                        }
                    }
                }],
                application_context: {
                    return_url: 'https://localhost:5192/complete-
order',
```

```
                    cancel_url: 'https://localhost:5192/cancel-
order',
                    brand_name: 'Park4Flow'
                }
            }
        })

        return response.data.links.find(link => link.rel ===
'approve').href
    } catch (error) {
        console.error('PayPal API Error:', error.response ?
error.response.data : error.message)
    }
}

const captureOrder = async (orderID) => {
    try {
        const accessToken = await generateAccessToken()

        const response = await axios({
            url:
`${process.env.PAYPAL_BASE_URL}/v2/checkout/orders/${orderID}/capture
`,
            method: 'post',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${accessToken}`
            }
        })

        console.log('Success pay:', response.data)
        return response.data
    } catch (error) {
        console.error('Pay error:', error.response ?
error.response.data : error.message)
        throw error
    }
}
```

Додаток И

Юніт-тестування бізнес-логіки серверної частини

Таблиця И.1 – Тест-кейс №1 (таблиця виконана самостійно)

Ідентифікатор тесту		Тест-кейс №1				
Назва		Тестування бізнес логіки серверної частини				
Власник		Радько Марк Максимович				
Дата проведення		21.04.2025				
Мета тестування		Перевірити корректність роботи математичних методів бізнес логіки серверної частини, перевірити стабільність роботи, обробку помилок				
№	Метод	Тест	Статус	Очікуваний результат	Фактичний результат	Коментар
1	CommissionService .getCommissionPercent	Читання комісії з файлу settings.json	Пройдено	Читає комісію з файлу (commission_percent : 7.5)	Отримано 7.5	-
		Запис нового значення	Пройдено	Записує новий відсоток (15.2) у файл	Файл оновлено з { commission_percent: 15.2 }	-
		Перевірка валідації	Пройдено	Викидає помилку при <0 або >100	Помилки виникають (-5, 150)	-
2	CommissionService	Основний розрахунок	Пройдено	Сума 200 з комісією 10% = 220.00	Отримано 220.00	-

Продовження таблиці И.1

№	Метод	Тест	Статус	Очікуваний результат	Фактичний результат	Коментар
	.getCommissionPercent	Округлення	Не пройдено, виправлено	Сума 123.456 з комісією 7.5% ≈ 132.71	Отримано 132.72	Відбувається округлення за правилами математики, потребує виправленню.
3	UserDebtService .createUserDebt	Створення боргу, якщо баланс негативний і немає існуючого боргу	Пройдено	Виклик UserDebt.create з правильними даними (UserUserID: 1, Amount: '5.00' тощо)	Об'єкт створено з вірними полями	-
		Не створюється борг, якщо баланс позитивний	Пройдено	UserDebt.create не викликається	Метод не викликав створення запису	-
		Не створюється борг, якщо вже є несплачений борг	Не пройдено, виправлено	UserDebt.create не викликається	Створення дублікату	Створюється дублікат через відсутню перевірку, потребує виправлення

Продовження таблиці И.1

№	Метод	Тест	Статус	Очікуваний результат	Фактичний результат	Коментар
4	UserDebtService.setIsRepaid	Позначає борг як сплачений, якщо баланс ≥ 0	Пройдено	Виклик UserDebt.update з { isRepaid: true }	Запис оновлено коректно	-
		Не оновлює борг, якщо баланс негативний	Пройдено	UserDebt.update не викликається	Борг залишається несплаченим	-
		Обробляє відсутній userID	Пройдено	UserDebt.update не викликається	Помилка не виникає, функція завершується без дій	-
5	UserDebtService.overdueDebt	Блокує користувача, якщо борг прострочено і баланс негативний	Пройдено	Виклик User.update з { IsBanned: true }	Користувача заблоковано	-
		Не блокує, якщо баланс став невід'ємним	Пройдено	User.update не викликається	Користувач не заблокований	-
		Блокує користувача, якщо баланс не знайдено (негативний сценарій)	Пройдено	Виклик User.update з { IsBanned: true }	Користувача заблоковано (через відсутність балансу)	-
6	UserBalanceService.chargeUser	Стягує кошти та створює транзакцію типу payment	Пройдено	Баланс зменшено на 30, транзакція створена з типом payment	newBalance: '70.00', тип транзакції вірний	-

Продовження таблиці И.1

№	Метод	Тест	Статус	Очікуваний результат	Фактичний результат	Коментар
		Обробляє перехід у негативний баланс (транзакція типу debt)	Пройдено	Баланс стає -20.00, транзакція створена з типом debt	Виклик TransactionService.create з правильними параметрами	-
		Відхиляє негативну суму (amount <= 0)	Пройдено	Помилка Amount to charge must be positive	Помилка Amount to charge must be positive в консолі	-
		Відхиляє запит для неіснуючого балансу	Пройдено	Помилка User balance not found	User balance not found	-
7	UserBalanceService .topUpUserBalance	Поповнює баланс та створює транзакцію типу deposit	Пройдено	Баланс збільшено на 25, транзакція створена з джерелом paypal	newBalance: '75.00', тип транзакції вірний	-
		Відхиляє негативну суму (amount <= 0)	Пройдено	Помилка Top-up amount must be positive	Top-up amount must be positive	-
		Відхиляє запит для неіснуючого балансу	Пройдено	Помилка User balance not found.	UserBalance.findOne повертає null	-
8	UserBalanceService.refundUser	Повертає кошти та створює транзакцію типу refund	Пройдено	Баланс збільшено на 15, транзакція	newBalance: '35.00', тип	-

Продовження таблиці И.1

№	Метод	Тест	Статус	Очікуваний результат	Фактичний результат	Коментар
				створена з типом refund	транзакції вірний	-
		Відхиляє негативну суму (amount <= 0)	Пройдено	Помилка Refund amount must be positive	Refund amount must be positive	-
		Відхиляє запит для неіснуючого балансу	Пройдено	Помилка User balance not found.	UserBalance.findOne повертає null	-
9	UserBalanceService .getUserBalance	Повертає баланс та валюту	Пройдено	Об'єкт з полями balance (тип Decimal) та currency	{ balance: new Decimal('42.50'), currency: 'EUR' }	-
		Відхиляє запит для неіснуючого балансу	Пройдено	Помилка User balance not found	UserBalance.findOne повертає null	-
10	UserBalanceService .calculateTotalPaymentWithCommission	Делегує розрахунок комісії до CommissionService	Пройдено	Виклик CommissionService.calculateTotalWithCommission(100)	Повернено значення 107.5 (якщо комісія 7.5%)	-
11	TransactionService.create	Повинен створювати нову транзакцію з правильними полями	Пройдено	Об'єкт транзакції створено з усіма правильними полями, включно з датою	Transaction.create викликано з об'єктом, що містить усі очікувані поля	-

Продовження таблиці И.1

№	Метод	Тест	Статус	Очікуваний результат	Фактичний результат	Коментар
12	TransactionService.update	Повинен оновлювати статус транзакції	Пройдено	Оновлення статусу транзакції, повертається масив [1] як кількість оновлених рядків	Transaction.update викликано з коректним об’єктом і умовою TransactionID: 42	-
13	SubscriptionService .extendOrCreateSubscription	Повинен продовжити наявну підписку	Пройдено	Знаходить активну підписку користувача, додає дні до кінця, оновлює SubPayID та активує підписку	Повертає оновлений об’єкт підписки, викликає save()	-
		Повинен створити нову підписку, якщо не існує	Пройдено	Створює новий об’єкт підписки з тривалістю з тарифу, встановлює як активну	Subscription.create викликано з коректними параметрами, повертає створену підписку	-
14	SubscriptionService .deactivateExpiredSubscriptions	Повинен деактивувати прострочені підписки	Пройдено	Встановлює isActive: false для всіх підписок, де SubscriptionEnd <	Subscription.update викликано з правильним фільтром, повертає [1]	-

Продовження таблиці И.1

№	Метод	Тест	Статус	Очікуваний результат	Фактичний результат	Коментар
				поточної дати і isActive = true		
		Повинен логувати помилку у разі невдачі	Пройдено	У разі помилки при оновленні — запис у console.error	console.error викликано з повідомленням про помилку і об’єктом помилки	-
15	SubPayService.create	Повинен створити новий запис про оплату підписки	Пройдено	SubPay.create викликається з усіма полями: UserUserID, TariffPlanID, Amount, Currency, DateAndTime, PayPurpose	Метод викликано з правильними параметрами, повернуто об'єкт створеної оплати	-
		Повинен викинути помилку, якщо SubPay.create завершиться невдачею	Пройдено	У разі помилки база повертає виняток, сервіс повинен його пробросити	SubPay.create викликав throw, сервіс також викинув помилку	-
16	SalesPoliciesService.getApplicableSale	Повинен викинути помилку, якщо не передано UserID або ParkingID	Пройдено	Метод має кидати помилку з повідомленням "UserID and	Помилка викинута відповідно до очікувань	-

Продовження таблиці И.1

№	Метод	Тест	Статус	Очікуваний результат	Фактичний результат	Коментар
				ParkingID are required parameters."		
		Повинен повернути 0, якщо немає жодної знижки	Пройдено	Якщо не знайдено персональної або загальної знижки — повертається 0	Метод повертає 0	-
		Повинен повернути персональну знижку, якщо вона існує, а загальної немає	Пройдено	Повертається значення SalePercent з персональної політики	Метод повертає 10 (як очікувалося)	-
		Повинен повернути загальну знижку, якщо вона існує, а персональної немає	Пройдено	Повертається значення SalePercent із загальної політики	Метод повертає 20 (як очікувалося)	-
		Повинен повернути комбіновану знижку, якщо існують обидві (персональна та загальна)	Пройдено	Комбінована знижка за формулою: 1 - (1 - personal) * (1 - general) → округлення до цілих	Метод повертає 28 (10% персональна + 20% загальна → 28% комбінована)	Комбінація через мультиплікативний підхід
		Повинен логувати та пробросити помилки з бази	Пройдено	Якщо SalesPolicies.findOne кидає помилку, вона має бути зловлена,	Помилка "DB error" була зловлена,	-

Продовження таблиці И.1

№	Метод	Тест	Статус	Очікуваний результат	Фактичний результат	Коментар
				залогована, а потім проброшена повторно	залогована й проброшена	
17	ParkPlaceService .setIsTakenTrue	Повертає 404, якщо паркомісце не знайдено	Пройдено	res.status(404) і { error: 'ParkPlace not found' }	Повернуто відповідь з кодом 404 і відповідним повідомленням	-
		Оновлює IsTaken: true, оновлює DemandFactor	Пройдено	Оновлення стану, виклик updateDemandFactor, повернення 200	Всі оновлення і перерахунки виконані, повернуто статус 200	-
		Обробляє помилки всередині (наприклад, БД недоступна)	Пройдено	Викинута помилка перехоплена, повернено 500 і опис	Правильна обробка винятку	-
18	ParkPlaceService .setIsTakenFalse	Оновлює IsTaken: false, викликає перерахунок коефіцієнта попиту	Пройдено	Оновлення стану, виклик updateDemandFactor, повернення 200	Стан змінено, DemandFactor оновлено	-
19	ParkPlaceService	Правильно оновлює DemandFactor та	Пройдено	Коефіцієнт = 1 + (taken / total), ціна = base * коефіцієнт	Успішно застосовано	Обробка з Sequelize.literal

Продовження таблиці И.1

№	Метод	Тест	Статус	Очікуваний результат	Фактичний результат	Коментар
	.updateDemandFactor	CurrentPrice (звичайна ситуація)			оновлення до обох таблиць	
		Обробляє випадок поділу на 0 (немає паркомісць)	Пройдено	Якщо total = 0, DemandFactor має бути 1.0	DemandFactor зберіг значення 1.0	-
20	arkingBalanceService.addToParkingBalance	Додавання коштів та створення транзакції revenue	Пройдено	Баланс оновлюється, транзакція створюється, повертається новий баланс	Баланс оновився до '150.00', транзакція створена успішно	-
		Відсутній запис балансу	Пройдено	Повертається помилка Parking balance not found.	Отримано помилку Parking balance not found.	-
		Сума ≤ 0	Пройдено	Повертається помилка Amount must be a positive number.	Отримано помилку Amount must be a positive number.	-
21	ParkingBalanceService.deductFromParkingBalance	Зняття коштів та створення транзакції payout	Пройдено	Баланс оновлюється, транзакція створюється,	Баланс оновився до '70.00', транзакція	-

Продовження таблиці И.1

№	Метод	Тест	Статус	Очікуваний результат	Фактичний результат	Коментар
				повертається новий баланс	створена успішно	
		Відсутній власник паркінгу	Пройдено	Повертається помилка Parking owner (UserUserID) not found. Cannot create transaction.	Отримано відповідну помилку	-
22	ParkingBalanceService.getParkingBalance	Отримання балансу з валютою	Пройдено	Повертається баланс з округленням до двох знаків після коми та валюта	Отримано balance: 250.56, currency: 'UAH'	-
		Відсутній баланс	Пройдено	Повертається помилка Parking balance not found.	Отримано помилку Parking balance not found.	-
23	CancellationPoliciesService.getWhereParkingAndTime	Повернення правильного відсотка штрафу при знайденій політиці	Пройдено	Повертає { CancellationFeePercent: 20 }	Отримано { CancellationFeePercent: 20 }	-
		Логування та повернення 0% штрафу, якщо політика не знайдена	Не пройдено, виправлено	Повертає { CancellationFeePercent: 0 }	Отримано undefined	Метод повертає undefined повністю, замість

Продовження таблиці И.1

№	Метод	Тест	Статус	Очікуваний результат	Фактичний результат	Коментар
						об'єкта { CancellationFeePercent: 0 }, потребує виправлення
		Логування помилки при винятку з бази даних	Не пройдено, виправлено	Логує помилку, повертає { CancellationFeePercent: 0 }	Помилка залогована, отримано undefined	Метод повертає undefined повністю, замість об'єкта { CancellationFeePercent: 0 }, потребує виправлення
24	BonusService.payBonuses	Застосування бонусів і повернення правильної суми	Пройдено	10 бонусів по 1 USD, знижка 10 USD, повертає 90	Повернуто 90	Бонуси зменшено до 0
		Невалідна сума (0 або менше)	Пройдено	Помилка "Failed to apply bonus discount"	Помилка "Failed to apply bonus discount"	-
		Користувача не знайдено	Пройдено	Помилка "Failed to apply bonus discount	Помилка "Failed to apply bonus discount	-

Продовження таблиці И.1

№	Метод	Тест	Статус	Очікуваний результат	Фактичний результат	Коментар
		Відсутній запис про вартість бонуса	Пройдено	Помилка "Failed to apply bonus discount"	Помилка "Failed to apply bonus discount"	-
		Недостатньо коштів після знижки	Пройдено	Помилка "Failed to apply bonus discount"	Помилка "Failed to apply bonus discount"	-
		0 бонусів – не змінювати баланс	Пройдено	Повертає повну суму (100), не викликає save()	Повертає 100	Бонуси не використовуються
25	AEScipher. encryptData	Шифрування та розшифрування валідного рядка	Пройдено	Повертається зашифроване значення, яке успішно розшифровується до початкового	Рядок шифрується та успішно розшифровується	-
		Обробка null та undefined	Пройдено	Повертається null	Повертається null	-
26	AEScipher. decryptData	Обробка null та undefined	Пройдено	Повертається null	Повертається null	-
		Шифрування різних рядків	Пройдено	Результати шифрування різних рядків відрізняються	Зашифровані рядки не співпадають	-
		Обробка пошкоджених або некоректних даних	Пройдено	Генерується помилка	Викидається помилка	-

Додаток К

Тестування REST API через середовище Postman

Таблиця К.1 – Тест-кейс №1 (таблиця виконана самостійно)

Ідентифікатор тесту		Тест-кейс №1				
Назва		Postman тестування серверних ендпоїнтів				
Власник		Радько Марк Максимович				
Дата проведення		25.04.2025				
Мета тестування		Перевірити коректність роботи серверних контроллерів, маршрутів, фільтрів доступу				
Категорія	Метод	Ендпоінт	Статус	Очікуваний результат	Фактичний результат	Примітки
User	POST	/api/users/register	Пройдено	Користувач успішно зареєстрований	201 Created	—
User	POST	/api/users/login	Пройдено	Повернуто токен авторизації	200 OK	—
User	PUT	/api/users/ban	Пройдено	Користувача заблоковано	200 OK	—
User	PUT	/api/users/unban	Пройдено	Користувача розблоковано	200 OK	—
User	PUT	/api/users/update	Помилка, виправлено	Дані оновлено	400 Bad Request	Некоректний формат дати народження (валидація на сервері)
User	GET	/api/users/details	Пройдено	Отримано деталі користувача	200 OK	—

Продовження таблиці К.1

Категорія	Метод	Ендпоінт	Статус	Очікуваний результат	Фактичний результат	Примітки
User	POST	/api/users/pass/code	Пройдено	Код підтвердження надіслано	200 OK	—
User	POST	/api/users/pass/reset	Пройдено	Пароль змінено	200 OK	—
User	GET	/api/users/check	Пройдено	Перевірка токена	200 OK	—
User	GET	/api/users/email/:email	Пройдено	Користувача знайдено	200 OK	—
User	DELETE	/api/users/delete	Помилка, виправлено	Користувача видалено	500 Internal Server Error	Видалення пов’язано з FK обмеженням (зв’язані таблиці: vehicles, reservations)
Vehicle	POST	/api/vehicles/create	Пройдено	Транспорт створено	201 Created	—
Vehicle	PUT	/api/vehicles/update/:id	Пройдено	Дані оновлено	200 OK	—
Vehicle	DELETE	/api/vehicles/delete/:id	Пройдено	Транспорт видалено	200 OK	—
Vehicle	GET	/api/vehicles/user	Пройдено	Отримано список	200 OK	—
TariffPlan	POST	/api/tariffPlans/create	Пройдено	Тариф створено	201 Created	—
TariffPlan	PUT	/api/tariffPlans/update	Пройдено	Дані оновлено	200 OK	—
TariffPlan	DELETE	/api/tariffPlans/delete	Пройдено	Тариф видалено	200 OK	—

Продовження таблиці К.1

Категорія	Метод	Ендпоінт	Статус	Очікуваний результат	Фактичний результат	Примітки
TariffPlan	GET	/api/tariffPlans/currency/:currency	Пройдено	Отримано список тарифів	200 OK	—
Subscription	POST	/api/subscriptions/paypal/create	Пройдено	Посилання створено	200 OK	—
Subscription	POST	/api/subscriptions/paypal/confirm	Пройдено	Підписка активована	200 OK	—
Subscription	POST	/api/subscriptions/liqpay/create	Помилка	Посилання створено	502 Bad Gateway	Збої зі сторони LiqPay API (некоректна відповідь від провайдера)
Subscription	POST	/api/subscriptions/liqpay/confirm	Пройдено	Підписка активована	200 OK	—
SalesPolicies	POST	/api/salesPolicies/create	Пройдено	Політика створена	201 Created	—
SalesPolicies	DELETE	/api/salesPolicies/delete/general	Пройдено	Загальна політика видалена	200 OK	—
SalesPolicies	DELETE	/api/salesPolicies/delete/personal/:email	Пройдено	Персональна політика видалена	200 OK	—
Reservations	POST	/api/reservations/create	Пройдено	Резервація створена	201 Created	—
Reservations	PUT	/api/reservations/skip/:id	Пройдено	Резервація скасована	200 OK	—
Reservations	GET	/api/reservations/user	Пройдено	Отримано список резервацій	200 OK	—

Продовження таблиці К.1

Категорія	Метод	Ендпоінт	Статус	Очікуваний результат	Фактичний результат	Примітки
Reservations	GET	/api/reservations/parking	Пройдено	Отримано статистику	200 OK	—
Payment	POST	/api/payments/paypal/create	Пройдено	Посилання створено	200 OK	—
Payment	POST	/api/payments/paypal/confirm	Пройдено	Оплату підтверджено	200 OK	—
Payment	POST	/api/payments/liqpay/create	Помилка	Посилання створено	504 Gateway Timeout	Сервер LiqPay не відповідає вчасно (таймаут 30с)
Payment	POST	/api/payments/liqpay/confirm	Пройдено	Оплату підтверджено	200 OK	—
Payment	POST	/api/payments/balance	Пройдено	Оплату виконано з балансу	200 OK	—
ParkPlace	POST	/api/parkPlaces/create	Пройдено	Місце створено	201 Created	—
ParkPlace	PUT	/api/parkPlaces/update/:id	Пройдено	Дані оновлено	200 OK	—
ParkPlace	DELETE	/api/parkPlaces/delete/:id	Пройдено	Місце видалено	200 OK	—
ParkPlace	GET	/api/parkPlaces/getInfo/:id	Пройдено	Дані отримано	200 OK	—
Parking	POST	/api/parkings/create	Пройдено	Паркінг створено	201 Created	—
Parking	PUT	/api/parkings/update	Пройдено	Паркінг оновлено	200 OK	—

Продовження таблиці К.1

Категорія	Метод	Ендпоінт	Статус	Очікуваний результат	Фактичний результат	Примітки
Parking	DELETE	/api/parkings/delete/:id	Пройдено	Паркінг видалено	200 OK	—
Parking	GET	/api/parkings/getStatistics	Пройдено	Дані отримано	200 OK	—
ParkingAction	POST	/api/parkingActions/user/start	Пройдено	Паркування розпочато	200 OK	—
ParkingAction	POST	/api/parkingActions/IOT/start	Пройдено	IOT паркування стартувало	200 OK	—
ParkingAction	POST	/api/parkingActions/user/stop	Пройдено	Паркування завершено	200 OK	—
ParkingAction	POST	/api/parkingActions/IOT/stop	Пройдено	IOT паркування зупинено	200 OK	—
OpenAI	POST	/api/openAI/analyze	Пройдено	Відповідь згенерована	200 OK	—
FavouriteParking	POST	/api/favoriteParkings/add	Пройдено	Додано в улюблені	200 OK	—
FavouriteParking	DELETE	/api/favoriteParkings/delete	Пройдено	Видалено з улюблених	200 OK	—
FavouriteParking	GET	/api/favoriteParkings/favouriteParkings/user	Пройдено	Список отримано	200 OK	—
Deposit	POST	/api/deposits/paypal/create	Пройдено	Посилання створено	200 OK	—
Deposit	POST	/api/deposits/paypal/confirm	Пройдено	Поповнення підтверджено	200 OK	—

Продовження таблиці К.1

Категорія	Метод	Ендпоінт	Статус	Очікуваний результат	Фактичний результат	Примітки
Deposit	POST	/api/deposits/liqpay/create	Помилка	Посилання створено	502 Bad Gateway	Проблеми на стороні LiqPay
Deposit	POST	/api/deposits/liqpay/confirm	Пройдено	Поповнення підтверджено	200 OK	—
CancellationPolicy	POST	/api/cancellationPolicies/create	Пройдено	Політика створена	201 Created	—
CancellationPolicy	PUT	/api/cancellationPolicies/update	Пройдено	Політика оновлена	200 OK	—
CancellationPolicy	DELETE	/api/cancellationPolicies/delete	Пройдено	Політика видалена	200 OK	—
BonusesCost	POST	/api/bonusesCosts/create	Пройдено	Бонус створено	201 Created	—
BonusesCost	PUT	/api/bonusesCosts/update/:id	Пройдено	Дані оновлено	200 OK	—
BonusesCost	DELETE	/api/bonusesCosts/delete/:id	Пройдено	Видалено	200 OK	—
BonusesCost	GET	/api/bonusesCosts/currency/:currency	Пройдено	Отримано список	200 OK	—

Додаток Л

Мануальне тестування функціональності веб-клієнта

Таблиця Л.1 – Тест-кейс №1 (таблиця виконана самостійно)

Ідентифікатор тесту		Тест-кейс №1				
Назва		Тестування функціоналу авторизації				
Власник		Радько Марк Максимович				
Дата проведення		01.06.2025				
Мета тестування		Перевірити коректність роботи інтерфейсу авторизації, обробку помилок, UX-логику, валідацію та інтеграцію з бекендом.				
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Відображення форми авторизації	Перехід на сторінку авторизації	Відображається форма з полями email та пароль	Форма відображається	Пройдено	-
2	Введення валідних даних	email: test@gmail.com, пароль: 12345678	Вхід успішний, перенаправлення на головну сторінку	Вхід виконано	Пройдено	-
3	Введення невалідного email	email: test, пароль: 12345678	Повідомлення про помилку в email	Помилка не з'являється, вхід не виконано	Не пройдено	Потребує виправлення
4	Введення неправильного пароля	email: test@gmail.com,	Повідомлення про неправильний логін або пароль	Помилка з'являється	Пройдено	-

Продовження таблиці Л.1

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
		пароль: wrongpass				
5	Введення порожніх полів	email: "", пароль: ""	Валідаційне повідомлення "Поле обов’язкове"	Помилки з'являються	Пройдено	-
6	Поведінка при відсутності інтернет-з’єднання	Вимкнений інтернет, спроба входу	Повідомлення про помилку мережі	Повідомлення відображається	Пройдено	-
7	Автоматичний редірект авторизованого користувача	Користувач уже авторизований	Перенаправлення на головну сторінку	Перенаправлення виконано	Пройдено	-
8	Кнопка "Показати пароль"	Натиснути на іконку ока	Пароль відображається у вигляді тексту	Пароль стає видимим	Пройдено	-
9	Посилання "Забули пароль?"	Клік на посилання	Перехід на сторінку відновлення паролю	Перехід виконується	Пройдено	-

Таблиця Л.2 – Тест-кейс №2 (таблиця виконана самостійно)

Ідентифікатор тесту		Тест-кейс №2				
Назва		Тестування функціоналу реєстрації				
Власник		Радько Марк Максимович				
Дата проведення		01.06.2025				
Мета тестування		Перевірити коректність роботи інтерфейсу реєстрації, валідацію полів, обробку помилок та інтеграцію з сервером.				
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Відображення форми реєстрації	Перехід на сторінку реєстрації	Відображається форма з усіма необхідними полями	Форма відображається	Пройдено	-
2	Введення коректних даних	Email, Пароль, Ім’я, Прізвище, Телефон, Валюта	Успішна реєстрація	Перехід на сторінку авторизації	Пройдено	-
3	Введення існуючого Email	Email, який вже зареєстрований	Повідомлення про те, що цей email вже зареєстрований	Повідомлення відображається	Пройдено	-
4	Пропуск одного обов’язкового поля	Всі поля заповнені, крім номера телефону	Повідомлення про необхідність заповнення усіх полів	Повідомлення відображається	Пройдено	-
5	Валідація порожньої форми	Всі поля порожні	Відображення повідомлень про обов’язковість кожного поля	Повідомлення відображається	Пройдено	-
6	Вибір валюти зі списку	Валюта: USD	Значення передається на сервер	Валюта збережена	Пройдено	-

Продовження таблиці Л.2

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
7	Повідомлення про успішну реєстрацію	Повне коректне заповнення форми	Повідомлення про успішну реєстрацію	Повідомлення з’являється	Пройдено	-
8	Перевірка перенаправлення після реєстрації	Реєстрація завершена	Перенаправлення на сторінку входу	Реалізовано коректно	Пройдено	-
9	Спроба повторно відправити форму (дубль запиту)	Подвійне натискання на кнопку "Зареєструватися"	Обробка дубля запобігає створенню декількох користувачів	Дублів не створено	Пройдено	Кнопка заблокована під час запиту, серверна частина блокує створення акаунту з використаним email
10	Посилання "Є акаунт?"	Клік на посилання	Перехід на сторінку авторизації	Перехід виконується	Пройдено	-
11	Поведінка при відсутності інтернету	Відключений інтернет, спроба реєстрації	Повідомлення про неможливість з’єднання	Відповідне повідомлення	Пройдено	-

Таблиця Л.3 – Тест-кейс №3 (таблиця виконана самостійно)

Ідентифікатор тесту		Тест-кейс №3				
Назва		Тестування функціоналу відновлення паролю				
Власник		Радько Марк Максимович				
Дата проведення		01.06.2025				
Мета тестування		Перевірити коректність відображення модального вікна, валідацію email та паролю, логіку коду підтвердження, а також інтеграцію з API.				
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Відображення модального вікна	Клік на кнопку "Забули пароль?"	Відкривається модальне вікно з полем для email	Вікно з’являється	Пройдено	-
2	Валідація порожнього поля email	Нічого не введено	Повідомлення: "Email is required"	Помилка показується	Пройдено	-
3	Введення невалідного email	invalid-email	Повідомлення: "Invalid email address"	Помилка показується, помилку обробляє серверна частина	Не пройдено	Потребує виправлення
4	Введення валідного email	user@example.com	Надсилання коду, toast-повідомлення успіху, перехід на крок 2	Перехід виконано	Пройдено	-
5	Відображення поля для вводу коду (6 цифр)		З’являється 6 полів для вводу коду	Відображено правильно	Пройдено	-

Продовження таблиці Л.3

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
6	Автоперехід між полями коду	Вводимо послідовно цифри	Курсор автоматично переходить до наступного поля	Перехід працює	Пройдено	-
7	Backspace веде на попереднє поле	Натискаємо Backspace на порожньому полі	Переміщення фокусу на попереднє поле	Перехід працює	Пройдено	-
8	Валідація неповного коду	Введено лише 5 цифр	Toast-помилка: "Please enter the full 6-digit code"	Помилка з’являється	Пройдено	-
9	Валідація порожнього нового пароля	Поле NewPassword порожнє	Повідомлення: "Password is required"	Помилка з’являється	Пройдено	-
10	Підтвердження пароля не збігається	Паролі: 12345678 і 87654321	Повідомлення: "Passwords do not match"	Помилка з’являється	Пройдено	-
11	Введено коректні дані	Email, код (6 цифр), 2 однакові паролі	Toast "Password successfully reset", вікно закривається	Пароль оновлено, вікно закрите	Пройдено	-
12	Введення неправильного коду	Невірний код: 000000	Toast-помилка з повідомленням сервера	Отримано помилку	Пройдено	-

Продовження таблиці Л.3

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
13	Перевірка toast-повідомлень	Успішна і неуспішна відправка	Toast-повідомлення відображаються зверху	Toast працює	Пройдено	-
14	Поведінка без інтернету	Вимкнений інтернет при надсиланні форми	Toast-помилка "Network error"	Помилка відображена	Пройдено	-

Таблиця Л.4 – Тест-кейс №4 (таблиця виконана самостійно)

Ідентифікатор тесту	Тест-кейс №4					
Назва	Тестування функціоналу сторінки профілю					
Власник	Радько Марк Максимович					
Дата проведення	01.06.2025					
Мета тестування	Перевірити коректність відображення персональних даних, редагування профілю, поповнення балансу, керування транспортними засобами, збереження вибраного авто та відображення модальних вікон.					
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Завантаження профілю	Валідний токен у localStorage	Дані користувача завантажуються і відображаються у відповідних полях	Дані завантажені, поля заповнені	Пройдено	-

Продовження таблиці Л.4

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
2	Відображення кнопки "Top Up"	Наявність даних користувача	Кнопка відображається у секції балансу	Кнопка з’являється і викликає модальне вікно поповнення	Пройдено	-
3	Зміна імені та збереження	Зміна значення поля First Name	З’являється кнопка "Save Changes", після натискання дані зберігаються	Кнопка з’являється, дані зберігаються, “Profile updated successfully”	Пройдено	-
4	Відображення транспортних засобів	Користувач має додані авто	Виводиться список машин у вигляді карток з діями “Edit” та “Delete”	Список відображається, авто можна вибрати, відмітка про вибране авто працює	Пройдено	-
5	Видалення транспортного засобу	Клік “Delete” + підтвердження	Автомобіль видаляється, список оновлюється	Видалення працює, ID оновлюється у localStorage	Пройдено	-
6	Додавання транспортного засобу	Клік на кнопку “Add Vehicle”	Перенаправлення на сторінку додавання авто	Відкривається модальне окно для реєстарції ТС	Пройдено	-
7	Відкриття модального вікна редагування авто	Клік по кнопці “Edit”	Відкривається VehicleModal з даними авто	Модальне вікно відкривається, після редагування дані оновлюються	Пройдено	-

Продовження таблиці Л.4

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
8	Обробка зображень авто	Зображення не завантажується	Виводиться дефолтне зображення /placeholder-vehicle.png	Зображення підставляється	Пройдено	-
9	Валідація відсутності токена	В localStorage відсутній jwtToken	Дані не завантажуються, жодного рендеру профілю	Профіль не рендериться, помилки не виникає	Пройдено	-
10	Повернення до збереженого вибраного авто	В localStorage є валідний selectedVehicleID	Авто з цим ID має бути виділене, інші — неактивні	Виділення працює, клас selected застосовується	Пройдено	-
11	Обробка невалідного selectedVehicleID	В localStorage збережений ID, якого вже не існує	Заміна на перше авто в списку, оновлення selectedVehicleID в localStorage	Значення оновлюється, авто виділяється	Пройдено	-
12	Відображення повідомлення про відсутність авто	У користувача немає авто	Виводиться повідомлення “No vehicles added yet” і кнопка для додавання	Повідомлення відображається	Пройдено	-

Таблиця Л.5 – Тест-кейс №5 (таблиця виконана самостійно)

Ідентифікатор тесту		Тест-кейс №5				
Назва		Тестування модального вікна додавання/редагування транспортного засобу				
Власник		Радько Марк Максимович				
Дата проведення		02.06.2025				
Мета тестування		Перевірити коректність відображення модального вікна, валідацію полів, завантаження зображення, логіку додавання та редагування транспортного засобу, а також інтеграцію з API				
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Відкриття модального вікна	Клік на кнопку "Add Vehicle"	Відображається форма з порожніми полями	Форма відображається	Пройдено	-
2	Валідація при пустих полях	Натиснути "Add" без заповнення	Повідомлення про обов’язкові поля	Повідомлення з’явились	Пройдено	-
3	Завантаження фото	Вибір файлу .jpg	Прев’ю зображення відображається	Зображення з'явилось	Пройдено	-
4	Введення некоректного номера авто	"12345678"	Помилка або попередження	Немає перевірки формату	Частково пройдено	Формат номера авто не перевіряється
5	Успішне додавання авто	Валідні значення всіх полів + фото	Повідомлення "Vehicle added!"	Повідомлення з’явилось	Пройдено	-
6	Автоматичне закриття	Очікування після додавання	Вікно закривається через 2с	Закрилось через 2с	Пройдено	-
7	Редагування існуючого авто	Відкрити з переданим ТС	Поля заповнені з даних	Значення заповнені	Пройдено	-

Продовження таблиці Л.5

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
8	Редагування без зміни фото	Змінити модель авто	Успішне оновлення без фото	Дані оновились	Пройдено	-
9	Помилка API	Відключений сервер	Вивід помилки під формою	Повідомлення з’явилось	Пройдено	-
10	Подвійне натискання на Add	Двічі натиснути кнопку	Повторне натискання блокується, показано спінер	Повтор не можливий, є спінер	Пройдено	-

Таблиця Л.6 – Тест-кейс №6 (таблиця виконана самостійно)

Ідентифікатор тесту	Тест-кейс №6					
Назва	Тестування сторінки перегляду бронювань користувача					
Власник	Радько Марк Максимович					
Дата проведення	02.06.2025					
Мета тестування	Перевірити коректність завантаження бронювань, відображення карток, фільтрації, сортування, пагінації, перегляду деталей, та скасування активного бронювання з обробкою помилок.					
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Завантаження сторінки бронювань	Вхід на сторінку з авторизованим токеном	Виводиться список бронювань користувача або повідомлення "No reservations found"	Дані завантажились і відобразились у вигляді карток	Пройдено	-

Продовження таблиці Л.6

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
2	Відображення активних бронювань	Статус фільтра: active	Виводяться тільки бронювання зі статусом active	Показані лише активні бронювання	Пройдено	-
3	Відображення неактивних бронювань	Статус фільтра: inactive	Виводяться тільки бронювання зі статусом inactive	Показані лише неактивні бронювання	Пройдено	-
4	Сортування за датою	Натиснути "Sort by Date"	Змінюється порядок відображення (asc ↔ desc)	Порядок змінюється відповідно до вибраного напряму	Пройдено	-
5	Перехід між сторінками	Натискання кнопок "Previous", "Next", номери сторінок	Відображаються відповідні сторінки бронювань	Переходи працюють, кнопки блокуються при крайніх значеннях	Пройдено	-
6	Перегляд деталей бронювання	Клік на картку бронювання	Відкривається модальне вікно з детальною інформацією	Модальне вікно відкривається з усіма деталями	Пройдено	-
7	Скасування активного бронювання	Клік по "Cancel" → Підтвердження	Бронювання скасовується, з’являється повідомлення про повернення або відсутність повернення	Скасування проходить, з’являється повідомлення з сумою повернення та комісією	Пройдено	-

Продовження таблиці Л.6

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
8	Обробка помилки скасування	Імітація відмови API або скасування без інтернету	Виводиться повідомлення про помилку скасування	Повідомлення про помилку з’являється з іконкою	Пройдено	-
9	Порожній список активних бронювань	Видалити всі активні та перезавантажити	Виводиться повідомлення “No active reservations found” з кнопкою повернення до активних	Повідомлення з’являється, кнопка працює	Пройдено	-
10	Оновлення списку бронювань	Натиснути кнопку “Refresh”	Відбувається повторний запит до API, оновлення списку	Дані оновлюються, спінер завантаження зникає після завершення	Пройдено	-

Таблиця Л.7 – Тест-кейс №7 (таблиця виконана самостійно)

Ідентифікатор тесту	Тест-кейс №7
Назва	Тестування функціоналу адміністратора з керування тарифними планами
Власник	Радько Марк Максимович
Дата проведення	02.06.2025
Мета тестування	Перевірити завантаження, фільтрацію, створення, редагування, видалення тарифних планів, роботу модального вікна та взаємодію з API.

Продовження таблиці Л.7

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Завантаження тарифних планів	Відкриття сторінки	Виводиться список тарифів або повідомлення “No tariff plans found”	Список успішно завантажено	Пройдено	-
2	Обробка помилки при завантаженні	Симуляція помилки API	Виводиться повідомлення про помилку завантаження	Виведено alert з повідомленням	Пройдено	Перевірено через штучну помилку
3	Фільтрація тарифів за валютою	Обрати валюту USD у фільтрі	Виводяться лише тарифні плани з валютою USD	Відфільтровані плани відповідають валюті	Пройдено	-
4	Відкриття модального вікна створення	Натискання на “Add New Plan”	Відкривається модальне вікно з формою для створення нового тарифного плану	Вікно з’являється	Пройдено	-
5	Відкриття модального вікна редагування	Натискання на кнопку Edit на одному з планів	Відкривається модальне вікно з попередньо заповненими даними	Вікно з’являється з даними тарифного плану	Пройдено	-
6	Видалення тарифного плану	Натискання кнопки Delete + підтвердження	План видаляється зі списку, з’являється	План зник, повідомлення “Tariff plan deleted	Пройдено	-

Продовження таблиці Л.7

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
			повідомлення про успіх	successfully” з’явилось		
7	Відмова від видалення тарифу	Натискання Delete + скасування підтвердження	План не видаляється	План залишився в списку	Пройдено	-
8	Робота індикатора завантаження	Очікування після дії (створення/видалення)	Показується анімація завантаження на час запиту	Спінер відображається під час запиту	Пройдено	-
9	Закриття модального вікна	Натискання на X або Cancel в модальному вікні	Модальне вікно закривається	Вікно зникає	Пройдено	-
10	Виведення повідомлень про помилки	Симуляція помилки видалення (наприклад: неправильний ID)	Виводиться червоне повідомлення про помилку	Повідомлення з текстом помилки виводиться в alert	Пройдено	-

Таблиця Л.8 – Тест-кейс №8 (таблиця виконана самостійно)

Ідентифікатор тесту	Тест-кейс №8
Назва	Тестування модального вікна створення/редагування тарифного плану
Власник	Радько Марк Максимович
Дата проведення	02.06.2025

Продовження таблиці Л.8

Мета тестування		Перевірити валідацію полів форми, створення та редагування тарифного плану, повідомлення про помилки, успішне завершення операцій та закриття модального вікна.				
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Відображення модального вікна створення	Клік на кнопку "Add New Plan"	Відкривається модальне вікно з порожніми полями	Модальне вікно відкривається, поля пусті	Пройдено	-
2	Валідація: пусті поля	Submit без заповнення	Виводяться повідомлення про помилку для всіх обов’язкових полів	Помилки з’являються під кожним порожнім полем	Пройдено	-
3	Валідація: негативна тривалість	Duration = -5	Повідомлення: "Please enter valid duration (at least 1 day)"	Повідомлення виведено	Пройдено	-
4	Валідація: ціна = 0	Price = 0	Повідомлення: "Please enter valid price (greater than 0)"	Повідомлення виведено	Пройдено	-
5	Валідація: не обрано валюту	Currency = ""	Повідомлення: "Please select currency"	Повідомлення виведено	Пройдено	-
6	Успішне створення тарифу	Duration = 30, Price = 9.99, Currency = USD, Type = Basic	План створюється, повідомлення про успіх, автоматичне закриття	План створено, повідомлення "Tariff plan created successfully!"	Пройдено	Модальне вікно закривається через 2 сек

Продовження таблиці Л.8

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
7	Відображення модального вікна редагування	Клік "Edit" для існуючого плану	Вікно відкривається з заповненими полями	Поля заповнені правильно	Пройдено	-
8	Успішне редагування тарифу	Змінено Duration з 30 на 45	Оновлення тарифу, повідомлення "Tariff plan updated successfully!", автозакриття	План оновлено, повідомлення виведено	Пройдено	-
9	Повідомлення про помилку при запиті	Імітація помилки API (відсутній токен/сервер недоступний)	Повідомлення про помилку "Failed to create/update tariff plan. Please try again."	Помилка обробляється, повідомлення виводиться	Пройдено	-
10	Закриття модального вручну	Клік по кнопці з X	Модальне вікно закривається	Закривається	Пройдено	-

Таблиця Л.9 – Тест-кейс №9 (таблиця виконана самостійно)

Ідентифікатор тесту	Тест-кейс №9
Назва	Тестування сторінки підписок
Власник	Радько Марк Максимович
Дата проведення	02.06.2025

Продовження таблиці Л.9

Мета тестування		Перевірити коректність відображення поточної підписки, завантаження тарифних планів, валідацію доступності кнопки "Register Parking", відкриття модального вікна оплати та інтеграцію з API.				
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Завантаження даних підписки	Користувач із активною підпискою	Виводиться тип тарифу, дата завершення та статус «Active»	Дані підписки коректно відображаються	Пройдено	-
2	Завантаження даних підписки	Користувач без підписки	Виводиться повідомлення про відсутність активної підписки	Повідомлення відображається	Пройдено	-
3	Відображення тарифних планів	Валюта: UAH (є відповідні плани)	Виводиться список тарифних планів з цінами, аналітикою і підтримкою	Плани відображаються згідно валюти	Пройдено	-
4	Відображення тарифних планів	Валюта: UZS (немає планів)	Виводиться повідомлення "No tariff plans available for your currency."	Повідомлення відображається	Пройдено	-
5	Перевірка відображення особливостей Base тарифу	Тариф типу "Base"	Відображаються "Base analytics" і "Base support"	Особливості коректно відображаються	Пройдено	-

Продовження таблиці Л.9

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
6	Перевірка відображення особливостей інших тарифів	Тариф типу "Extended"	Відображаються "Advanced analytics" і "Priority support"	Відображення правильне	Пройдено	-
7	Натискання кнопки "Subscribe"	Натискання на кнопку "Subscribe" у будь-якому тарифі	Відкривається модальне вікно оплати (PaymentModal)	Модальне вікно з’являється	Пройдено	-
8	Валідація кнопки "Register Parking"	Користувач без підписки	Кнопка неактивна, має сірий стиль	Кнопка неактивна	Пройдено	-
9	Валідація кнопки "Register Parking"	Користувач з активною підпискою	Кнопка активна, перенаправляє на /register-parking	Перенаправлення успішне	Пройдено	-
10	Поведінка при помилці завантаження даних	Імітація помилки API (відсутність токена або 500 error)	Виводиться повідомлення "Failed to load subscription data"	Повідомлення з’являється, помилка оброблена	Пройдено	-

Таблиця Л.10 – Тест-кейс №10 (таблиця виконана самостійно)

Ідентифікатор тесту	Тест-кейс №10

Продовження таблиці Л.10

Назва		Тестування модального вікна оплати				
Власник		Радько Марк Максимович				
Дата проведення		03.06.2025				
Мета тестування		Перевірити коректність відображення модального вікна для різних типів оплати (payment, subscription, deposit), вибір валюти та способу оплати, валідацію суми, логіку кнопок, відображення помилок та інтеграцію з API.				
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Відкриття модального вікна	Відкрити модал з типом payment	Відображається заголовок "Complete Payment", поля сума (з localStorage), вибір валюти, метод оплати, кнопки "Pay Now" та "Cancel"	Модальне вікно з'явилося коректно	Пройдено	-
2	Відкриття модального вікна	Відкрити модал з типом subscription	Заголовок "Subscribe", вибір валюти, метод оплати, кнопка "Proceed to Payment", кнопка "Cancel"	Модальне вікно відкрилося і відображається коректно	Пройдено	-
3	Відкриття модального вікна	Відкрити модал з типом deposit	Заголовок "Top Up Balance", поле для вводу суми, вибір валюти, метод оплати, кнопки	Модальне вікно відкрилося, поле суми є активним	Пройдено	-

Продовження таблиці Л.10

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
			"Proceed to Payment" і "Cancel"			
4	Валідація суми при депозиті	Ввести некоректну суму (0 або від'ємне) у полі суми (deposit)	Після натискання кнопки оплати з'являється помилка "Please enter a valid amount"	Помилка відображена	Пройдено	-
5	Вибір валюти та метод оплати	Обрати валюту UAH та спробувати вибрати PayPal	PayPal повинен бути недоступним для UAH, показується повідомлення "PayPal is not available for UAH payments"	Повідомлення відображається, вибір PayPal заблоковано	Пройдено	-
6	Вибір валюти та метод оплати	Обрати валюту USD або EUR, перевірити доступність всіх методів оплати	Можна вибрати LiqPay, PayPal або Balance (якщо не UAH)	Методи вибираються коректно	Пройдено	-
7	Оплата через LiqPay	Натиснути "Pay Now" з вибраним методом LiqPay	Відправляється запит на створення платежу, відкривається нове вікно з URL оплати	Нове вікно відкрилось, запит пройшов	Пройдено	-

Продовження таблиці Л.10

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
8	Оплата через PayPal	Натиснути "Pay Now" з вибраним методом PayPal	Відправляється запит на PayPal, відкривається нове вікно з URL оплати	Нове вікно відкрилось, запит пройшов	Пройдено	-
9	Оплата через баланс	Обрати метод "Balance" та натиснути оплату	Якщо успішно — переадресація на '/payment-result'; у разі помилки — відображення повідомлення про помилку	Переадресація або помилка відображені	Пройдено	-
10	Помилка при відсутності даних	Відсутній `amount` або `Currency` в localStorage при типі `payment`	Відображення помилки "Payment details not found. Please try again."	Помилка відображена	Пройдено	-
11	Заборонений спам-клік	Клік на кнопку "Pay Now" під час завантаження (loading=true)	Кнопки заблоковані, додаткові кліки ігноруються	Поведінка коректна	Пройдено	-
12	Закриття модального вікна	Натиснути на кнопку "Cancel" або хрестик	Модальне вікно закривається	Модальне вікно закривається	Пройдено	-
13	Відображення помилок	Виникає помилка під час оплати	Відображається відповідне повідомлення	Помилка відображена	Пройдено	-

Продовження таблиці Л.10

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
		(наприклад, проблеми з API)	помилки в червоному блоці			
14	Використання бонусів	Для типу `payment` відмітити чекбокс "Use available bonuses"	Стан чекбокса зберігається, значення передається в запит на оплату	Чекбокс працює коректно	Пройдено	-

Таблиця Л.11 – Тест-кейс №11 (таблиця виконана самостійно)

Ідентифікатор тесту	Тест-кейс №11					
Назва	Тестування функціоналу сторінки реєстрації паркінгу					
Власник	Радько Марк Максимович					
Дата проведення	03.06.2025					
Мета тестування	Перевірити коректність валідації обов’язкових полів, логіку інтеракції з мапою, завантаження фото, відображення повідомлень про помилки та успіх, а також інтеграцію з API.					
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Відображення сторінки	Перехід на /register-parking	Відображається форма з усіма полями	Сторінка відкривається, усі елементи на місці	Пройдено	-

Продовження таблиці Л.11

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
2	Валідація обов'язкових полів	Натискання "Register Parking" без введення даних	Показується помилка: “*Address is required”*	Повідомлення з помилкою відображається	Пройдено	-
3	Завантаження фото	Обрати зображення JPEG до 5MB	Назва файлу відображається, з'являється прев'ю	Фото завантажено, прев’ю з’явилось	Пройдено	-
4	Вибір координат на карті	Клік мишею по карті	З’являється маркер і відображаються координати	Маркер з’являється, координати відображаються	Пройдено	-
5	Успішна реєстрація	Заповнені всі поля + фото + вибрані координати, увімкнено Dynamic Pricing	Повідомлення “Parking registered successfully! Redirecting…” + перехід на сторінку	Повідомлення з’являється, перенаправлення працює	Пройдено	Тестувалось із валідним токеном
6	Перевірка без токена	Видалено jwtToken з localStorage	Помилка “Authentication required”	Помилка відображається	Пройдено	-
7	Валідація поля “Demand Factor”	Dynamic Pricing увімкнено, введено 5.0	Поле не дозволяє значення >3.0	Браузер не дозволяє ввести 5.0	Пройдено	HTML-обмеження працюють
8	API недоступне	Вимкнутий бекенд або сервер повертає 500	Повідомлення: “Failed to register parking. Please try again.”	Повідомлення з’являється	Пройдено	-

Продовження таблиці Л.11

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
9	Видалення фото	Після вибору фото оновити сторінку	Прев’ю та назва файлу очищаються	Прев’ю відсутнє, файл не збережено після перезавантаження	Пройдено	-
10	Перевірка checkbox	Увімкнути/вимкнути Dynamic Pricing	З’являється або зникає поле “Demand Factor”	Поле з’являється/зникає коректно	Пройдено	-

Таблиця Л.12 – Тест-кейс №12 (таблиця виконана самостійно)

Ідентифікатор тесту	Тест-кейс №12
Назва	Тестування функціоналу перегляду паркінгів користувача
Власник	Радько Марк Максимович
Дата проведення	03.06.2025
Мета тестування	Перевірити завантаження паркінгів користувача, коректність відображення, обробку помилок, переходи до сторінки керування, а також навігацію до реєстрації нового паркінгу.

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Відображення завантаження	Перехід на сторінку без попереднього кешу	Показується індикатор завантаження "Loading your parkings..."	Індикатор з’являється до завантаження даних	Пройдено	-

Продовження таблиці Л.12

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
2	Перевірка відсутності токена	Видалити jwtToken з localStorage	Відображається помилка “Authentication required”	Помилка відображається, з’являється блок з Retry	Пройдено	-
3	Перевірка помилки сервера	Підмінити відповідь API на помилку 500	Відображається повідомлення про помилку завантаження та кнопка “Retry”	Все працює згідно очікування	Пройдено	-
4	Кнопка Retry після помилки	Клік по кнопці “Retry”	Повторна спроба запиту до API	Дані завантажуються повторно після кліку	Пройдено	-
5	Відображення паркінгів	Валідний токен, API повертає 2+ паркінги	Відображаються картки з фото, назвою, адресою, статусом, прайсом, фактором попиту	Дані виводяться згідно макету	Пройдено	Перевірено з мок-даними
6	Обробка відсутнього зображення	API повертає пусте поле PhotoImage	Відображається зображення /placeholder-parking.png	Плейсхолдер працює, помилок в консолі немає	Пройдено	-
7	Обробка неактивного паркінгу	IsActive: false	Показується тег “Inactive”	Відображається сірий бейдж	Пройдено	-

Продовження таблиці Л.12

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
8	Обробка DynamicPricing: true/false	DynamicPricing: true або false	Показується відповідний тег (Dynamic/Fixed)	Відображається правильно	Пройдено	-
9	Переход до керування паркінгом	Клік на картку паркінгу	Значення ParkingID зберігається в localStorage, редірект на /parking-manage	Все працює, localStorage зберігає дані, навігація відбувається	Пройдено	-
10	Відсутність паркінгів	API повертає порожній масив	Відображається повідомлення “You don't have any parkings yet” та кнопка додати	Повідомлення з’являється, кнопка активна	Пройдено	-
11	Кнопка "Add Parking"	Клік на кнопку "Add Parking" у заголовку	Відбувається перехід на /register-parking	Навігація працює коректно	Пройдено	-
12	Відображення demand factor	DynamicPricing: true, DemandFactor: 2.2	Відображається тег “Demand: 2.2”	Тег з’являється, формат правильний	Пройдено	-

Таблиця Л.13 – Тест-кейс №13 (таблиця виконана самостійно)

Ідентифікатор тесту	Тест-кейс №13
Назва	Тестування функціоналу карти та взаємодії з паркуваннями
Власник	Радько Марк Максимович

Продовження таблиці Л.13

Дата проведення		03.06.2025				
Мета тестування		Перевірити коректне завантаження Google Maps, відображення маркерів, взаємодію з InfoWindow, додавання/видалення з обраного, створення бронювання та навігацію.				
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Завантаження Google Maps	Відкрити сторінку	Карта завантажується, показується місцезнаходження користувача або fallback	Карта з’являється, маркер користувача — на місці	Пройдено	-
2	Відображення паркувань	Авторизований користувач, наявні паркування	На карті з’являються маркери для кожного паркування	Усі маркери відображено	Пройдено	-
3	Взаємодія з маркером	Клік на маркер	Відкривається InfoWindow з деталями паркування	InfoWindow відкривається, містить всю інформацію	Пройдено	-
4	Додавання до обраного	Натиснути "Add to Favorites"	Паркування додається до списку обраних, кнопка змінює текст на "Remove Favorite"	Паркування додано, текст кнопки змінився	Пройдено	-
5	Видалення з обраного	Натиснути "Remove Favorite"	Паркування видаляється з обраного, кнопка	Паркування видалено, кнопка оновила текст	Пройдено	-

Продовження таблиці Л.13

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
			змінює текст на "Add to Favorites"			
6	Перевірка статусу Dynamic Pricing	Відкрити InfoWindow	Відображається значення: Dynamic Pricing: Enabled/Disabled	Відображається коректно	Пройдено	-
7	Відкриття Google Навігації	Натиснути кнопку "Navigate"	Відкривається Google Maps з побудованим маршрутом до паркування	Google Maps відкривається з правильними координатами	Пройдено	-
8	Відкриття модального вікна бронювання	Натиснути кнопку "Reserve Spot"	Відкривається модальне вікно створення бронювання	Вікно відкривається	Пройдено	Тестування самої модалки — окремо
9	Обробка помилки при завантаженні карти	Вимкнено інтернет або помилковий ключ	Показується повідомлення "Google Maps failed to load"	Повідомлення з’являється	Пройдено	Тестовано з підставним API-ключем
10	Обробка помилки при отриманні паркувань	Відсутній токен або API недоступне	Показується повідомлення "Could not load parking lots"	Повідомлення з’являється	Пройдено	Перевірено при видаленому токені з localStorage

Таблиця Л.14 – Тест-кейс №14 (таблиця виконана самостійно)

Ідентифікатор тесту		Тест-кейс №14				
Назва		Тестування функціоналу сторінки улюблених паркувань				
Власник		Радько Марк Максимович				
Дата проведення		04.06.2025				
Мета тестування		Перевірити завантаження списку улюблених паркувань, обробку помилок, візуалізацію елементів, видалення паркувань та зміну активного елемента.				
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Завантаження улюблених паркувань	Наявність токену в localStorage	Відображається список улюблених паркувань	Паркування відображаються коректно	Пройдено	-
2	Помилка при відсутності токену	Відсутній jwtToken у localStorage	Виводиться повідомлення про помилку автентифікації	Повідомлення "Authentication required" відображається	Пройдено	-
3	Обробка порожнього списку паркувань	У відповідь з сервера приходить пустий масив	Виводиться текст "You don't have any favourite parkings yet"	Повідомлення відображається	Пройдено	-
4	Видалення паркування зі списку	Клік на іконку смітника, підтвердження у вікні підтвердження	Паркування видаляється зі списку та більше не відображається	Паркування зникає після видалення	Пройдено	-
5	Відміна видалення паркування	Клік на іконку смітника, відмова у вікні підтвердження	Паркування залишається в списку	Видалення не відбулося	Пройдено	-

Продовження таблиці Л.14

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
6	Зміна активного (вибраного) паркування	Клік по картці з паркуванням	Карта виділяється (додається клас selected)	Клас додається, виділення видно	Пройдено	-
7	Обробка помилок від API при завантаженні паркувань	Сервер повертає помилку (наприклад, 500)	Відображається повідомлення про помилку і кнопка Retry	Повідомлення відображається, кнопка Retry працює	Пройдено	-
8	Відображення інформації про паркування	Об'єкт має поля PhotoImage, Name, Address, Info, IsActive	Всі елементи інформації коректно відображаються	Відображення правильне	Пройдено	Перевірено також підстановку картинки за замовчуванням
9	Некоректний формат даних з API	Сервер повертає об'єкт замість масиву	Виводиться повідомлення "Invalid data format received"	Повідомлення про помилку з’являється	Пройдено	-

Таблиця Л.15 – Тест-кейс №15 (таблиця виконана самостійно)

Ідентифікатор тесту	Тест-кейс №15
Назва	Тестування функціоналу створення бронювання
Власник	Радько Марк Максимович
Дата проведення	04.06.2025

Продовження таблиці Л.15

Мета тестування		Перевірити коректність валідації форми, відображення повідомлень про помилки та успішну резервацію, інтеграцію з API та логіку автозакриття модального вікна.				
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Відображення модального вікна	Відкриття компонента CreateReservationModal	Відображається форма з полями "Start Time" і "End Time", кнопка закриття, заголовок	Вікно з’являється, всі елементи присутні	Пройдено	-
2	Валідація порожніх полів	Порожні значення в обох полях	Повідомлення "Please select both start and end times"	Повідомлення з’являється	Пройдено	-
3	Валідація: час початку пізніше часу завершення	startTime = 18:00, endTime = 17:00	Повідомлення "End time must be after start time"	Повідомлення з’являється	Пройдено	-
4	Валідація: стартовий час у минулому	startTime = минула дата, endTime = майбутня	Повідомлення "Start time cannot be in the past"	Повідомлення з’являється	Пройдено	-
5	Валідація: не вибрана машина	localStorage.selectedVehicleID = '-1' або null	Повідомлення "Please select a vehicle in your profile"	Повідомлення з’являється	Пройдено	-

Продовження таблиці Л.15

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
6	Валідація: відсутній parkingID	parkingID не передано в props	Повідомлення "Parking information is missing"	Повідомлення з’являється	Пройдено	-
7	Успішна резервація	Введені коректні startTime, endTime, є parkingID і vehicleID	Виводиться повідомлення успіху, деталі резервації (ID, вартість), через 3 сек. модалка закривається	Повідомлення з’являється, модалка закривається	Пройдено	-
8	Обробка помилки з API	Сервер повертає 500	Виводиться повідомлення "Failed to create reservation. Please try again."	Помилка оброблена, повідомлення з’являється	Пройдено	-
9	Закриття модального вікна вручну	Клік по іконці “X”	Модальне вікно закривається	Закриття спрацювало	Пройдено	-
10	Кнопка підтвердження неактивна при сабміті	Відправка форми з валідними даними	Кнопка стає неактивною зі статусом "Processing..." на час очікування відповіді від API	Поведінка відповідає очікуванню	Пройдено	-

Таблиця Л.16 – Тест-кейс №16 (таблиця виконана самостійно)

Ідентифікатор тесту		Тест-кейс №16				
Назва		Тестування функціоналу управління користувачами				
Власник		Радько Марк Максимович				
Дата проведення		04.06.2025				
Мета тестування		Перевірити коректність пошуку користувачів за email, валідацію введених даних, відображення даних користувача, логіку блокування/розблокування, інтеграцію з API.				
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Валідація порожнього email	Поле email залишене порожнім	Виводиться повідомлення "Please enter a valid email address"	Повідомлення з’являється	Пройдено	-
2	Валідація некоректного email	user@@mail, abc, @site.com	Виводиться повідомлення "Please enter a valid email format"	Повідомлення з’являється	Пройдено	-
3	Пошук неіснуючого користувача	Існуючий email, який не прив’язаний до жодного акаунта	Повідомлення "User not found"	Повідомлення з’являється	Пройдено	-
4	Успішний пошук користувача	Коректний email існуючого користувача	Виводиться блок з даними користувача	Дані відображаються	Пройдено	-
5	Відображення статусу користувача	Активний або заблокований користувач	Відображається напис "Active" або "Banned"	Відображається коректно	Пройдено	-

Продовження таблиці Л.16

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
6	Відображення деталей користувача	Після успішного пошуку	Відображаються email, ім’я, роль, телефон, дата реєстрації	Всі поля з’являються	Пройдено	Якщо відсутні дані (наприклад, телефон) – показано "Not provided"
7	Заблокувати активного користувача	Клік по кнопці "Ban User" → підтвердження	Статус змінюється на "Banned", повідомлення про успіх	Користувач заблокований	Пройдено	-
8	Розблокувати заблокованого користувача	Клік по кнопці "Unban User" → підтвердження	Статус змінюється на "Active", повідомлення про успіх	Користувач розблокований	Пройдено	-
9	Відмова у підтвердженні бану/анбану	Клік по кнопці → скасувати в діалозі підтвердження	Жодних змін, діалог закривається	Поведінка відповідає очікуванню	Пройдено	-
10	Повідомлення про помилку API	Імітація помилки з боку сервера	Повідомлення "Failed to fetch user data..." або "Failed to update user status..."	Повідомлення виводиться	Пройдено	-

Додаток М

Мануальне тестування мобільного застосунку

Таблиця М.1 – Тест-кейс №1

Ідентифікатор тесту		Тест-кейс №1				
Назва		Тестування функціоналу авторизації				
Власник		Радько Марк Максимович				
Дата проведення		05.06.2025				
Мета тестування		Перевірити коректність роботи інтерфейсу авторизації, обробку помилок, UX-логику, валідацію та інтеграцію з бекендом.				
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Перевірка відображення елементів інтерфейсу	-	Всі поля, кнопки та посилання відображаються коректно	Всі елементи присутні, оформлення відповідає макету	Пройдено	-
2	Введення коректної пари email/пароль	Email: test@mail.com Пароль: Qwerty123	Авторизація успішна, перехід на головну сторінку, toast "Login success!"	Авторизація пройшла успішно, відображено повідомлення, перехід на MainActivity	Пройдено	-

Продовження таблиці М.1

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
3	Введення некоректної пари email/пароль	Email: wrong@mail.com Пароль: wrongpass	Показ повідомлення про помилку авторизації	Toast "Login failed: Invalid credentials"	Пройдено	Повідомлення відображено
4	Надсилання форми з порожніми полями	Email: "" Пароль: ""	Показ повідомлення "Please enter email and password"	Повідомлення виведено	Пройдено	-
5	Робота чекбоксу "Remember me"	Обрано чекбокс	У разі успішної авторизації токен зберігається у SharedPreferences	Значення rememberMe=true збережено разом із токеном	Пройдено	Перевірено в SharedPreferences
6	Переходи за посиланнями "Forgot Password?" та "Sign up"	Клік по кожному з посилань	Відкриваються відповідні активності: PassRecoveryActivity, RegisterActivity	Відкриваються відповідні екрани	Пройдено	-
7	Робота кнопки-перемикача видимості пароля	Натискання на іконку ока	Пароль стає видимим/невидимим	Перемикач працює, видимість пароля змінюється	Пройдено	Зберігається курсор після зміни режиму вводу
8	Відображення toast-повідомлень	-	Повідомлення мають бути інформативні, не перекривати інтерфейс	Повідомлення відображаються коректно, не	Пройдено	-

Продовження таблиці М.1

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
				заважають взаємодії		
9	UX логіка: кнопка login неактивна, поки поля не заповнено	Поля порожні	Кнопка активна завжди, але при натисканні без даних — повідомлення про помилку	Відображається toast, дія не виконується	Пройдено	Можна розглянути деактивацію кнопки до введення даних
10	Повторний запуск збереженого акаунту з rememberMe=true	Відкриття додатку	Вхід без повторного введення даних, токен вже збережено	Переадресація на головну сторінку	Пройдено	Зафіксовано збереження авторизованого стану через SharedPreferences

Таблиця М.2 – Тест-кейс №2

Ідентифікатор тесту	Тест-кейс №2
Назва	Тестування функціоналу реєстрації
Власник	Радько Марк Максимович
Дата проведення	05.06.2025
Мета тестування	Перевірити коректність роботи інтерфейсу реєстрації, обробку помилок, UX-логику, валідацію, інтеграцію з бекендом та навігацію після успішної реєстрації

Продовження таблиці М.2

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Відображення елементів	Відкриття RegisterActivity	Всі поля (ім’я, прізвище, email, телефон, пароль, мова, валюта), кнопка REGISTER, посилання Login присутні	Всі елементи відображаються згідно дизайну	Пройдено	Інтерфейс відповідає специфікації
2	Валідація заповнення полів	Натиснути REGISTER з порожніми полями	Повідомлення про помилки введення (наприклад, порожні поля, некоректний email)	Повідомлення не з’явилось — бекенд повертає помилку через Toast	Частково	Не вистачає клієнтської валідації на етапі введення
3	Успішна реєстрація	Коректні дані (імена, валюта, мова, email, телефон, пароль)	Отримано повідомлення Registered!, перехід до LoginActivity, збереження мови в SharedPreferences	Всі кроки виконано, Toast, навігація та збереження працюють	Пройдено	Реєстрація працює стабільно
4	Обробка помилок з бекенду	Використати вже зареєстрований email	Повідомлення Registration failed: Email already exists	Повідомлення виводиться через Toast, активність не змінюється	Пройдено	Серверне повідомлення коректно оброблено
5	Перевірка зміни типу пароля	Натискання на іконку еуе в полі паролю	Зміна режиму видимості паролю	Видимість змінюється відповідно до натискання	Пройдено	Зручний UX-функціонал

Продовження таблиці М.2

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
6	Вибір мови та валюти	Вибір зі спіннерів мов та валют	Вибрані значення додаються до запиту реєстрації	Дані передаються у registerRequest та обробляються на сервері	Пройдено	Варіанти зі списку коректно передаються
7	Перехід до сторінки входу	Натискання на посилання Login	Перехід до LoginActivity	Навігація працює, активність закривається	Пройдено	UX логіка відповідає очікуванням
8	Відображення повідомлень про помилки	Відсутність Інтернет-з’єднання	Повідомлення про помилку Registration failed: Network error	Повідомлення з’являється у Toast	Пройдено	Поведінка у разі мережевої помилки адекватна

Таблиця М.3 – Тест-кейс №3

Ідентифікатор тесту	Тест-кейс №3
Назва	Тестування карти з паркінгами та діалогу з деталями
Власник	Радько Марк Максимович
Дата проведення	05.06.2025
Мета тестування	Перевірити коректність відображення карти, маркерів, логіку відкриття діалогу з інформацією про паркінг, роботу кнопок (навігація, бронювання, паркування), обробку координат та UX-поведінку.

Продовження таблиці М.3

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Відкриття фрагменту з картою	Запуск HomeFragment	Карта завантажується, показуються маркери тестових паркінгів	Карта завантажилась, маркери видно	Пройдено	-
2	Передача координат паркінгу через FragmentResult	lat=47.249912, lng=39.697196	Камера переміщується до паркінгу, відкривається діалог	Камера перемістилась, діалог відкрився	Пройдено	-
3	Переміщення камери до координат, що не мають паркінгу	lat=0.0, lng=0.0	Камера переміщується, діалог не з’являється	Камера перемістилась, діалогу немає	Пройдено	-
4	Відображення діалогу з інформацією	Натискання на маркер	З’являється діалог з назвою, описом, динамічним ціноутворенням	Діалог з’явився, усі поля присутні	Пройдено	-
5	Кнопка “Navigate”	Натиснута кнопка	Відкривається Google Maps для навігації до координат	Google Maps відкрився з правильними координатами	Пройдено	-
6	Кнопка “Park”	Натиснута кнопка	Виводиться повідомлення про спробу паркування	Toast з’явився	Пройдено	-
7	Кнопка “Reservation”	Натиснута кнопка	Відкривається CreateReservationActivity	Активність відкрилася	Пройдено	-

Продовження таблиці М.3

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
8	Додавання до улюбленого	Натискання на зірку	Змінюється іконка, показується Toast	Зірка змінилась, Toast відображено	Пройдено	-
9	Тестування в нічному режимі	Увімкнено темну тему	Карта стилізується відповідно до теми	Темна тема застосована	Пройдено	-
10	Обробка відсутніх або некоректних координат	lat=null, lng="abc"	Карта відкривається, але без переміщення, діалог не відкривається	Карта відкрилась, переміщення і діалог не відбулося	Пройдено	Присутнє логування помилок

Таблиця М.4 – Тест-кейс №4

Ідентифікатор тесту	Тест-кейс №4					
Назва	Тестування створення бронювання					
Власник	Радько Марк Максимович					
Дата проведення	05.06.2025					
Мета тестування	Перевірити коректність роботи екрана створення бронювання: взаємодію з UI-елементами, валідацію часу, обробку виняткових ситуацій, інтеграцію з бекендом та UX-поведінку.					
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Відкриття сторінки CreateReservation Activity	Вхідний параметр ParkingID = 15, vehicleID = 3 у SharedPrefs	Сторінка відкривається з полями для введення часу, кнопкою створення	Сторінка завантажилась, усі елементи відображаються	Пройдено	-

Продовження таблиці М.4

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
2	Перевірка відсутності ParkingID	ParkingID = -1	Показ повідомлення "Parking ID not provided", сторінка закривається	Повідомлення з’явилось, користувач перенаправлений назад	Пройдено	-
3	Відсутність vehicleID у SharedPreferences	SharedPrefs не містить selectedVehicleID	Показ повідомлення "No vehicle selected", закриття активності	Повідомлення з’явилось, активність завершена	Пройдено	-
4	Вибір коректних дати та часу початку/завершення	Обрано: StartTime = 10:00, EndTime = 12:00	Поля заповнюються, кнопка активна	Час встановлено, дані видно в полях	Пройдено	Дата/час конвертуються коректно
5	Валідація при незаповнених полях	Поля часу не вибрано	Повідомлення "Please select both times"	Повідомлення з’являється, запит не надсилається	Пройдено	Перевірка заповнення обов'язкових полів
6	Валідація на неправильну послідовність часу	StartTime > EndTime	Повідомлення "Start time must be before end time"	Повідомлення показано	Пройдено	-
7	Створення бронювання при валідних даних	Коректні ParkingID, VehicleID, час, токен	Запит надсилається, повідомлення від сервера відображається у Toast	Запит успішно відправлено, Toast показав "Reservation	Пройдено	Бронювання створено, відповідь оброблено

Продовження таблиці М.4

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
				created successfully"		
8	Створення без JWT токена	JWT не записано в SharedPrefs	Показ повідомлення "No token found. Please login again."	Повідомлення показано, запит не відправлено	Пройдено	Захист від неавторизовани х дій
9	Поведінка при помилці з боку сервера	Сервер повертає помилку 400	Повідомлення з описом помилки у Toast	В Toast виводиться повідомлення "Reservation time is invalid"	Пройдено	-
10	UX: натискання кнопки "Назад" на тулбарі	Натискання кнопки	Закриття поточної активності	Повернення до попереднього екрану	Пройдено	-

Таблиця М.5 – Тест-кейс №5

Ідентифікатор тесту	Тест-кейс №5
Назва	Тестування функціоналу профілю користувача
Власник	Радько Марк Максимович
Дата проведення	05.06.2025
Мета тестування	Перевірити правильність завантаження та оновлення даних профілю, логіку взаємодії з транспортними засобами, валідацію введення, інтеграцію з API та UX-досвід.

Продовження таблиці М.5

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Завантаження даних профілю після входу	Токен авторизованого користувача	Дані користувача (email, ім’я, прізвище, телефон, баланс, бонуси) коректно відображаються в полях	Всі поля успішно заповнено, токен зчитано, дані відображено	Пройдено	-
2	Оновлення даних користувача	Нові ім’я, прізвище та номер телефону	Сервер повертає повідомлення про успішне оновлення, зміни зберігаються	Оновлення пройшло успішно, повідомлення з серверу показано через Toast	Пройдено	-
3	Обробка помилок при оновленні	Невалідний номер телефону або порожні поля	Відображення помилки/відповідного повідомлення про помилку	Відображено Toast з помилкою "Error: ..." від ViewModel	Пройдено	Обробка помилок реалізована через LiveData
4	Перехід до поповнення рахунку	Натискання кнопки “Fill up”	Перехід до PaymentActivity з вказаним типом транзакції	Перехід відбувся коректно, тип передано через Intent	Пройдено	Реалізовано через Intent.putExtra
5	Відображення переліку	Запит з токеном до API	Дані про ТЗ (марка, категорія, номер, зображення)	Усі зареєстровані ТЗ відображено,	Пройдено	-

Продовження таблиці М.5

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
	транспортних засобів		відображаються в RecyclerView	зображення завантажено через Glide		
6	Видалення транспортного засобу	Натискання на кнопку видалення біля конкретного ТЗ	Транспортний засіб видаляється, список оновлюється	ТЗ видалено, список оновлено, з’явився Toast “Vehicle deleted”	Пройдено	-
7	Вибір транспортного засобу	Натискання на елемент RecyclerView	Виділення активного елемента, збереження ID в SharedPreferences	ID збережено, фон змінено на selected_item_background	Пройдено	Автоматичний вибір, якщо ТЗ один — працює
8	Обробка помилок під час завантаження або видалення ТЗ	Відсутність інтернет-з'єднання або помилковий токен	Вивід повідомлення про помилку через Toast	Відображено текст помилки з vehicleViewModel.error	Пройдено	-
9	Відображення залишку на рахунку та бонусів	Отримані з API значення Balance, Currency, Bonuses	Відображаються відповідні значення в полях "Balance: ..." і "Bonuses: ..."	Значення збігаються з тими, що отримано з бекенду	Пройдено	-

Таблиця М.6 – Тест-кейс №6

Ідентифікатор тесту		Тест-кейс №6				
Назва		Тестування функціоналу оплати				
Власник		Радько Марк Максимович				
Дата проведення		05.06.2025				
Мета тестування		Перевірити коректність роботи інтерфейсу платежів, підтримку валют, вибір способу оплати, валідацію введених даних, інтеграцію з сервером, логіку транзакцій та UX-поведінку додатку під час створення/підтвердження платежу.				
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Відкриття екрану PaymentActivity	Intent з типом транзакції DEPOSIT	Відображення полів введення суми, вибору валюти та способу оплати	Всі елементи інтерфейсу з'являються коректно	Пройдено	-
2	Вибір валюти	Клік по currencySelector, обрано "USD"	Автоматичне оновлення доступних методів оплати (PayPal, LiqPay, Balance)	Методи змінюються відповідно до вибраної валюти	Пройдено	-
3	Спроба створити депозит без введення суми	Поле суми порожнє, натиснута кнопка Pay	Відображення повідомлення про помилку: "Please enter a valid amount"	Повідомлення з'являється, транзакція не ініційована	Пройдено	-
4	Створення депозиту через LiqPay	Введено суму 100.00, валюта UAH, метод LiqPay, тип DEPOSIT	Перехід у браузер за посиланням на оплату, збереження transactionID і approvalUrl	Браузер відкриває сторінку оплати, дані збережені у SharedPreferences	Пройдено	Відображається інструкція "Payment Instructions" після відкриття браузера

Продовження таблиці М.6

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
5	Повернення до застосунку без підтвердження	Закрито браузер без завершення оплати	Додаток чекає підтвердження, наступного запуску виконується повторна перевірка	Діалог підтвердження не з’являється до повернення, повторна перевірка працює	Пройдено	-
6	Повторний запуск — підтвердження LiqPay	При повторному запуску додатку transactionId не нульовий	Автоматичний виклик confirmPendingTransaction, підтвердження транзакції	Підтвердження виконується, діалог "Success" з повідомленням про успішну оплату	Пройдено	-
7	Створення оплати за резервацію з бонусами через Balance	Тип PAYMENT, резервація в SharedPreferences, обрано метод Balance, активовано чекбокс бонусів	Відображення суми, ініціація оплати без переходу до браузера	Виведено повідомлення про успішну оплату: "Payment completed successfully!"	Пройдено	Без переходу у браузер, логіка з бонусами працює коректно
8	Помилка підтвердження — неправильна URL або токен	Неправильний токен або approvalUrl	Відображення помилки: "Confirmation failed"	Відображення повідомлення про помилку, зупинка прогрес-бара	Пройдено	-

Продовження таблиці М.6

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
9	Закриття активності вручну	Клік на кнопку Cancel або Close	Закриття екрана без змін у транзакціях	Екран закривається, дані транзакції очищуються	Пройдено	-
10	Некоректна валідація методу оплати	Не обрано спосіб оплати	Відображення помилки: "Please select payment method"	Повідомлення відображено, кнопка Pay неактивна	Пройдено	Валідація введення повна

Таблиця М.7 – Тест-кейс №7

Ідентифікатор тесту		Тест-кейс №7				
Назва		Тестування функціоналу відновлення пароля				
Власник		Радько Марк Максимович				
Дата проведення		05.06.2025				
Мета тестування		Перевірити коректність логіки UI/UX, валідації введення email, коду, пароля, переходу між кроками, обробки помилок та взаємодії з ViewModel				
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Введення порожнього email	""	Відображення повідомлення Please enter your email	Повідомлення з’являється через Snackbar	Пройдено	-
2	Введення коректного email	test@example.com	Відображення Code sent to email, перехід до кроку введення коду	Перехід до наступного етапу, Toast і	Пройдено	Повідомлення користувачу надіслане

Продовження таблиці М.7

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
	та натискання "Send Code"			Snackbar показані		
3	Введення не повного коду (менше 6 цифр)	12345, пароль Test1234!	Відображення повідомлення Please fill all fields correctly	Повідомлення з’являється через Snackbar	Пройдено	Блокує перехід до reset
4	Введення повного коду та нового пароля	123456, Test1234!	Повідомлення Password changed successfully, перехід до логіну	Toast показує успіх, виконується startActivity(LoginActivity)	Пройдено	Пароль оновлено
5	Введення коректних даних, але симуляція помилки на сервері	notfound@email.com, код 123456, пароль Test1234!	Відображення повідомлення про помилку (з ViewModel error)	Snackbar з текстом помилки	Пройдено	Обробка помилок через LiveData
6	Автоматичне перемикання фокусу між полями коду	Введення цифри в перше поле	Курсор переходить на наступне поле	Перемикання працює	Пройдено	-
7	Введення некоректного пароля (порожнє поле)	Код 123456, пароль ""	Повідомлення Please fill all fields correctly	Відображення Snackbar	Пройдено	Захист від пустих паролів

Таблиця М.8 – Тест-кейс №8

Ідентифікатор тесту		Тест-кейс №8				
Назва		Тестування функціоналу додавання транспортного засобу				
Власник		Радько Марк Максимович				
Дата проведення		05.06.2025				
Мета тестування		Перевірити коректність логіки введення та валідації даних, завантаження зображення, інтеграції з бекендом та відображення повідомлень у випадку помилок і успішного створення				
№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Відкриття екрану	-	Екран відкривається, показано поля та кнопки	Інтерфейс відображається коректно	Пройдено	-
2	Натискання кнопки "Upload image" без дозволу	-	Запит дозволу на читання сховища	Система запитує дозвіл	Пройдено	Android permission flow
3	Натискання "Upload image" при наданому дозволі	-	Відкривається галерея	Галерея відкривається, зображення обране	Пройдено	Зображення оброблено, текст кнопки змінено на Image Selected
4	Введення порожніх полів та натискання "Add Vehicle"	"" в усіх полях	Toast з повідомленням Please fill all fields	Повідомлення з’являється	Пройдено	-
5	Введення всіх полів без зображення	Текст у полях, без вибору фото	Помилка (imagePart == null), Toast про відсутність фото	Повідомлення про виняток або помилку	Частково	Виняток можливий при imagePart,

Продовження таблиці М.8

№	Етап тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус	Коментар
						доцільно додати додаткову перевірку
6	Введення даних + фото, відсутній токен	Поля заповнено, зображення вибрано, токен відсутній	Toast: No token found. Please login again.	Повідомлення з’являється	Пройдено	-
7	Повне введення валідних даних з фото та токеном	Категорія, номер, марка, модель, фото, дійсний токен	Toast: Vehicle added successfully, закриття активності	Повідомлення про успішне додавання, завершення активності	Пройдено	-
8	Симуляція помилки з бекенду	Невалідний токен або помилковий запит	Toast з повідомленням помилки	Повідомлення про помилку показується	Пройдено	LiveData error
9	Натискання кнопки назад у тулбарі	-	Закриття активності, повернення назад	Повернення до попереднього екрану	Пройдено	-