

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ «VROOMIE»
DEVELOPMENT OF THE «VROOMIE» MOBILE APPLICATION

Виконав(ла): студент(ка) 4 курсу, групи КНТ-151

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ЛАВРОВ Я.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ПОЛЯКОВ М.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРИЗВИЩЕ та ініціали)

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

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

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЛАВРОВА Яна Вікторівна
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка мобільного застосунку «Vroomie».
Development of the «Vroomie» mobile application

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

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. 6. Експлуатація, тестування та експериментальне дослідження програми.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	ПОЛЯКОВ М.О., професор		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання « 14 » квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент(ка)

Підпис
(підпис)

ЯН ЛАВРОВ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Михайло ПОЛЯКОВ
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 136 с., 12 табл., 62 рис., 3 дод., 19 джерел.

ОРЕНДА АВТО, VROOMIE, CAR RENTAL, WPF, C#, MONGODB, КАРШЕРІНГ.

Об'єкт дослідження – процес автоматизації оренди автомобілів для покращення доступності та зручності сервісу.

Предмет дослідження – мобільний додаток для автоматизації процесу оренди автомобілів, що забезпечує користувачам зручний доступ до бронювання транспорту, перегляду доступних авто та управління власними замовленнями.

Мета роботи – розробка мобільного додатку для бронювання автомобілів, який дозволяє користувачам швидко знаходити, орендувати транспортні засоби та керувати своїм бронюванням в інтерактивному середовищі.

Матеріали, методи та технічні засоби: використано принципи об'єктно-орієнтованого програмування, застосовано мову програмування C#, фреймворк WPF для розробки графічного інтерфейсу, базу даних MongoDB для зберігання інформації про користувачів та автомобілі, середовище розробки Microsoft Visual Studio 2022.

Результати. Створено мобільний додаток – «Vroomie», який забезпечує зручний процес оренди автомобілів. Функціонал включає реєстрацію та авторизацію користувачів, перегляд доступних авто, вибір параметрів оренди, оформлення бронювання та оплату. Реалізовано особистий кабінет із можливістю перегляду історії замовлень, керування платіжними методами, збереження улюблених авто та налаштування профілю. Додатково розроблено

адміністративну панель для керування списком автомобілів та обробки запитів користувачів.

Висновки. Створений додаток покращує процес оренди автомобілів, знижує часові витрати користувачів та оптимізує управління автопарком. Завдяки автоматизації процесу оренди, клієнти можуть швидко та безпечно бронювати транспортні засоби відповідно до своїх потреб. Система також полегшує роботу адміністраторів, дозволяючи ефективно керувати даними та взаємодією з користувачами.

Галузь використання – сервіси з оренди автомобілів, онлайн-платформи для бронювання транспорту, мобільні додатки для каршерінгу та автоматизовані системи управління прокатом авто.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 136 p., 12 tables, 62 pictures, 3 additions, 19 sources.

CAR RENTAL, VROOMIE, CAR RENTAL, WPF, C#, MONGODB, CARSHARING.

The object of research is the process of automating car rental to improve the availability and convenience of the service.

The subject of the study is a mobile application for automating the car rental process, which provides users with convenient access to booking transport, viewing available cars and managing their own orders.

The purpose of the study is to develop a mobile application for car booking that allows users to quickly find, rent vehicles and manage their bookings in an interactive environment.

Materials, methods and techniques: the principles of object-oriented programming were used, the C# programming language, the WPF framework for developing a graphical interface, the MongoDB database for storing information about users and cars, the Microsoft Visual Studio 2022 development environment were used.

Results. A mobile application, Vroomie, has been created that provides a convenient car rental process. The functionality includes user registration and authorization, viewing available cars, selecting rental options, making a reservation, and paying. A personal account has been implemented with the ability to view order history, manage payment methods, save favorite cars, and customize your profile. Additionally, we have developed an administrative panel for managing the list of cars and processing user requests. Conclusions. The created application improves the car rental process, reduces user time and optimizes fleet management. By automating the rental process, customers can quickly and securely book vehicles

according to their needs. The system also facilitates the work of administrators, allowing them to effectively manage data and user interaction.

Applications include car rental services, online platforms for booking vehicles, mobile applications for car sharing, and automated car rental management systems.

ЗМІСТ

С.

Перелік скорочень та умовних познач.....	Ошибка! Закладка не определена.
Вступ.....	12
1 Аналіз предметної області.....	14
1.1 Особливості онлайн оренди авто.....	14
1.2 Опис предмета та мети дослідження.....	15
1.3 Аналіз існуючих додатків для онлайн оренди авто.....	15
1.3.1 Uber Rent	16
1.3.2 Turo	17
1.3.3 Getaround.....	18
1.4 Формальна постановка задачі	21
1.5 Висновки за розділом 1	22
2 Матеріали і методи.....	24
2.1 Вибір середовища розробки.....	24
2.1.1 Аналоги середовища розробки	25
2.2 Вибір архітектури проєктування	27
2.2.1 Аналоги архітектури проєктування	29
2.3 Вибір бази даних	32
2.3.1 Аналоги баз даних.....	33
2.4 Вибір мови програмування	37
2.4.1 Аналоги мови програмування.....	38
2.5 Висновки за розділом 2	40
3 Основні рішення щодо реалізації компонентів системи.....	42
3.1 Функціональні можливості розробленого програмного забезпечення .	42
3.2 Загальна архітектура схеми програми	43
3.3 Принципи роботи програмного забезпечення	44
3.4 Вибір шаблону для архітектури.....	48
3.4.1 Діаграма класів	48
3.4.2 Візуалізація компонентів системи.....	52

3.4.3 Діаграма взаємодії основних елементів.....	52
3.5 Концептуальна модель бази даних.....	53
3.6 Взаємодія модулів системи.....	57
3.6.1 Діаграма випадків використання.....	58
3.7 Висновки за розділом 3	62
4 Керівництво програміста.....	63
4.1 Призначення та умови використання додатку.....	63
4.2 Основні характеристики програми.....	64
4.3 Інструкція із запуску та використання додатку.....	65
4.4 Вхідні та вихідні дані.....	66
4.5 Системні повідомлення	67
4.6 Висновки за розділом 4	68
5 Керівництво оператора	70
5.1 Призначення додатку.....	70
5.2 Умови роботи додатку	70
5.2.1 Апаратні вимоги.....	70
5.2.2 Вимоги до користувача	70
5.3 Робота з додатком	71
5.4 Системні повідомлення для користувача	96
5.5 Висновки за розділом 5	97
6 Експлуатація, тестування та експериментальне дослідження програми	99
6.1 Загальні положення.....	99
6.2 Підготовка до експлуатації додатку.....	99
6.3 Розробка тестової документації.....	100
6.4 Методологія тестування та процес перевірки.....	101
6.5 Аналіз результатів тестування.....	103
6.6 Висновки за розділом 6	104
Висновки	105
Перелік джерел посилання	106
Додаток А Технічне завдання	108

Додаток Б Текст програми 116

Додаток В Слайди презентації..... 126

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

APP – Application;

MDB – MongoDB;

MOB – Mobile;

VR – Vroomie;

VS – Visual Studio;

WPF – Windows Presentation Foundation;

ПЗ – програмне забезпечення.

ВСТУП

Оренда автомобілів є важливою складовою сучасної транспортної системи, забезпечуючи користувачам гнучкість та зручність у пересуванні без необхідності купівлі власного транспорту. У всьому світі ринок оренди авто стрімко розвивається, завдяки цифровим технологіям, які дозволяють значно спростити процес бронювання та використання транспортних засобів. Мобільні додатки для оренди автомобілів відіграють ключову роль у цьому процесі, оскільки забезпечують швидкий доступ до необхідної інформації, спрощують взаємодію між користувачем та компанією, а також сприяють оптимізації управління автопарком.

Об'єкт дослідження – процес автоматизації оренди автомобілів для покращення доступності та зручності сервісу.

Предмет дослідження – мобільний додаток для автоматизації процесу оренди автомобілів, що забезпечує користувачам зручний доступ до бронювання транспорту, перегляду доступних авто та управління власними замовленнями.

Мета роботи – розробка мобільного додатку для бронювання автомобілів, який дозволяє користувачам швидко знаходити, орендувати транспортні засоби та керувати своїм бронюванням в інтерактивному середовищі.

Для досягнення поставленої мети необхідно провести аналіз сучасних рішень у сфері оренди автомобілів, розробити функціональну структуру додатку, реалізувати основні можливості, такі як авторизація користувачів, пошук авто, оформлення бронювання, вибір способу оплати та управління профілем користувача. Також важливо забезпечити високий рівень безпеки даних та зручність інтерфейсу.

Результатом роботи є створення мобільного додатку, що дозволяє автоматизувати процес оренди автомобілів, надаючи користувачам інтуїтивно зрозумілий і функціональний інструмент для швидкого доступу до

транспортних засобів. Даний додаток сприяє підвищенню рівня зручності, зменшенню часу на оформлення оренди та покращенню взаємодії між компанією та їхніми клієнтами.

Зважаючи на активний розвиток цифрових сервісів та зростання попиту на оренду авто, створення такого мобільного додатку є актуальним завданням, що дозволить покращити доступність транспортних засобів для широкого кола користувачів та зробити процес оренди максимально комфортним і швидким.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Особливості онлайн оренди авто

Сучасні цифрові технології значно змінили підхід до оренди автомобілів, зробивши цей процес більш доступним, швидким і зручним для користувачів. Онлайн-платформи та мобільні додатки дозволяють бронювати авто у будь-який час доби без необхідності відвідування офісу компанії, що забезпечує максимальну гнучкість для клієнтів. Важливим аспектом є прозоре ціноутворення, адже користувачі заздалегідь можуть ознайомитися з вартістю оренди, доступними тарифами та додатковими послугами, такими як страхування, GPS-навігація чи дитяче крісло.

Автоматизація процесу оформлення дозволяє значно скоротити час, необхідний для завершення угоди, оскільки в онлайн-сервісах реалізовано цифрову перевірку документів, електронні контракти та можливість підписання угоди дистанційно. Додаткову зручність забезпечують різноманітні способи оплати, включаючи банківські картки, електронні гаманці та інші цифрові платіжні системи. Завдяки вбудованим системам рейтингу та відгуків користувачі можуть оцінювати якість послуг та залишати коментарі щодо стану автомобілів, що сприяє підвищенню рівня довіри до сервісу.

Інтеграція геолокаційних технологій дозволяє швидко знаходити найближчі доступні автомобілі та отримувати маршрути до пунктів видачі. Деякі сервіси навіть пропонують безконтактне отримання авто, коли користувач може розблокувати транспортний засіб через мобільний додаток без особистої зустрічі з представником компанії.

Таким чином, онлайн оренда автомобілів поєднує у собі передові технології, автоматизовані процеси та зручний користувацький досвід, що робить цей сервіс привабливим та доступним для широкого кола споживачів.

1.2 Опис предмета та мети дослідження

Предметом дослідження є мобільний додаток, призначений для автоматизації процесу оренди автомобілів. Такий додаток забезпечує користувачам зручний доступ до повного спектру послуг, пов'язаних з орендою транспорту, починаючи від перегляду доступних автомобілів, вибору оптимального варіанту за характеристиками та вартістю, і закінчуючи процесом бронювання, оплатою та управлінням власними замовленнями. Сучасні цифрові технології дозволяють реалізувати широкий функціонал, що включає можливість фільтрації пропозицій за параметрами, перегляд відгуків інших користувачів та надання рекомендацій на основі історії бронювань.

Метою роботи є розробка мобільного додатку для бронювання автомобілів, який дозволяє користувачам швидко знаходити, орендувати транспортні засоби та керувати своїми бронюванням в інтерактивному середовищі. Основний акцент буде робитись на мінімізації часу, необхідного для оформлення оренди, та забезпеченні максимальної зручності взаємодії з сервісом. Для цього реалізується інтуїтивно зрозумілий інтерфейс, що дозволить користувачам легко орієнтуватися у функціоналі, а також впровадиться система керування бронюванням, яка дозволяє переглядати історію оренд, вносити зміни в поточні замовлення та швидко отримувати необхідну інформацію про автомобіль.

1.3 Аналіз існуючих додатків для онлайн оренди авто

Сучасний ринок мобільних додатків для оренди автомобілів пропонує безліч рішень, що забезпечують користувачам можливість швидкого бронювання транспорту. Для детальнішого аналізу було обрано три популярні аналоги: Uber Rent, Turo та Getaround. Ці сервіси надають різні підходи до оренди авто, кожен із яких має свої переваги та недоліки, що варто враховувати при розробці власного додатку.

1.3.1 Uber Rent

Uber Rent – це сервіс від компанії Uber, що дозволяє користувачам бронювати автомобілі напряму через додаток Uber. Основна перевага сервісу – його інтеграція в екосистему Uber, що робить процес оренди максимально швидким та зручним для користувачів, які вже мають обліковий запис. Оренда здійснюється через партнерські компанії, що гарантує високу якість обслуговування [1].

Переваги:

- інтеграція з додатком Uber, що дозволяє швидко оформити оренду без необхідності реєстрації у новому сервісі;
- великий вибір автомобілів завдяки співпраці з великими прокатними компаніями;
- зручна система бронювання та простий інтерфейс.

Недоліки:

- відсутність можливості вибору конкретної моделі авто, оскільки система автоматично підбирає доступний транспорт;
- відсутність функції додавання авто в обране для швидкого доступу до бажаних варіантів;
- обмежений набір фільтрів для пошуку авто, що ускладнює підбір транспорту за специфічними критеріями.

На рисунку 1.1 зображено інтерфейс мобільного додатку Uber Rent [1].

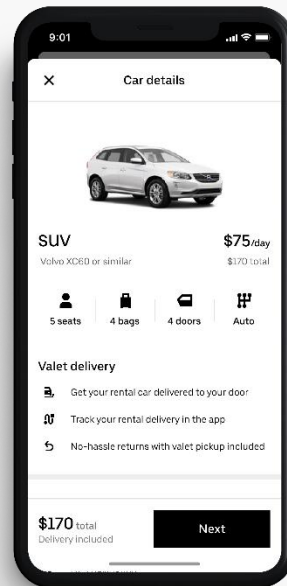


Рисунок 1.1 – Інтерфейс мобільного додатку Uber Rent [1]

1.3.2 Turo

Turo – це платформа для оренди автомобілів напряму у приватних власників. Користувачі можуть обирати серед широкого асортименту авто, які пропонують інші люди, а не компанії. Turo працює за принципом каршерінгу та дозволяє орендувати як бюджетні, так і преміальні авто [2].

Переваги:

- гнучка система фільтрації та пошуку автомобілів за різними параметрами (марка, модель, ціна, місцезнаходження);
- інтеграція з картографічними сервісами для зручного пошуку та навігації до авто;
- система рейтингу та відгуків, яка допомагає користувачам оцінювати якість сервісу перед бронюванням.

Недоліки:

- відсутність можливості інтеграції з Apple Pay та Google Pay для швидких платежів;

- відсутність push-сповіщень про статус бронювання або зміни умов оренди;
- недостатньо оптимізований інтерфейс для користувачів, що впливає на швидкість навігації в додатку.

На рисунку 1.2 зображено інтерфейс мобільного додатку Turo [2].

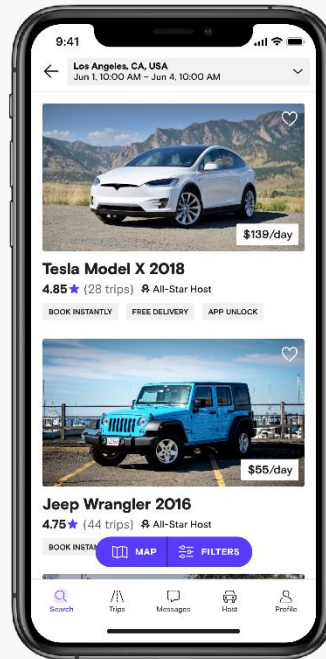


Рисунок 1.2 – Інтерфейс мобільного додатку Turo [2]

1.3.3 Getaround

Getaround – це сервіс, що поєднує концепцію каршерінгу та традиційної оренди. Його головна особливість – технологія безконтактної оренди, що дозволяє користувачам бронювати та відкривати авто через мобільний додаток без необхідності зустрічі з власником [3].

Переваги:

- можливість вибору авто для оренди з водієм чи без;
- інтеграція зі сторонніми сервісами для перевірки водійських прав перед орендою;
- автоматизована система страхування, яка додається до кожного

бронювання, забезпечуючи безпеку угод.

Недоліки:

- відсутність можливості перегляду відгуків користувачів про конкретне авто перед бронюванням;
- відсутність гнучкої системи цінових знижок для постійних користувачів;
- обмежені варіанти оплати – відсутність підтримки популярних мобільних платіжних систем.

На рисунку 1.3 зображено інтерфейс мобільного додатку Getaround [3].



Рисунок 1.3 – Інтерфейс мобільного додатку Getaround [3]

У рамках цього дослідження буде створено новий програмний продукт – мобільний додаток «Vroomie», який автоматизує процес оренди автомобілів, роблячи його зручним та доступним для широкого кола користувачів. Основні функціональні можливості додатку включатимуть:

- оренду автомобіля з можливістю вибору параметрів, таких як наявність водія;
- перегляд налаштувань авто, зокрема, його характеристик, фото та доступності;

- вибір автомобілів різних брендів, що дозволить користувачам знайти транспорт відповідно до своїх уподобань;
- інтерактивну мапу для відображення доступних автомобілів у найближчих локаціях;
- управління бронюванням та особистим профілем користувача.

У таблиці 1.1 зображено порівняння вже існуючих програмних продуктів та додатку, який буде розроблений в ході виконання роботи.

Таблиця 1.1 – Порівняння вже існуючих програмних продуктів та додатку, що розробляється

Критерії	Vroomie	Uber Rent	Turo	Getaround
Вибір автомобілів різних брендів	+	+	+	–
Можливість додавання авто до улюблених	+	–	–	+
Інтерактивна мапа	+	–	–	–
Можливість вибору авто з водієм	+	–	–	+
Персоналізоване управління бронюванням	+	–	–	–
Офлайн-доступ до бронювань	+	–	+	–
Зв'язок с технічною підтримкою прямо у додатку	+	–	–	–

Таким чином, у ході аналізу сучасного ринку мобільних додатків для оренди автомобілів було виявлено, що сервіси, такі як Uber Rent, Turo та Getaround, мають різні підходи до надання послуг, кожен з яких має свої

переваги та недоліки. Uber Rent відрізняється інтеграцією з іншими послугами Uber, що підвищує зручність для користувачів. Туго пропонує більш гнучкий підхід, дозволяючи орендувати авто у приватних осіб, що розширює варіанти та можливості для клієнтів. Getaround, завдяки можливості орендувати авто з водієм чи без, забезпечує додатковий рівень зручності та автономії для користувачів.

1.4 Формальна постановка задачі

Формалізація задачі є ключовим етапом у розробці програмного продукту, оскільки вона визначає основні параметри, які впливатимуть на його функціональність, продуктивність та обмеження.

Вхідними даними для мобільного додатку «Vroomie» є облікові дані користувача, такі як електронна пошта та пароль для авторизації, а також особисті дані, включаючи номер телефону, ім'я, прізвище та інші параметри, необхідні для створення та використання облікового запису.

Вихідними даними є інформація про автомобілі, доступні для оренди, зокрема їхні характеристики, фотографії, місцезнаходження та інші супутні дані, а також загальна інформація про сервіс, умови оренди та правила використання.

Результатом роботи стане розробка мобільного додатку «Vroomie», який надаватиме користувачам можливість зручного бронювання автомобілів через смартфон. Додаток включатиме функціонал для авторизації користувачів, перегляду доступних авто, вибору транспортного засобу за характеристиками, інтеграції з мапою для відображення місця розташування авто, керування бронюванням та особистим профілем.

Критеріями оцінювання ефективності роботи додатку є:

- час відгуку програми – швидкість завантаження даних, обробка запитів користувача та оперативність взаємодії з сервером;
- зручність інтерфейсу – наявність інтуїтивного дизайну, простота

використання, швидкий доступ до основних функцій без зайвих дій;

– надійність – стійкість додатку до збоїв, коректна робота всіх функцій, безпека даних користувачів та їхня конфіденційність.

Обмеженнями для реалізації додатку є сумісність із платформами Android та iOS, що вимагає адаптації інтерфейсу та функціоналу під особливості кожної ОС.

Крім того, для зберігання та обробки інформації буде використовуватися база даних MongoDB, що визначає формат збереження даних, структуру взаємодії з сервером та забезпечення масштабованості проєкту.

1.5 Висновки за розділом 1

У першому розділі було проведено аналіз предметної області, що включає особливості онлайн оренди автомобілів, формулювання мети та предмета дослідження, огляд існуючих аналогів, а також формальну постановку задачі.

Було розглянуто основні аспекти онлайн оренди авто, визначено ключові переваги та проблеми цього сегмента. Проаналізовано три популярні сервіси – Uber Rent, Turo та Getaround, які мають свої унікальні функціональні особливості, переваги та недоліки. На основі проведеного аналізу визначено основні відмінності між цими сервісами та майбутнім додатком Vroomie, що дозволило сформулювати перелік ключових функцій, які необхідно реалізувати для підвищення конкурентоспроможності продукту.

Також була проведена формальна постановка задачі, де визначено вхідні та вихідні дані, бажані результати, критерії оцінювання ефективності роботи додатку та можливі обмеження. Важливим аспектом є підтримка платформ Android та iOS, інтеграція з базою даних MongoDB, а також високий рівень продуктивності та зручності використання.

Таким чином, у цьому розділі сформовано основу для подальшої

розробки програмного продукту, що дозволить ефективно реалізувати всі необхідні функції, враховуючи особливості предметної області та вимоги користувачів.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір середовища розробки

Розробка програмного забезпечення вимагає використання відповідного середовища, яке забезпечить зручність у написанні коду, налагодженні, тестуванні та управлінні проєктом. Одним із ключових аспектів вибору середовища є його функціональні можливості, сумісність із необхідними технологіями та підтримка розробника.

Visual Studio – це інтегроване середовище розробки (IDE), створене компанією Microsoft. Воно підтримує широкий спектр мов програмування, включаючи C#, C++, Python, JavaScript та багато інших. Visual Studio надає потужні інструменти для написання, налагодження та тестування коду, а також має вбудовану підтримку для різних фреймворків і платформ [4].

Для реалізації програмного продукту було обрано Visual Studio, оскільки це потужне IDE, яке надає широкі можливості для роботи з мовою C# та фреймворком .NET. Visual Studio пропонує зручні засоби для налагодження, підтримку контролю версій, інструменти для створення інтерфейсів та інтеграцію з базами даних. Додатковими перевагами є вбудовані засоби для роботи з технологіями WPF, що використовується у розробці графічного інтерфейсу користувача.

Особливості Visual Studio:

- зручний редактор коду – підтримка підсвічування синтаксису, автодоповнення та рефакторингу;
- вбудовані інструменти налагодження – дозволяють ефективно знаходити та виправляти помилки;
- підтримка Git та інших систем контролю версій – полегшує спільну розробку та управління кодом;
- розширюваність – можливість додавання розширень та плагінів для покращення функціональності.

2.1.1 Аналоги середовища розробки

MonoDevelop – кросплатформене середовище розробки, що підтримує C# та .NET. Воно є альтернативою Visual Studio для розробки під Linux та macOS, проте поступається в функціональності та стабільності. [5].

Переваги:

- безкоштовність;
- підтримка кросплатформеності;
- інтеграція з Xamarin для мобільної розробки.

Недоліки:

- менший набір інструментів порівняно з Visual Studio;
- обмежена підтримка нових функцій .NET, менш стабільна робота на великих проєктах.

Code::Blocks – легке та швидке IDE, що підтримує різні мови програмування, але має обмежену підтримку C# та .NET, що ускладнює розробку WPF-додатків [6].

Переваги:

- швидкість роботи;
- мінімальне споживання ресурсів;
- підтримка кількох мов програмування.

Недоліки:

- обмежена підтримка .NET;
- слабкі можливості рефакторингу;
- відсутність вбудованих засобів для розробки GUI-додатків.

JetBrains Rider – сучасне середовище розробки для .NET, що має розширені можливості рефакторингу коду, проте є платним і потребує значних ресурсів системи[7].

Переваги:

- розширені інструменти аналізу коду;
- потужна інтеграція з іншими продуктами JetBrains;

- кроссплатформеність.

Недоліки:

- висока вартість;
- значне споживання ресурсів;
- потребує стабільного інтернет-з'єднання для деяких функцій.

У таблиці 2.1 зображено порівняння середовищ розробки.

Таблиця 2.1 – Порівняння середовищ розробки

Критерії	Visual Studio	Mono Develop	Code: :Blocks	JetBrains Rider
Підтримка C# та .NET	+	+	—	+
Кроссплатформеність	+	+	—	—
Інструменти налагодження	+	—	+	+
Підтримка GUI-розробки	+	+	—	+
Підтримка багатомовності	+	—	—	+
Інтеграція з Git	+	+	—	+
Розширюваність (плагіни)	+	+	—	—

Таким чином, у ході аналізу різних середовищ розробки, таких як MonoDevelop, Code::Blocks та JetBrains Rider, можна зробити висновок, що кожне з цих середовищ має свої переваги, але для потреб розробки найкращим вибором є Visual Studio.

MonoDevelop – це непоганий варіант для розробки C# додатків, але він часто має обмежену підтримку складних проєктів і може не бути настільки зручним для великих чи багатофункціональних додатків.

Code::Blocks – хоч і є чудовим інструментом для розробки на C++, все ж не має таких потужних інструментів для розробки в C# чи WPF, що обмежує його використання в специфічних проєктах.

JetBrains Rider – це сильний інструмент для C#, але він вимагає ліцензії, що може бути непривабливим для деяких розробників, а також є важчим порівняно з іншими середовищами.

Отже, Visual Studio має найкращу підтримку для розробки C# додатків, безліч потужних інструментів для налагодження, тестування та інтеграції з різними бібліотеками та сервісами, що робить його найбільш ефективним вибором для моїх завдань. Враховуючи всі переваги, VS – це найкраще середовище для роботи, оскільки воно забезпечує зручність, стабільність та велику кількість функцій, що прискорюють процес розробки.

2.2 Вибір архітектури проєктування

Для розробки додатку було обрано архітектуру MVVM (Model-View-ViewModel), оскільки вона є однією з найефективніших для створення додатків на платформі WPF, зокрема для розділення логіки додатку та його інтерфейсу [8].

MVVM є архітектурним патерном, що дозволяє організувати код у три основні компоненти:

Model (Модель) – це частина архітектури, яка відповідає за дані та логіку бізнес-процесів. Модель може включати класи, які взаємодіють з базою даних, зберігають та обробляють інформацію, але не знають нічого про інтерфейс користувача. Вона повинна бути незалежною від View та ViewModel, що дозволяє спростити тестування та забезпечити гнучкість у зміні інтерфейсу користувача без впливу на бізнес-логіку.

View (Подання) – відповідає за відображення даних і взаємодію з користувачем. Це користувацький інтерфейс, що включає в себе всі елементи інтерфейсу (кнопки, текстові поля, списки та інші елементи управління). View отримує дані безпосередньо через прив'язку (data binding) до ViewModel, при цьому не повинна містити логіки або значного коду. Її мета – лише показувати інформацію користувачу та передавати взаємодії (наприклад, натискання

кнопок) у ViewModel.

ViewModel (Модель подання) – є проміжною ланкою між Model та View. Вона відповідає за обробку даних, які надходять від Model, і підготовку їх для відображення в View. ViewModel використовує зв'язки даних (data binding) для оновлення View, і передає вхідні дані або команди (наприклад, натискання кнопки) від View до Model. Це дозволяє зберігати View незалежним від конкретної реалізації логіки та даних, що дає змогу створювати більш підтримувані та тестовані додатки.

Особливості архітектури MVVM:

- розділення обов'язків: MVVM дозволяє чітко розділити обов'язки між компонентами додатку, що полегшує розробку та тестування;
- прив'язка даних (Data Binding): один з основних аспектів MVVM – це використання двостороннього зв'язку даних між View та ViewModel, що значно спрощує взаємодію між інтерфейсом користувача і даними;
- тестованість: оскільки ViewModel відокремлений від View, це дозволяє тестувати бізнес-логіку без необхідності взаємодії з інтерфейсом користувача, що підвищує якість коду;
- масштабованість: архітектура MVVM добре підходить для складних додатків, де потрібно підтримувати масштабованість та гнучкість при розробці нових функцій.

Вибір архітектури MVVM дозволить зберегти чистоту коду, підвищити тестованість та забезпечити чітке розділення між логікою додатку та його інтерфейсом, що в результаті сприятиме більш ефективному процесу розробки і подальшій підтримці програми.

2.2.1 Аналоги архітектури проєктування

MVP – патерн, який має схожі компоненти з MVC, але з деякими відмінностями. У цьому патерні Model відповідає за дані і бізнес-логіку, View – за відображення інтерфейсу, а Presenter – за логіку взаємодії між View і Model. Presenter отримує дані з Model і передає їх до View [9].

Переваги:

- тестованість: оскільки Presenter відокремлений від View, його можна тестувати без необхідності взаємодії з графічним інтерфейсом;
- легке керування логікою: Presenter відповідає за більшість логіки, що дозволяє зберігати View простим і зручним для взаємодії з користувачем;
- більший контроль над View: View є пасивним компонентом і не містить логіки, що дозволяє Presenter контролювати оновлення інтерфейсу.

Недоліки:

- складність у великих додатках: для великих додатків, де є багато Presenter, це може призвести до збільшення складності підтримки та масштабування проєкту;
- взаємодія з View: у порівнянні з MVVM, де використовуються зв'язки даних, MVP потребує більше зусиль для керування відображенням даних у View.

MVC є одним із найбільш популярних архітектурних патернів, який широко застосовується для створення веб-додатків та додатків на різних платформах. У цьому патерні розділяються три основні компоненти: Model, View і Controller. Model відповідає за збереження і обробку даних, View – за відображення інформації користувачу, а Controller – за обробку введених даних і команд від користувача та оновлення View [10].

Переваги:

- чітке розділення: контролер активно взаємодіє з Model і View, що дозволяє зберігати чистоту і підтримуваність коду;
- широка підтримка: MVC є стандартом для багатьох фреймворків,

таких як ASP.NET, Ruby on Rails, Angular, що дозволяє легко застосовувати цей патерн у різних технологіях;

- гнучкість: підходить для масштабних проєктів з багатьма компонентами.

Недоліки:

- тісний зв'язок між компонентами: Controller часто має великий зв'язок як з View, так і з Model, що може ускладнити тестування та модифікацію коду;

- менш зручний для складних інтерфейсів: взаємодія між View і Controller може бути не такою зручною для додатків з інтенсивною графічною частиною.

Clean Architecture є більш загальним і гнучким патерном, що розділяє додаток на шари, кожен з яких має свою роль. У цьому патерні є чотири основні шари: Entities (бізнес-логіка), Use Cases (операції з бізнес-логікою), Interface Adapters (адаптери для взаємодії з користувачем або зовнішніми системами) і Frameworks/Drivers (взаємодія з зовнішніми технологіями, такими як бази даних, UI-фреймворки і т.д.) [11].

Переваги:

- чітка ізоляція шарів: clean Architecture дозволяє чітко розділяти бізнес-логіку, інтерфейс користувача та інші компоненти, що робить код більш підтримуваним і масштабованим;

- гнучкість: цей патерн дуже підходить для великих додатків, оскільки дозволяє легко змінювати або замінювати окремі компоненти без значного впливу на інші частини системи;

- тестованість: оскільки компоненти ізольовані, тестувати їх можна незалежно один від одного.

Недоліки:

- складність: clean Architecture може бути складним для впровадження, особливо в малих або середніх проєктах, де така висока ізоляція може бути зайвою;

– перевантаження структурою: для невеликих додатків чи прототипів цей патерн може здатися надмірно складним і не завжди ефективним.

У таблиці 2.2 зображено порівняння архітектур проєктування.

Таблиця 2.2 – Порівняння архітектур проєктування

Критерії	MVVM	MVP	MVC	Clean Architecture
Простота реалізації	+	+	+	—
Прив'язка даних	+	—	—	+
Тестованість	+	+	—	—
Гнучкість змін інтерфейсу	+	—	—	+
Складність підтримки	+	+	—	—
Ізоляція компонентів	+	+	—	+
Масштабованість	+	+	+	—

Таким чином, на основі порівняння архітектур, можна зробити висновок, що MVVM є найкращим вибором для розробки мобільних додатків, особливо для таких платформ, як WPF, де важливе ефективне розділення логіки та інтерфейсу, а також зручність тестування через ViewModel. Ця архітектура дозволяє реалізувати двосторонню прив'язку даних, що робить розробку більш ефективною та зручною для користувачів.

Інші архітектури, такі як MVC, MVP та Clean Architecture, мають свої переваги, проте вони не є такими ідеальними для розробки мобільних додатків. MVC підходить для веб-додатків, але має тісну залежність між контролером і View, що ускладнює підтримку та тестування. MVP також підходить для додатків з активною взаємодією користувача, але вимагає більше зусиль для підтримки та адаптації. Clean Architecture, хоч і є надзвичайно гнучким і масштабованим рішенням, підходить більше для

великих складних систем, що робить її менш підходящою для простих або середньої складності мобільних додатків.

Отже, для розробки мобільних додатків MVVM є найкращим вибором, оскільки забезпечує зручне та ефективне управління даними та інтерфейсами, залишаючи інші архітектури більш підходящими для інших типів проєктів.

2.3 Вибір бази даних

MongoDB – це не реляційна (NoSQL) база даних, що зберігає дані у вигляді документів у форматі JSON (BSON), що забезпечує високу гнучкість в організації даних. Замість традиційних таблиць, як у реляційних базах даних, MongoDB використовує колекції для зберігання документів, де кожен документ може мати різну структуру, що дозволяє легко змінювати схему даних без необхідності вносити зміни до всієї бази даних [12].

MongoDB підтримує горизонтальне масштабування, що дозволяє легко масштабувати додаток, додавати нові сервери або обробляти великі обсяги даних за допомогою шардінгу. Це робить її ідеальним вибором для сучасних веб та мобільних додатків, які потребують масштабованості та високої доступності.

Особливості MongoDB:

- гнучкість схеми: однією з головних особливостей MongoDB є відсутність суворої схеми даних. Кожен документ може мати власну структуру, що дозволяє змінювати структуру даних без порушення цілісності всіх записів у базі;
- висока масштабованість: MongoDB підтримує горизонтальне масштабування через шардінг, що дозволяє ефективно працювати з великими обсягами даних і зберігати дані на декількох серверах;
- простота використання: завдяки своїй простій архітектурі та API, MongoDB легко інтегрується в сучасні додатки, що робить її популярною серед розробників. Її використання спрощує роботу з великими обсягами

непов'язаних даних;

- підтримка складних запитів: MongoDB дозволяє створювати складні запити, зокрема індексацію, агрегацію, та пошук за різними критеріями, що забезпечує високу швидкість роботи з великими обсягами даних;

- висока доступність: MongoDB підтримує реплікацію даних на кілька серверів, що забезпечує високу доступність та відмовостійкість, що критично важливо для мобільних додатків з великим потоком користувачів;

- JSON-подібний формат даних: оскільки дані в MongoDB зберігаються у форматі BSON (подібному JSON), це забезпечує легкість роботи з даними та інтеграцію з багатьма сучасними веб-технологіями та фреймворками;

- підтримка транзакцій: хоча MongoDB традиційно не підтримує ACID транзакції, з останніми версіями вона має можливість підтримки транзакцій на кількох документах, що робить її більш гнучкою для складних застосунків.

2.3.1 Аналоги баз даних

Окрім MongoDB, існує кілька інших популярних баз даних, які також можуть бути використані для розробки мобільних додатків. Розглянемо три основні аналоги: MySQL, PostgreSQL та Cassandra. Кожна з цих баз даних має свої особливості, переваги та недоліки.

MySQL – це реляційна база даних, яка використовує таблиці для зберігання даних. Вона є однією з найпопулярніших реляційних баз даних, що широко застосовується в веб-розробці та інших додатках, що потребують високої продуктивності та зручного управління даними [13].

Переваги:

- стандартна SQL – архітектура: завдяки SQL MySQL є стандартним вибором для багатьох розробників, що працюють з реляційними базами даних.

- висока надійність: MySQL відома своєю стабільністю та здатністю обробляти великі обсяги запитів;
- підтримка транзакцій: підтримує транзакції з ACID властивостями, що робить її хорошим вибором для додатків, де важлива точність операцій;
- поширеність і підтримка: базу даних підтримує велика кількість інструментів і ресурсів для розробників.

Недоліки:

- жорстка схема: реляційна природа MySQL вимагає суворої схеми, що ускладнює зміни структури даних без порушення цілісності;
- масштабованість: MySQL може бути менш ефективною в плані горизонтального масштабування порівняно з NoSQL базами даних, як MongoDB;
- не підходить для складних, нереляційних даних: для мобільних додатків з динамічними або нереляційними даними MySQL може бути не найкращим вибором.

PostgreSQL – це потужна об'єктно-реляційна база даних з відкритим вихідним кодом. Вона підтримує реляційну архітектуру, але також дозволяє зберігати та обробляти складніші типи даних, що робить її гнучкою і потужною базою даних [14].

Переваги:

- підтримка складних типів даних: PostgreSQL дозволяє працювати з не тільки реляційними, але й складними типами даних, такими як JSON, що дозволяє їй успішно поєднувати реляційну та NoSQL природу;
- масштабованість і продуктивність: PostgreSQL підтримує горизонтальне масштабування і високопродуктивну обробку запитів;
- сумісність з SQL: як і MySQL, PostgreSQL використовує SQL, але додає додаткові функції для роботи з складними запитами.

Недоліки:

- складність в налаштуванні: встановлення та конфігурація

PostgreSQL може бути складнішою порівняно з іншими базами даних;

- більш важка інтеграція з NoSQL: хоча PostgreSQL підтримує JSON, її інтеграція з NoSQL рішеннями не є такою ж ефективною, як у MongoDB.

Cassandra – це розподілена база даних NoSQL, спеціально розроблена для роботи з великими обсягами даних у реальному часі. Вона ідеально підходить для обробки даних, які часто змінюються, і для додатків, які потребують високої доступності та масштабованості [15].

Переваги:

- горизонтальне масштабування: Cassandra дозволяє горизонтально масштабувати базу даних без втрати продуктивності;
- висока доступність і відмовостійкість: завдяки реплікації та зберігання даних на декількох серверах, cassandra забезпечує високу доступність і відмовостійкість;
- підтримка великих даних: ідеально підходить для додатків, які потребують обробки великих обсягів інформації в реальному часі.

Недоліки:

- складна архітектура: налаштування і підтримка Cassandra може бути складною через її розподілену природу;
- не підтримує транзакції: Cassandra не підтримує повні Acid транзакції, що може бути проблемою для додатків, де потрібна повна точність у виконанні операцій;
- слабша підтримка складних запитів: у порівнянні з реляційними базами даних, Cassandra обмежує можливості для складних SQL-запитів, оскільки використовує власну мову запитів.

У таблиці 2.3 зображено порівняння баз даних.

Таблиця 2.3 – Порівняння баз даних

Критерії	MongoDB	MySQL	PostgreSQL	Cassandra
Масштабованість	+	—	—	+
Продуктивність	+	+	+	—
Підтримка транзакцій	+	+	—	+
Гнучкість запитів	+	+	—	+
Сумісність з JSON	+	—	+	+
Інтеграція з фреймворками	+	+	—	+
Керування та адміністрування	+	+	—	—
Підтримка реплікації	+	+	+	—

Таким чином, на основі порівняння, MongoDB є найкращим вибором для проєкту з оренди автомобілів. Вона забезпечує високу масштабованість, гнучку схему даних та зручність роботи з великими обсягами даних, що є важливими вимогами для мобільного додатку. Крім того, MongoDB добре інтегрується з сучасними технологіями та фреймворками, що полегшує процес розробки.

MySQL та PostgreSQL також мають свої переваги, такі як підтримка складних SQL-запитів та повна підтримка транзакцій з ACID властивостями. Однак, їх обмежена масштабованість та жорстка схема даних можуть стати перешкодою для ефективної роботи з динамічними даними та великою кількістю користувачів у реальному часі.

Cassandra, хоча й підходить для обробки великих розподілених даних, має складну архітектуру і не підтримує ACID транзакції, що робить її менш зручною для проєкту, де важливі та гнучкість.

Отже, MongoDB найкраще підходить, а інші бази даних можуть бути корисними в інших контекстах, але вони не відповідають специфіці проєкту.

2.4 Вибір мови програмування

C# – це об'єктно-орієнтована мова програмування, розроблена компанією Microsoft як частина .NET Framework. Вона широко використовується для створення різноманітних додатків, включаючи веб-додатки, настільні програми та мобільні рішення. C# є мовою з сильною типізацією, що забезпечує високий рівень безпеки типів, і надає розробникам потужні можливості для створення надійного і продуктивного коду [16].

Основні особливості мови C#:

- об'єктно-орієнтованість: C# підтримує принципи об'єктно-орієнтованого програмування, що дозволяє створювати гнучкі, масштабовані та модульні програми;
- підтримка асинхронного програмування: мова надає потужні механізми для асинхронного програмування через використання `async` та `await`, що дозволяє ефективно обробляти введення-виведення та інші операції, не блокуючи головний потік;
- безпека типів: C# забезпечує перевірку типів під час компіляції, що знижує кількість помилок у програмі та підвищує надійність;
- підтримка багатозадачності: мова дозволяє ефективно працювати з багатозадачністю та паралельними обчисленнями, що є важливим для сучасних додатків;
- гарна інтеграція з .net екосистемою: C# має відмінну сумісність з іншими інструментами і бібліотеками .net, що полегшує розробку та підтримку програм.

Для розробки додатку було обрано C# як мову програмування на платформі WPF (Windows Presentation Foundation). WPF – це потужна технологія для створення графічних інтерфейсів користувача в Windows, яка дозволяє легко створювати сучасні та привабливі інтерфейси. WPF підтримує декларування інтерфейсу через XAML, що дозволяє ефективно розділяти логіку програми та її презентаційний шар. Також WPF надає багаті можливості

для роботи з графікою, анімаціями та інтерактивними елементами, що дозволяє створювати високоякісні інтерфейси для додатків.

Завдяки потужним можливостям C# і WPF, ця комбінація є ідеальним вибором для розробки сучасного, масштабованого та зручного для користувача мобільного додатку.

2.4.1 Аналоги мови програмування

Java – це об'єктно-орієнтована мова програмування, яка відома своєю платформонезалежністю завдяки технології Java Virtual Machine (JVM). Вона широко використовується для розробки веб-додатків, мобільних програм (особливо для Android) і настільних програм [17].

Переваги:

- платформонезалежність завдяки jvm;
- висока продуктивність у середовищі серверних додатків;
- багатий набір бібліотек і фреймворків для розробки;
- підтримка багатозадачності та паралельного програмування.

Недоліки:

- вища складність порівняно з C# для роботи з графічними інтерфейсами користувача;
- потребує більше ресурсів для роботи порівняно з іншими мовами;
- менше інтеграцій з сучасними інструментами для розробки настільних додатків.

Python – це мова програмування високого рівня, що широко використовується в розробці веб-додатків, науці про дані, штучному інтелекті та автоматизації. Вона відома своєю простотою та читаємністю коду [18].

Переваги:

- легкість у навчанні та використанні завдяки простому синтаксису;
- широке використання в науці про дані, штучному інтелекті та машинному навчанні;

- багатий набір бібліотек для різноманітних застосувань;
- підходить для швидкого прототипування.

Недоліки:

- менша продуктивність порівняно з `C#` через інтерпретовану природу;
- не так підходить для розробки великих настільних додатків, як `C#`;
- обмежена підтримка для мобільних додатків в порівнянні з іншими мовами.

`C++` – це мова програмування низького рівня, яка надає великий контроль над пам'яттю і виконується з високою продуктивністю. Вона використовується для розробки ігор, системних програм, а також вбудованих систем [19].

Переваги:

- висока продуктивність та ефективність;
- гнучкість і контроль над ресурсами;
- широко використовується в розробці ігор та системних додатків.

Недоліки:

- вища складність порівняно з `C#` для розробки сучасних інтерфейсів користувача;
- займає більше часу для розробки, порівняно з мовами вищого рівня;
- вимагає детальнішого управління пам'яттю, що збільшує складність коду.

У таблиці 2.4 зображено порівняння мов програмування.

Таблиця 2.4 – Порівняння мов програмування

Критерії	C# (WPF)	Java	Python	C++
Простота синтаксису	+	—	+	—
Продуктивність	+	+	—	+
Розробка інтерфейсу	+	—	—	—
Мобільна розробка	+	+	—	—
Підтримка багатозадачності	+	+	—	+
Платформонезалежність	+	+	+	—

Таким чином, для розробки проєкту було обрано мову програмування C# з платформою WPF, що є найкращим вибором завдяки своїй зручності, високій продуктивності та можливостям створення сучасних настільних додатків з інтуїтивно зрозумілим графічним інтерфейсом.

Інші мови програмування, такі як Java, Python та C++, мають свої переваги та застосовуються в інших типах проєктів, проте не підходять для моєї розробки. Java більше підходить для серверних рішень, Python оптимальний для наукових розробок та швидкого прототипування, а C++ використовується для програм, де потрібна максимальна продуктивність. Проте, для створення додатка з графічним інтерфейсом та інтеграцією різних функцій, C# (WPF) є найбільш підходящим та ефективним варіантом.

2.5 Висновки за розділом 2

У процесі розробки проєкту було здійснено ретельний аналіз доступних інструментів і методів для вибору найбільш підходящих рішень. Зокрема, було обрано середовище розробки Visual Studio, яке є потужним і зручним інструментом для створення програмного забезпечення на платформі C# з використанням WPF. Цей вибір забезпечує максимальну інтеграцію з

потрібними інструментами для розробки настільних додатків, дозволяючи значно прискорити процес розробки та тестування.

Що стосується архітектури проєкту, було обрано MVVM, оскільки ця архітектурна модель дозволяє чітко розділити представлення, бізнес-логіку та дані, що забезпечує високу гнучкість, зручність у підтримці та тестуванні. У порівнянні з іншими архітектурами, MVVM є оптимальним вибором для розробки з використанням WPF, оскільки дозволяє максимально ефективно працювати з двостороннім зв'язком між View і ViewModel.

Базу даних було обрано MongoDB, яка є потужною NoSQL базою даних, що дозволяє ефективно працювати з великими обсягами неструктурованих даних. Вибір MongoDB був обумовлений вимогами до швидкості обробки запитів і гнучкості в роботі з даними, оскільки цей тип бази даних дозволяє зберігати дані в JSON-подібному форматі, що є зручним для роботи з різноманітними даними.

Мовою програмування було обрано C# на платформі WPF. Цей вибір забезпечує високу продуктивність, простоту використання та зручність при створенні графічних інтерфейсів, що є критично важливим для створення сучасних настільних додатків з інтуїтивно зрозумілим користувацьким інтерфейсом.

У підсумку, всі технології, які були обрані для реалізації проєкту, максимально відповідають його вимогам, забезпечують ефективну роботу з даними, зручність при розробці та тестуванні, а також можливість створення масштабованого та високопродуктивного програмного продукту.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

1.1 Функціональні можливості розробленого програмного забезпечення

Розроблена програма «Vroomie» є додатком для оренди автомобілів, що надає користувачам можливість швидко та зручно бронювати автомобілі через інтуїтивно зрозумілий інтерфейс. Основні функціональні можливості програми включають:

- а) реєстрація та авторизація:
 - 1) створення нового облікового запису;
 - 2) авторизація користувача через email та пароль.
- б) головний екран та пошук автомобілів:
 - 1) відображення списку доступних автомобілів;
 - 2) фільтрація за маркою;
 - 3) перегляд детальної інформації про автомобіль.
- в) бронювання автомобіля:
 - 1) вибір автомобіля та вказання параметрів оренди (дата, місце отримання, з водієм чи без);
 - 1) перегляд підсумку оренди перед підтвердженням;
 - 2) вибір способу оплати (Apple Pay або введення картки).
- г) управління бронюванням:
 - 1) перегляд активних оренд у профілі користувача.
- д) керування профілем:
 - 1) редагування особистих даних та зміна аватара;
 - 2) додавання та керування способами оплати;
 - 3) перегляд історії оренд;
 - 4) оцінювання додатку та залишення відгуків.
- е) адміністративна панель:
 - 1) додавання нових автомобілів у базу;

- 2) видалення автомобілів із системи;
- 3) відповіді на запити користувачів.

3.2 Загальна архітектура схеми програми

Розроблена система «Vroomie» побудована на основі архітектурного патерну MVVM (Model-View-ViewModel), що дозволяє розділити логіку, представлення та обробку даних, забезпечуючи модульність та зручність розширення.

Основні компоненти системи.

Model (Модель):

- відповідає за обробку даних та взаємодію з базою даних MongoDB;
- містить класи, що представляють основні сутності (автомобілі, користувачі, бронювання, бонуси).

View (Представлення):

- включає XAML-файли, що визначають графічний інтерфейс користувача;
- зв'язується з ViewModel через механізм прив'язки даних (data binding).

ViewModel (Модель представлення):

- містить логіку управління даними та передає їх між Model і View;
- використовує команди (Relaycommand) для взаємодії з інтерфейсом.

База даних MongoDB:

- NoSQL - база, що зберігає дані про користувачів, автомобілі, бронювання та бонуси;
- взаємодія з нею здійснюється через MongoClient.driver у Model.

Служби та допоміжні компоненти:

- інтеграція з картографічним сервісом для відображення авто;
- система оплати (Apple Pay або картки);

- контроль доступу користувачів (авторизація, реєстрація).

На рисунку 3.1 зображено архітектуру MVVM додатку.

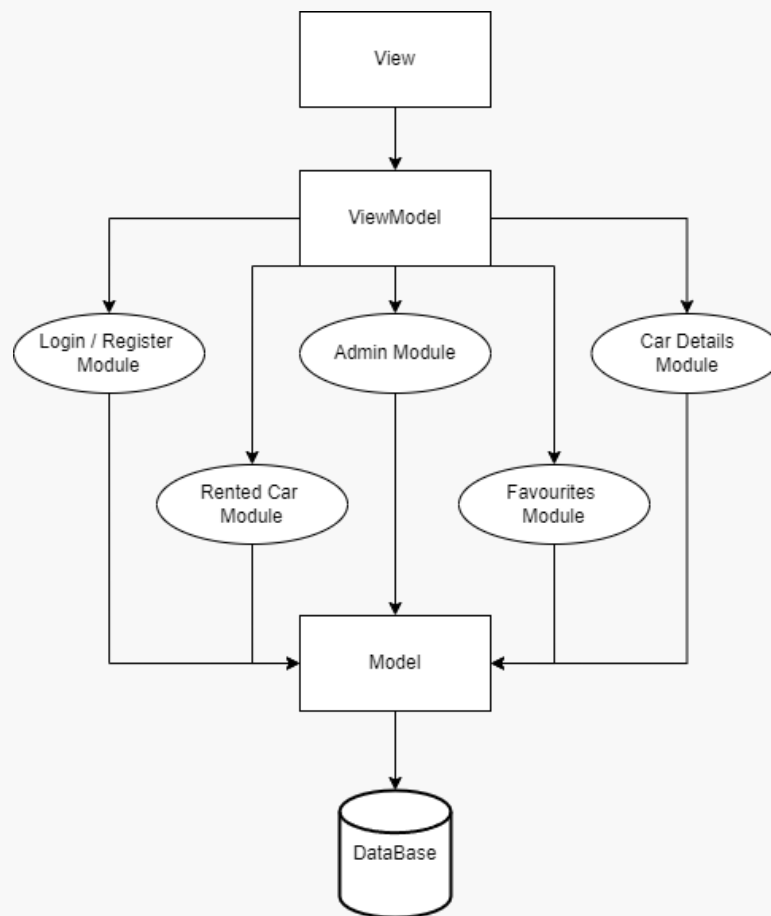


Рисунок 3.1 – Архітектура MVVM додатку

3.3 Принципи роботи програмного забезпечення

Програмне забезпечення «Vroomie» було розроблено з урахуванням найсучасніших підходів до створення ефективних та зручних для користувача систем оренди авто. Основна мета полягає в тому, щоб забезпечити користувачам інтуїтивно зрозумілий інтерфейс, швидку навігацію та надійну роботу на всіх етапах використання системи. Враховано принципи розподіленої обробки даних, інтеграцію з платіжними системами, а також ефективне управління даними користувачів і автомобілями. Вся система побудована на модульній архітектурі, що дозволяє легко розширювати

функціональність і адаптувати програму до змін в умовах ринку.

Функціональну схему зображено на рисунку 3.2.

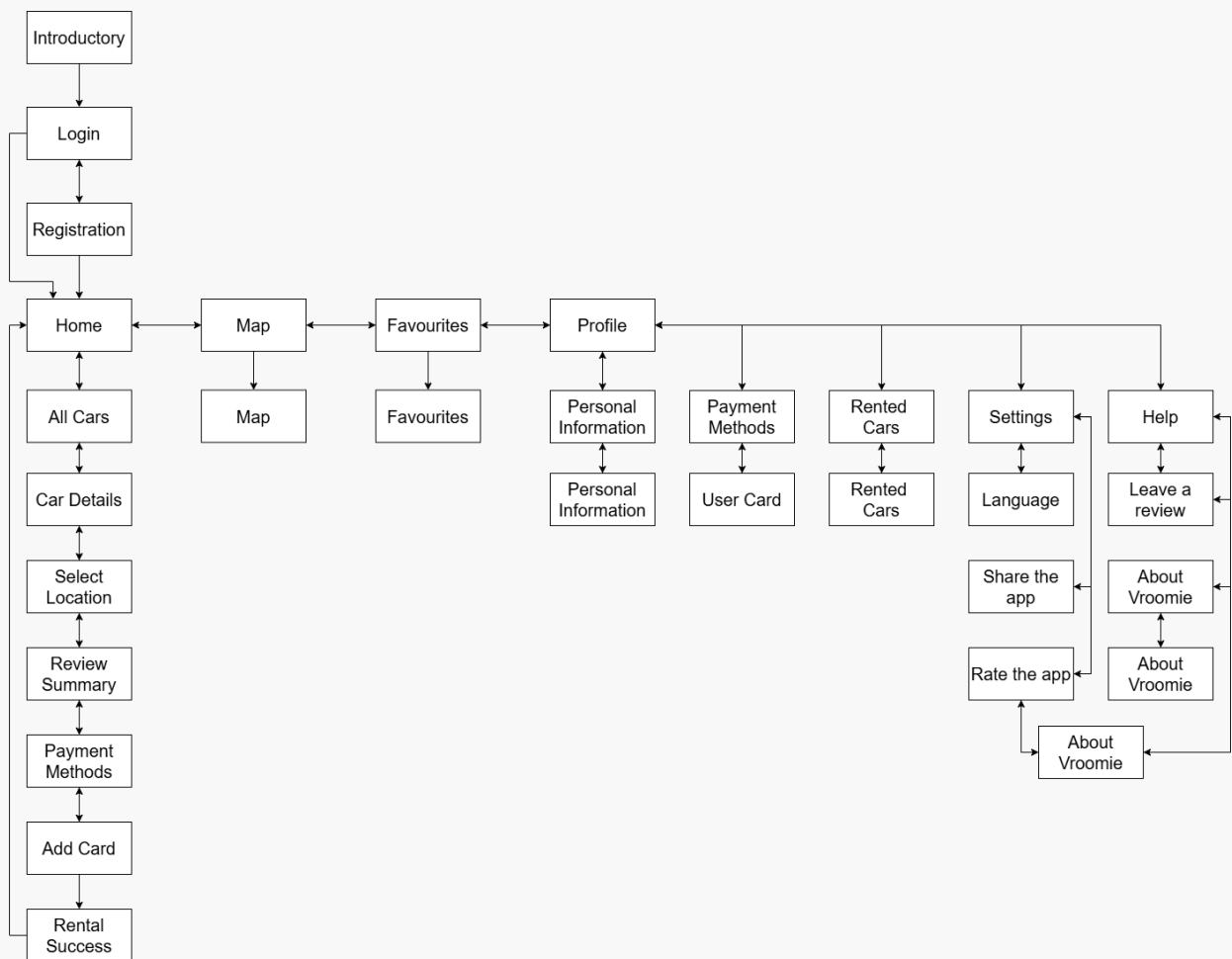


Рисунок 3.2 – Функціональна схема додатку «Vroomie»

Нижче наведено опис кожного етапу алгоритму користувача, який зображено на рисунку 3.2:

- Introductory: вітальна сторінка додатку;
- Login: вхід до акаунту користувача за поштою та паролем;
- Registration: реєстрація нового користувача;
- Home: головна сторінка додатку, на якій відображається: блок з популярними авто для оренди, горизонтальний список з різними брендами, меню навігації, кнопка профілю та кнопка пошуку авто;
- AllCars: сторінка додатку, на якій відображаються тільки ті марки

авто, які обрав користувач (Audi – відображається лише Audi);

- Car Details: сторінка додатку, на якій зображено характеристики автомобіля, який обрав користувач;

- Select Location: сторінка додатку, на якій зображено вибір дати та часу оренди авто. Також присутній вибір оренди (з водієм чи без);

- Review Summary: сторінка додатку, на якій зображено кінцеву інформацію перед оплатою авто: дату отримання та повернення авто, місце отримання, тип оренди, ціну за день, кінцеву суму оренди та кнопку переходу до вибору методу оплати;

- Payment Methods: сторінка додатку, на якій зображено вибір методу оплати оренди. На вибір доступні два методи: Apple Pay чи додавання власної банківської карти. При натисканні на додавання карти відкриється нова сторінка;

- Add card: сторінка додатку, на якій зображено додавання нової карти користувача. На ній зображено: поле вводу номеру карти, кінцевої дати, CVV код та ПІБ власника карти. Також присутня кнопка оплати;

- Rental Success: сторінка додатку, на яку переходить користувач після успішної оплати авто, на якій зображено текст успішної оплати та кнопку повернення до головного меню додатку;

- Map: сторінка додатку, на якій зображено мапу, локацію користувача та доступні авто для оренди;

- Favoutites: сторінка додатку, на якій зображено усі авто користувача, які він додав до свого списку улюблених авто;

- Profile: сторінка додатку, на якій зображено: привітання користувача, номер телефону, кнопку зміни фото профілю, кнопку зміни персональної інформації, кнопку перегляду збережених карток, кнопку перегляду орендованих авто, кнопку налаштувань, кнопку зв'язку з адміністрацією;

- Personal Information: сторінка додатку, на якій зображено можливість зміни персональної інформації користувача;

- Payment Methods: сторінка додатку, на якій зображено всі збережені картки користувача;
- Rented Cars: сторінка додатку, на якій зображено всі орендовані авто користувача;
- Settings: сторінка додатку, на якій зображено: кнопку ввімкнення та вимкнення повідомлень, кнопку зміни мови, кнопку «поділитись додатком» та кнопку оцінки додатку;
- Help: сторінка додатку, на якій користувач може почати чат з адміністратором для вирішення його запитань;
- Rate the App: сторінка додатку, на якій користувач може оцінити додаток по 5-ти зірковій шкалі та написати свій відгук;
- About Vroomie: сторінка додатку, на якій користувач може ознайомитись з інформацією про сам додаток та розробника.

Функціональну схему адміністратора зображено на рисунку 3.3.

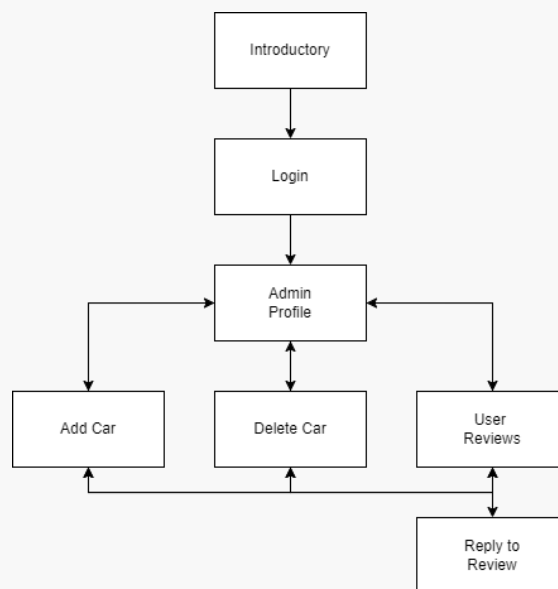


Рисунок 3.3 – Функціональна схема додатку «Vroomie»

Нижче наведено опис кожного етапу алгоритму адміністратора, який зображено на рисунку 3.3:

- Introductory: вітальна сторінка додатку;

- Login: вхід до акаунту адміна за спеціальними даними;
- Admin Profile: сторінка додатку, на якій зображено текст з функціями, які повинен виконувати адміністратор та кнопку виходу з акаунту. Також присутня навігаційна панель, на якій зображено такі кнопки: додавання та видалення авто, відповіді на запитання користувачів, профіль адміна;
- Add car: сторінка додатку, на якій адмін додає нові авто до бази даних;
- Delete car: сторінка додатку, на якій адмін видаляє вже додані авто до бази;
- User Reviews: сторінка додатку, на якій відображено список з активними запитаннями, на які ще не було надано відповіді;
- Reply to Review: сторінка додатку, на якій адмін надає відповідь на обране запитання.

3.4 Вибір шаблону для архітектури

У цьому підрозділі буде розглянуто вибір архітектурного шаблону, який був використаний для розробки програмного забезпечення оренди авто «Vroomie». Архітектура програми визначає основні принципи організації та взаємодії її компонентів, а також забезпечує ефективність, масштабованість та зручність у подальшій підтримці та розвитку системи. Вибір шаблону архітектури базувався на потребах проєкту, таких як інтеграція з платіжними системами, управління даними користувачів та автомобілів, а також забезпечення високої швидкодії та надійності роботи програми.

3.4.1 Діаграма класів

Діаграма класів є важливим елементом для розуміння структури програмного забезпечення, оскільки вона демонструє основні класи системи, їх атрибути та методи, а також взаємозв'язки між ними. У контексті програми

оренди авто «Vroomie», діаграма класів відображає ключові об'єкти, такі як користувачі, автомобілі, операції з оренди та платіжні системи, а також зв'язки між цими елементами.

Рисунки 3.4 – 3.6 демонструють структуру класів програми, зокрема показують, як взаємодіють основні компоненти для забезпечення функціональності додатку. Діаграма допомагає зрозуміти, як кожен клас впливає на роботу системи та як здійснюється передача даних між різними частинами програми.

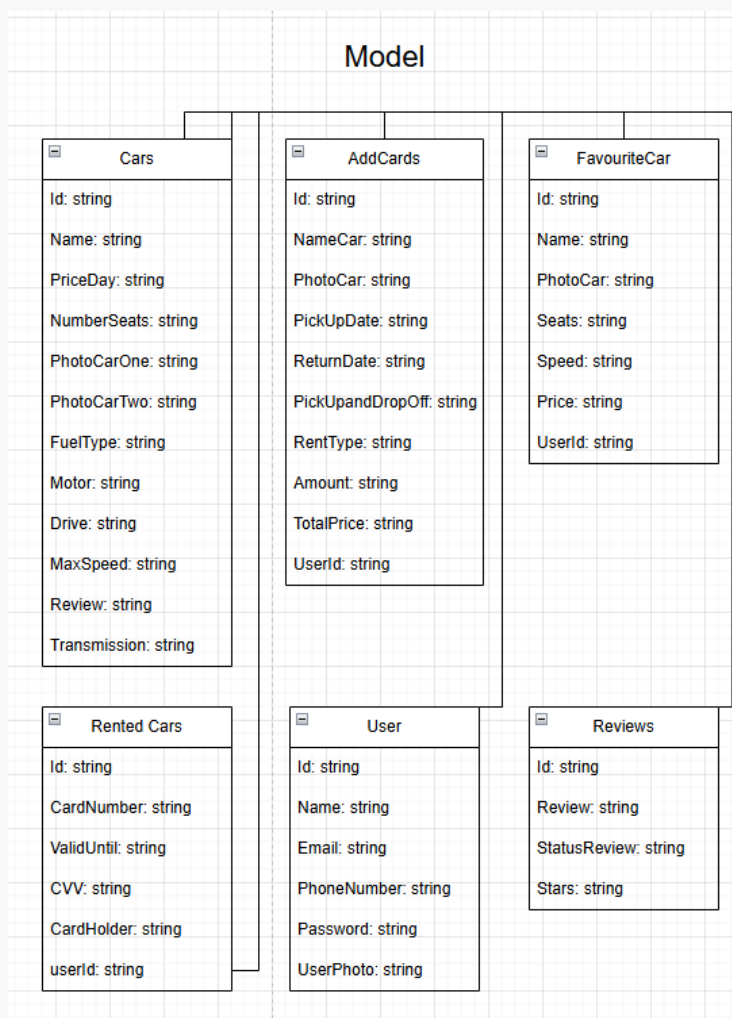


Рисунок 3.4 – Діаграма класів додатку Model

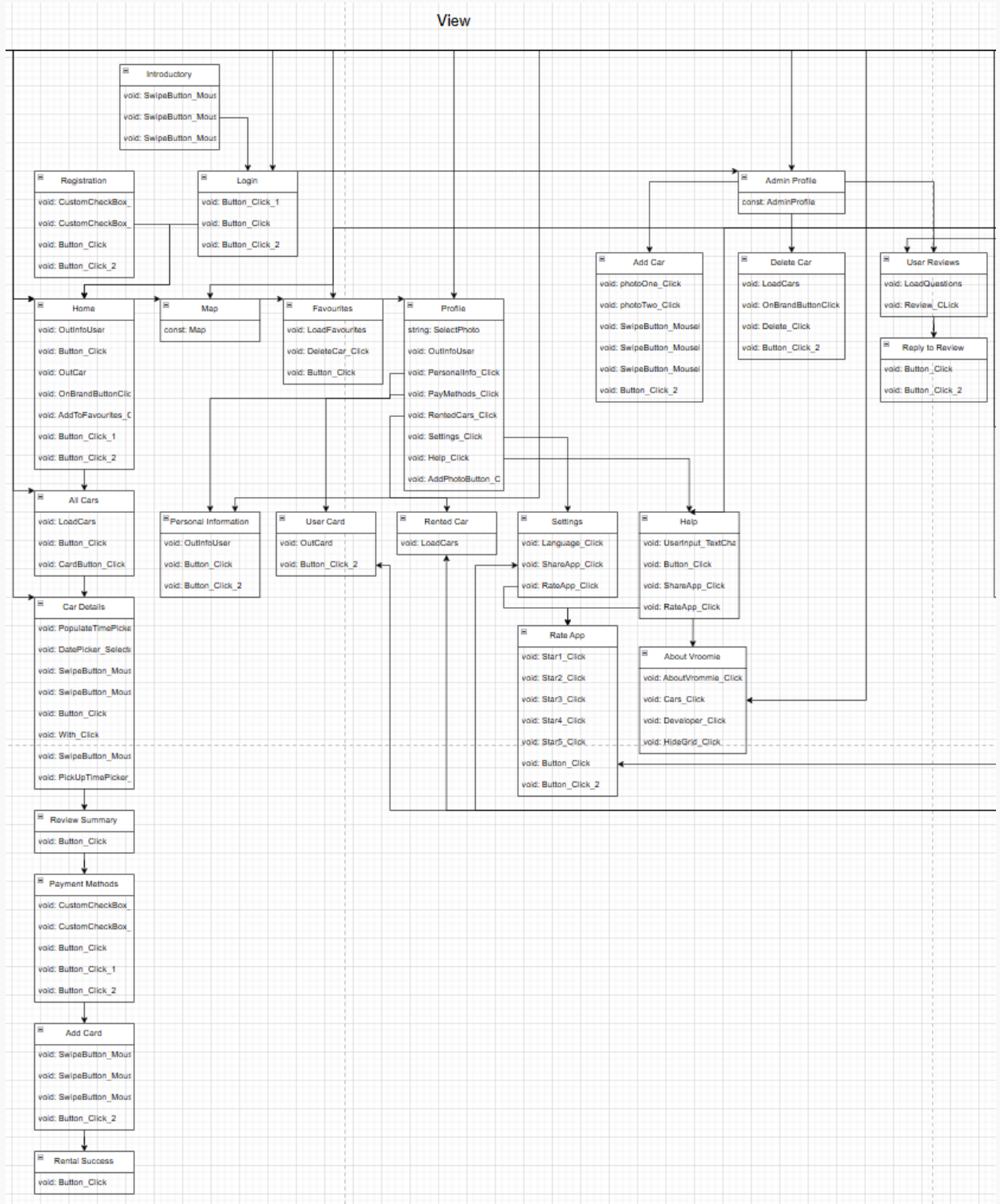


Рисунок 3.5 – Діаграма класів додатку View

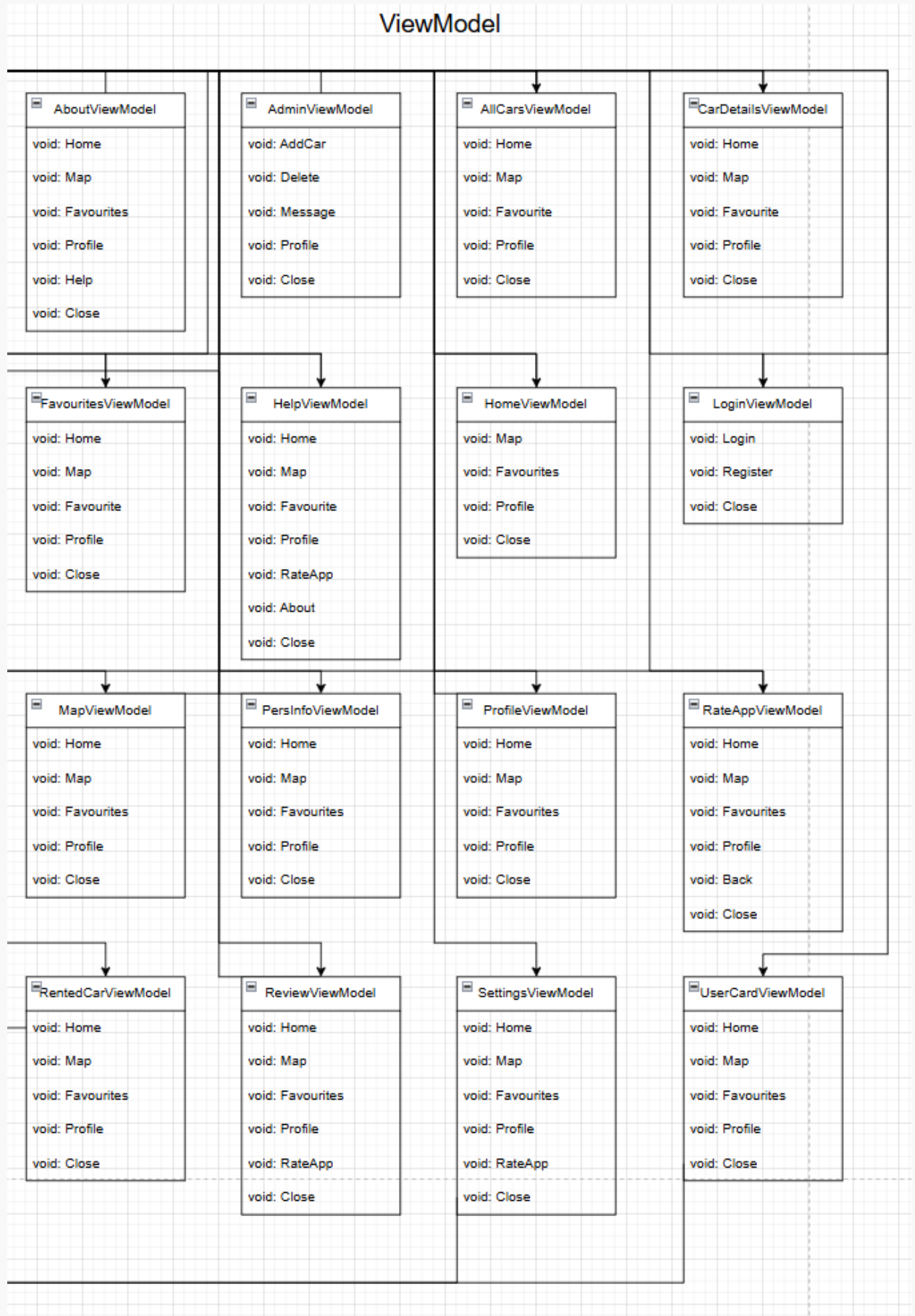


Рисунок 3.6 – Діаграма класів додатку ViewModel

3.4.2 Візуалізація компонентів системи

Візуалізація компонентів системи є важливою для розуміння структури та взаємодії основних частин програми. Вона демонструє, як окремі модулі та компоненти, такі як інтерфейс користувача, база даних, платіжні системи та серверна частина, взаємодіють між собою для забезпечення безперебійної роботи додатку оренди авто «Vroomie».

На рисунку 3.7 зображено візуалізацію компонентів системи. Ця візуалізація дає чітке уявлення про те, як компоненти взаємопов'язані, а також дозволяє краще зрозуміти процеси, що відбуваються в програмі на різних рівнях – від обробки запитів користувача до взаємодії з базою даних і зовнішніми сервісами.

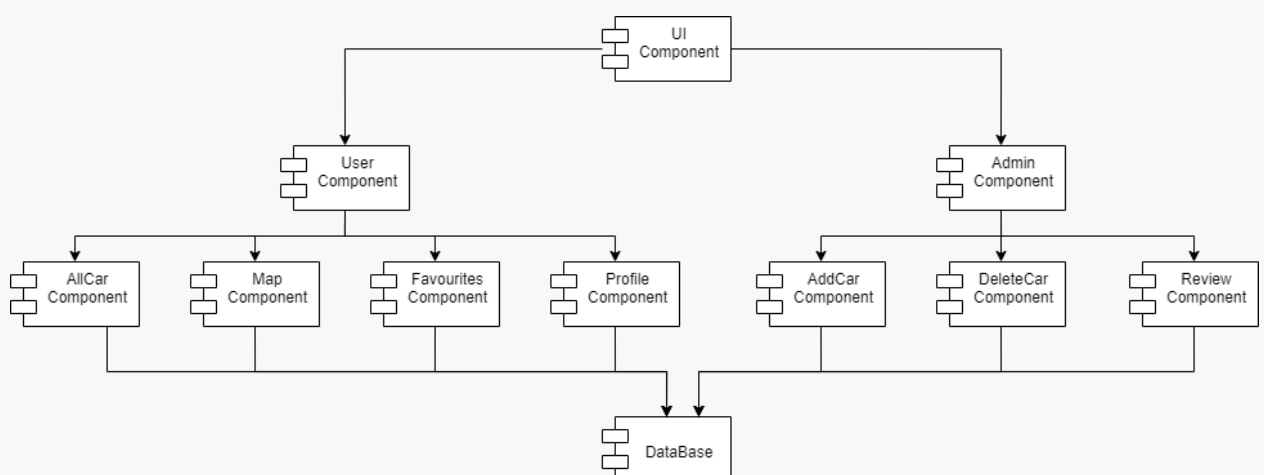


Рисунок 3.7 – Діаграма візуалізації компонентів системи

3.4.3 Діаграма взаємодії основних елементів

Діаграма взаємодії основних елементів системи демонструє, як різні компоненти програми взаємодіють один з одним під час виконання основних функцій, таких як реєстрація користувача, вибір автомобіля, оформлення оренди та здійснення платежу. Ця діаграма є важливою для розуміння процесів, що відбуваються в системі на рівні взаємодії між класами та

модулями.

На рисунку 3.8 зображено діаграму взаємодії основних елементів системи. Вона показує, як дані передаються між компонентами, наприклад, коли користувач обирає автомобіль, система взаємодіє з базою даних для перевірки наявності машини, а також здійснює обробку запиту на оплату через платіжні сервіси. Діаграма допомагає уявити логіку роботи системи та взаємозв'язок між її основними елементами.

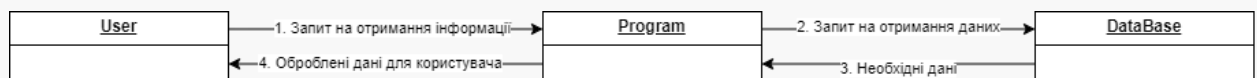


Рисунок 3.8 – Діаграма взаємодії основних елементів системи

3.5 Концептуальна модель бази даних

Концептуальна модель бази даних є основою для проєктування та організації зберігання даних у системі оренди авто «Vroomie». Вона відображає основні сутності, які зберігаються в базі даних, їх атрибути та взаємозв'язки між ними. Ця модель дозволяє забезпечити цілісність та узгодженість даних, а також ефективно організувати процеси взаємодії програми з базою даних.

На рисунку 3.9 зображено концептуальну модель бази даних. Модель включає таблиці для зберігання інформації про користувачів, автомобілі, оренди, платіжні системи та інші компоненти, що забезпечують функціонування програми. Зокрема, концептуальна модель допомагає зрозуміти, як дані з різних частин системи зберігаються, як вони пов'язані між собою та як відбувається їх обробка.

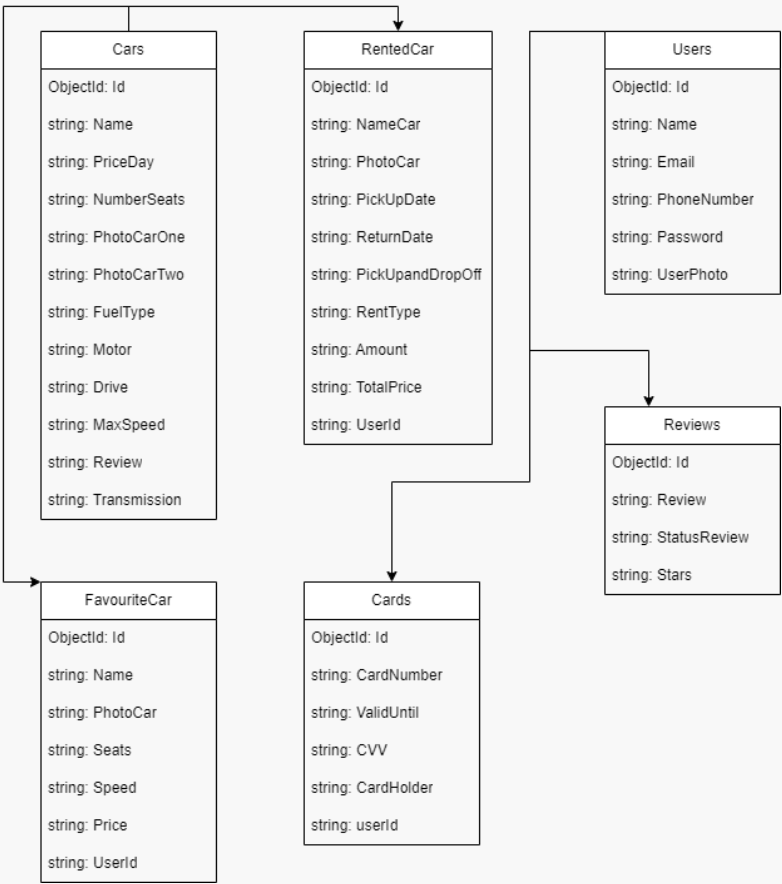


Рисунок 3.9 – Концептуальна модель бази даних

Сутність Cars – опис полів сутностей та їх типу наведено у таблиці 3.1.

Таблиця 3.1 – Опис полів сутності Cars

№	Назва поля	Тип даних
1	2	3
	_id	Objectid
1	Name	String
2	PriceDay	String
3	NumberSeats	String
4	PhotoCarOne	String
5	PhotoCarTwo	String
6	FuelType	String
7	Motor	String

Продовження таблиці 3.1

1	2	3
8	Drive	String
9	MaxSpeed	String
10	Review	String
11	Transmission	String

Сутність RentedCar – опис полів сутностей та їх типу наведено у таблиці 3.2.

Таблиця 3.2 – Опис полів сутності RentedCar

№	Назва поля	Тип даних
	_id	Objectid
1	NameCar	String
2	PhotoCar	String
3	PickUpDate	String
4	ReturnDate	String
5	PickUpandDropOff	String
6	RentType	String
7	Amount	String
8	TotalPrice	String
9	UserId	String

Сутність Users – опис полів сутностей та їх типу наведено у таблиці 3.3.

Таблиця 3.3 – Опис полів сутності Users

№	Назва поля	Тип даних
	_id	Objectid
1	Name	String
2	Email	String
3	PhoneNumber	String
4	Password	String
5	UserPhoto	String

Сутність FavouriteCar – опис полів сутностей та їх типу наведено у таблиці 3.4.

Таблиця 3.4 – Опис полів сутності FavouriteCar

№	Назва поля	Тип даних
	_id	Objectid
1	Name	String
2	PhotoCar	String
3	Seats	String
4	Speed	String
5	Price	String
6	UserId	String

Сутність Cards – опис полів сутностей та їх типу наведено у таблиці 3.5.

Таблиця 3.5 – Опис полів сутності Cards

№	Назва поля	Тип даних
	_id	Objectid
1	CardNumber	String
2	ValidUntil	String
3	CVV	String
4	CardHolder	String
5	userId	String

Сутність Reviews – опис полів сутностей та їх типу наведено у таблиці 3.6.

Таблиця 3.6 – Опис полів сутності Reviews

№	Назва поля	Тип даних
	_id	Objectid
1	Review	String
2	StatusReview	String
3	Stars	String

3.6 Взаємодія модулів системи

Взаємодія модулів системи є важливим аспектом для забезпечення безперебійної роботи додатку оренди авто «Vroomie». Це розділ визначає, як різні модулі та компоненти системи взаємодіють один з одним для виконання основних функцій, таких як обробка запитів користувача, взаємодія з базою даних, виконання платіжних операцій та відображення інформації на інтерфейсі користувача. Кожен модуль виконує певну задачу в рамках загального процесу.

Ця взаємодія забезпечує інтеграцію всіх компонентів системи та

дозволяє досягти високої ефективності та зручності в роботі програми. Взаємодія модулів є ключовим елементом для досягнення високої продуктивності та стабільності роботи додатку в цілому.

3.6.1 Діаграма випадків використання

Діаграма випадків використання є важливим інструментом для візуалізації основних сценаріїв взаємодії користувачів з системою оренди авто «Vroomie». Вона описує, як користувачі взаємодіють із системою через різні функціональні елементи, такі як реєстрація, вибір автомобіля, оформлення оренди та здійснення платежів. Всі ці сценарії відображаються на діаграмі у вигляді випадків використання, що показують взаємодію користувачів з системою в різних контекстах.

На рисунку 3.10 зображено діаграму випадків використання. Ця діаграма допомагає зрозуміти основні функціональні вимоги програми, а також визначити, які конкретні дії може виконувати користувач. Вона також дозволяє ефективно планувати розвиток системи та вдосконалення функцій, орієнтуючись на потреби кінцевих користувачів.

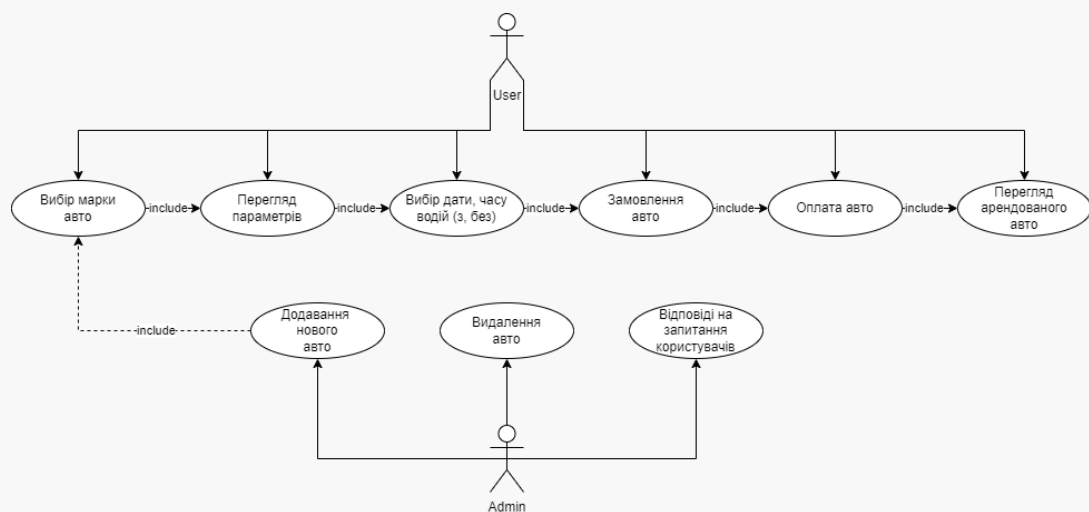


Рисунок 3.10 – Діаграма випадків використання

Вибір марки авто.

Основний потік:

- користувач відкриває сторінку вибору авто;
- система показує список доступних марок авто;
- користувач обирає марку авто.

Альтернативні потоки:

- користувач використовує пошук для швидкого вибору марки.

Помилкові потоки:

- немає доступних авто – система виводить відповідне повідомлення.

Перегляд параметрів.

Основний потік:

- користувач відкриває сторінку параметрів обраного авто;
- система відображає технічні характеристики;
- користувач переглядає параметри авто.

Альтернативні потоки:

- користувач використовує пошук для швидкого вибору марки.

Помилкові потоки:

- немає доступних авто – система виводить відповідне повідомлення.

Вибір дати, часу, водія (з водієм чи без).

Основний потік:

- користувач відкриває сторінку вибору дати та часу оренди;
- користувач обирає, чи потрібен йому водій;
- користувач обирає дату, час та наявність водія.

Альтернативні потоки:

- користувач обирає багаторазову оренду;
- користувач змінює дату або час після вибору.

Помилкові потоки:

- обране авто недоступне на обрану дату – система пропонує альтернативи.

Замовлення авто.

Основний потік:

- користувач підтверджує обране авто, дату і час;
- система показує підсумкову інформацію про замовлення;
- користувач натискає «Бронювати»;
- система формує замовлення.

Альтернативні потоки:

- користувач змінює параметри замовлення перед підтвердженням.

Помилкові потоки:

- замовлення не вдалося створити – система виводить повідомлення про помилку.

Оплата авто.

Основний потік:

- користувач обирає спосіб оплати (карта, Apple Pay);
- система перенаправляє на сторінку платежу;
- користувач заповнює дані для оплати;
- система підтверджує оплату.

Альтернативні потоки:

- користувач використовує Apple Pay для швидкої оплати.

Помилкові потоки:

- оплата не пройшла – система пропонує повторити або змінити спосіб оплати.

Перегляд орендованого авто.

Основний потік:

- користувач відкриває розділ «Мої бронювання»;
- система відображає інформацію про заброньоване авто;
- користувач переглядає деталі оренди.

Альтернативні потоки:

- користувач змінює дату або час оренди.

Помилкові потоки:

- інформація про бронювання недоступна – система повідомляє про

помилку.

Додавання нового авто (Адміністратор).

Основний потік:

- адміністратор відкриває сторінку додавання нового авто;
- адміністратор заповнює всю необхідну інформацію про авто;
- адміністратор натискає кнопку «Додати авто»;
- авто додано до списку автомобілів.

Альтернативні потоки:

- адміністратор додає фото авто.

Помилкові потоки:

- не вдалося додати авто – система виводить повідомлення про

помилку.

Видалення авто (Адміністратор).

Основний потік:

- адміністратор відкриває список авто;
- обирає авто для видалення;
- система підтверджує видалення;
- авто видаляється із системи.

Альтернативні потоки:

- адміністратор видаляє помилково-додане авто до списку.

Помилкові потоки:

- не вдалося видалити авто – система повідомляє про помилку.

Відповіді на запитання користувачів (Адміністратор).

Основний потік:

- адміністратор відкриває розділ «Запити користувачів»;
- обирає запит;
- відповідає на питання;
- система відправляє відповідь користувачу.

Альтернативні потоки:

- адміністратор додає посилання на довідкову інформацію у відповідь.

Помилкові потоки:

– запит неможливо опрацювати – система пропонує користувачу звернутися в службу підтримки.

3.7 Висновки за розділом 3

У третьому розділі було проведено детальний аналіз та проєктування програмного забезпечення, призначеного для системи оренди автомобілів. Було визначено функціональні можливості системи, зокрема вибір та перегляд авто, бронювання та оплату, що дозволило сформулювати вимоги до її реалізації.

Розглянуто загальну архітектуру програми, що складається з клієнтської та серверної частин, а також розроблено принципи її роботи, включаючи взаємодію між основними модулями. Було здійснено вибір архітектурного шаблону, що забезпечує розширюваність та зручність подальшої підтримки програмного забезпечення. У межах цього процесу створено діаграму класів, яка демонструє основні зв'язки між об'єктами системи, виконано візуалізацію компонентів та побудовано діаграму взаємодії елементів, що відображає логіку роботи програми.

Окрему увагу приділено розробці концептуальної моделі бази даних, яка визначає структуру зберігання інформації та взаємозв'язки між її основними сутностями. Це дозволяє забезпечити ефективне збереження та обробку даних, необхідних для функціонування системи.

Також було проаналізовано взаємодію модулів системи та створено діаграму випадків використання, яка ілюструє основні сценарії взаємодії користувача та адміністратора із системою. У результаті проведеної роботи було сформовано чітку концепцію побудови програмного забезпечення, що сприятиме ефективній реалізації та впровадженню системи в експлуатацію.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення та умови використання додатку

Додаток «Vroomie» призначений для організації процесу оренди автомобілів, надаючи користувачам зручний інтерфейс для вибору, бронювання та оплати оренди транспортних засобів. Він забезпечує інтеграцію з базою даних, платіжними сервісами та системою локації, що дозволяє користувачам швидко знаходити доступні автомобілі, переглядати їх характеристики, здійснювати оплату та контролювати історію оренд.

Програма підтримує два типи користувачів:

- клієнти, які реєструються, обирають автомобіль, оформлюють бронювання та здійснюють оплату.
- адміністратори, які керують автопарком, обробляють звернення клієнтів і контролюють наявність автомобілів у системі.

Апаратні вимоги.

Для користувача (клієнта):

- операційна система: Android 8.0 або IOS 15.1;
- оперативна пам'ять: 2 ГБ і більше;
- вільного місця: 1 ГБ вільного місця;
- додаткові вимоги: Підключення до Інтернету.

Для сервера:

- операційна система: Windows Server 2019+, Ubuntu 20.04;
- процесор: Intel Xeon або еквівалентний AMD;
- оперативна пам'ять: 8 ГБ і більше;
- місце на диску: 10 ГБ вільного місця (залежно від обсягу даних);
- база даних: MongoDB 5.0+.

Додаткові вимоги:

- стабільне з'єднання з Інтернетом, наявність захищеного SSL-сертифіката для безпечного обміну даними.

Програма «Vroomie» використовує архітектуру клієнт-сервер, де

користувач отримує доступ до функціоналу через клієнтський додаток, а сервер обробляє запити, взаємодіє з базою даних і здійснює управління бізнес-логікою.

4.2 Основні характеристики програми

Додаток «Vroomie» забезпечує користувачам повноцінний процес оренди автомобілів через зручний інтерфейс, інтуїтивну навігацію та сучасні технології. Функціональні можливості забезпечують виконання ключових задач, а нефункціональні характеристики гарантують стабільну, безпечну та ефективну роботу системи.

Функціональні можливості.

Додаток надає такі основні функції для користувачів:

- реєстрація та авторизація користувачів (вхід через Email та пароль);
- перегляд доступних автомобілів (фільтрація за брендом);
- детальна інформація про автомобіль (опис, технічні характеристики, фото);
- оренда автомобіля (вибір дати, часу, місця отримання та повернення, з водієм чи без);
- оплата оренди (Apple Pay або введення даних картки);
- відстеження поточної оренди (інформація про активне бронювання);
- карта локацій автомобілів;
- додавання у обране (збереження улюблених автомобілів);
- перегляд історії оренд (інформація про минулі бронювання);
- користувацький профіль (редагування даних, зміна фото, налаштування);
- форма звернення до адміністрації (запити, скарги, питання);
- оцінка програми (система відгуків і рейтингів).

Нефункціональні характеристики.

Окрім функціоналу, додаток «Vroomie» забезпечує такі властивості:

- зручність використання – інтуїтивний інтерфейс із сучасним дизайном;
- кросплатформність – підтримка IOS, Android;
- продуктивність – швидке завантаження контенту, ефективна обробка запитів;
- надійність – стабільна робота програми, мінімізація збоїв;
- безпека – захист персональних даних;
- масштабованість – можливість розширення системи та підтримка нових функцій;
- автоматичне оновлення – підтримка актуальних версій без додаткових дій користувача.

4.3 Інструкція із запуску та використання додатку

Додаток «Vroomie» розроблений для зручного використання та забезпечує користувачам можливість швидко орендувати автомобіль. У цьому розділі описано покроковий процес встановлення, запуску та використання додатку.

Для початку роботи з «Vroomie» необхідно виконати такі кроки.

Крок 1. Завантаження додатку:

- завантаження додатку з Play Market або App Store;
- перевірка, що пристрій відповідає мінімальним системним вимогам.

Крок 2. Запуск додатку:

- відкриття «Vroomie» через меню програм.

4.4 Вхідні та вихідні дані

Функціонування додатку «Vroomie» базується на обробці вхідних та вихідних даних, що забезпечують взаємодію користувача із системою та дозволяють здійснювати процес оренди автомобілів.

Вхідні дані – це інформація, що вводиться в систему користувачами або адміністраторами, а також дані, отримані з бази даних.

Основні вхідні дані.

Дані про автомобілі:

- марка, модель;
- рік випуску;
- технічні характеристики (тип двигуна, пробіг, коробка передач тощо);
- вартість оренди;
- фото автомобіля;
- місце розташування авто.

Дані користувачів:

- ім'я, прізвище;
- контактні дані (номер телефону, Email);
- пароль для входу;
- платіжна інформація;
- обране авто, історія оренд.

Дані адміністратора:

- додавання нових авто;
- видалення автомобілів;
- відповіді на запитання користувачів.

Вихідні дані – це інформація, яку користувач отримує в результаті взаємодії із системою.

Основні вихідні дані.

Результати пошуку авто:

- список доступних для оренди автомобілів;
- фільтрований список за обраною маркою.

Орендоване авто користувачем:

- підтверджена інформація про автомобіль, що орендується;
- дані про період оренди;
- локація отримання авто;
- підтвердження оплати.

Інформація про автомобіль:

- повний опис обраного авто;
- відгуки інших користувачів;
- фото та характеристика машини.

Нові автомобілі, внесені адміністратором:

- список автомобілів.

4.5 Системні повідомлення

У процесі роботи з додатком «Vroomie» користувач може стикатися з різними системними повідомленнями, які інформують про успішне виконання дій або можливі помилки. Системні повідомлення поділяються на помилки та повідомлення про успіх, що дозволяють користувачу орієнтуватися в роботі додатку та коригувати свої дії.

Повідомлення про помилки.

Помилка при авторизації у додаток:

- текст повідомлення: «Користувача не знайдено»;
- дії: додаток не пропускає користувача далі та дає змогу повторити спробу авторизації, перевірити введені дані або відновити пароль.

Помилка при пошуку авто:

- текст повідомлення: «Автомобіль не знайдено»;
- дії: додаток не відображає жодних результатів, залишаючи поле пошуку активним для введення нових параметрів.

Помилка при оформленні оренди авто:

- текст повідомлення: «Не всі поля заповнені»;
- дії: додаток не дозволяє підтвердити бронювання, поки всі необхідні параметри не будуть обрані.

Помилка при додаванні платіжної картки:

- текст повідомлення: «Не всі поля заповнені, повторіть спробу»;
- дії: додаток не додає карту в платіжну систему та залишає форму відкритою для редагування.

Повідомлення про успіх.

Успішне оформлення оренди авто:

- текст повідомлення: «Успішна оренда. Переглянути орендоване авто можна у власному профілі.»;
- дії: додаток відображає підтвердження з інформацією про оренду та перенаправляє користувача до списку його бронювань.

Успішне оновлення даних користувача:

- текст повідомлення: «Дані успішно змінено»;
- дії: додаток зберігає оновлену інформацію (ім'я, пароль тощо) та підтверджує зміни на екрані профілю.

Такі повідомлення допомагають користувачу орієнтуватися в роботі з додатком та мінімізують можливі помилки при введенні даних або використанні функцій сервісу.

4.6 Висновки за розділом 4

У цьому розділі було розглянуто ключові аспекти роботи мобільного додатку «Vroomie», зокрема його призначення, характеристики, порядок використання, структуру вхідних і вихідних даних, а також систему повідомлень.

Було визначено, що додаток призначений для оренди автомобілів, а його використання можливе на пристроях, які відповідають певним апаратним

вимогам. Окреслено основні функціональні та нефункціональні характеристики, що забезпечують комфортне користування сервісом, а також описано процеси запуску та роботи з додатком.

Окремо розглянуто вхідні та вихідні дані, серед яких інформація про автомобілі, реєстраційні дані користувачів та деталі оренди. Також було визначено систему повідомлень, що допомагає користувачеві орієнтуватися у функціоналі додатку, повідомляючи про помилки або успішне виконання операцій.

Таким чином, цей розділ описує технічні та функціональні особливості «Vroomie», необхідні для його ефективного використання та взаємодії з користувачами.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення додатку

Мобільний додаток «Vroomie» призначений для зручної та швидкої оренди автомобілів. Він дозволяє користувачам переглядати доступні авто, обирати відповідний транспортний засіб, налаштовувати параметри оренди (дата, час, тип оренди – з водієм або без), здійснювати оплату та отримувати підтвердження бронювання.

Додаток забезпечує зручну взаємодію як для звичайних користувачів, які шукають транспортний засіб, так і для адміністраторів, що керують базою даних авто. Інтуїтивний інтерфейс, можливість збереження улюблених автомобілів та підтримка різних платіжних методів роблять «Vroomie» ефективним рішенням для оренди транспорту.

5.2 Умови роботи додатку

5.2.1 Апаратні вимоги

Для коректної роботи додатка «Vroomie» необхідно:

- операційна система: Android 8.0 або IOS 15.1;
- оперативна пам'ять: 2 ГБ і більше;
- вільного місця: 1 ГБ вільного місця;
- додаткові вимоги: Підключення до Інтернету.

5.2.2 Вимоги до користувача

Для успішного використання додатку «Vroomie» користувач повинен мати базові навички роботи зі смартфоном, зокрема вміння встановлювати мобільні додатки, працювати з інтерфейсом та здійснювати онлайн-оплату. Обов'язковою умовою є наявність особистого акаунту в системі, який створюється під час реєстрації. Крім того, для коректної роботи додатку

необхідне стабільне підключення до Інтернету, оскільки всі операції, включаючи перегляд доступних автомобілів, бронювання, оплату та отримання підтверджень, виконуються в режимі онлайн.

5.3 Робота з додатком

Після запуску, користувача зустрічає вітальна сторінка додатку, яка зображена на рисунку 5.1, та надає можливість перейти на сторінку логіну.

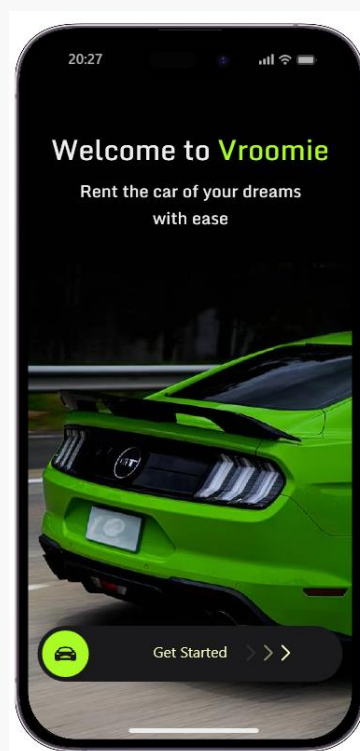


Рисунок 5.1 – Вітальна сторінка «Vroomie»

Однією з перших сторінок, з якою взаємодіє користувач у додатку «Vroomie», є сторінка авторизації (Login) (рисунок 5.2). Вона призначена для входу в особистий акаунт та надає можливість продовжити роботу з додатком.

На сторінці Login відображаються такі елементи:

- текстове поле з привітанням: «Welcome back! Continue on your way», яке створює комфортну атмосферу для користувача;
- поля введення логіну та паролю, які необхідні для авторизації

користувача у системі;

- кнопка «Sign In», яка стає активною після введення коректних даних. натискання на цю кнопку ініціює перевірку введеної інформації та вхід у систему;

- кнопка «Sign Up», яка дозволяє новим користувачам створити акаунт, якщо вони ще не зареєстровані у системі.

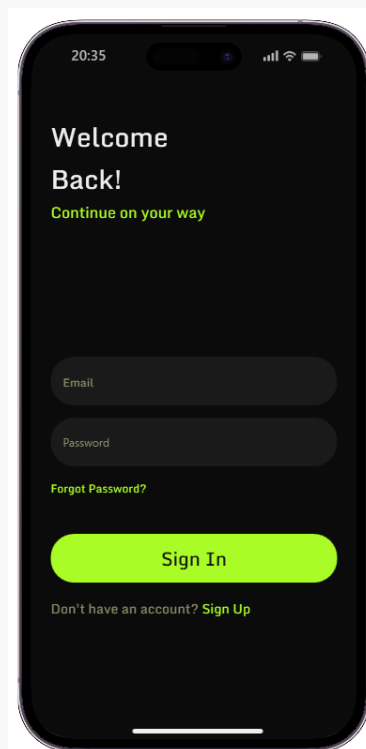


Рисунок 5.2 – Сторінка Login «Vroomie»

Сторінка Registration (рисунок 5.3) призначена для створення нового облікового запису в додатку «Vroomie». Вона дозволяє новим користувачам зареєструватися, заповнивши необхідні дані.

На сторінці Registration розташовані наступні елементи:

- а) текстове поле з привітанням: «Create Account. Start on your way», яке мотивує користувача завершити реєстрацію та розпочати користування додатком;

- б) чотири поля введення:

- 1) ПІБ (повне ім'я) – для введення ідентифікаційних даних

користувача;

2) Email – для зв’язку та подальшої аутентифікації;

3) номер телефону – для можливого зв’язку та безпеки акаунту;
пароль – для захисту облікового запису.

в) чекбокс підтвердження умов користування, який користувач має обов’язково відзначити перед завершенням реєстрації;

г) кнопка «Sign Up», яка стає активною після введення коректних даних. Натискання на неї створює акаунт, і у разі успішної реєстрації користувача автоматично перенаправляє на головну сторінку додатку;

д) кнопка «Sign In», яка дозволяє користувачу повернутися на сторінку авторизації, якщо в нього вже є акаунт.

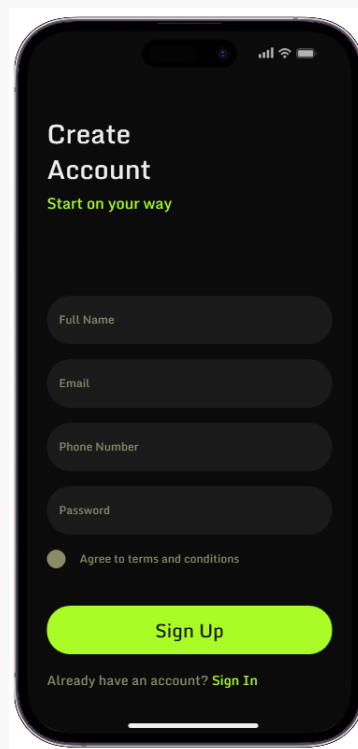


Рисунок 5.3 – Сторінка Registration «Vroomie»

Головна сторінка Home (рисунок 5.4) є основним екраном додатку «Vroomie», з якого користувач отримує доступ до основних функцій сервісу. Інтерфейс сторінки містить зручні елементи навігації та корисну інформацію для швидкого вибору авто.

На сторінці Home розташовані наступні елементи:

а) кнопка швидкого переходу до особистого профілю – розташована у верхньому куті у вигляді кружка з іконкою профілю, що дозволяє миттєво перейти до розділу Profile;

б) локація користувача – статичний напис, що відображає поточне місцезнаходження користувача;

в) кнопка пошуку – дозволяє знайти необхідне авто за назвою чи параметрами;

г) напис «Top Brands» – заголовок, що вказує на розділ популярних брендів автомобілів;

д) горизонтальний список доступних брендів авто – інтерактивний перелік автомобільних марок, наявних у додатку. користувач може перегортати список та обирати бажаний бренд;

е) популярне авто тижня – окремий блок, що відображає найбільш популярне авто за останній тиждень. у цьому блоці є дві кнопки:

1) «Додавання цього авто до улюбленого» – дозволяє зберегти авто у розділі Favourites.

2) «Перейти до характеристик авто» – відкриває сторінку з детальною інформацією про автомобіль;

ж) меню навігації – розташоване в нижній частині екрана та містить чотири основні кнопки для переходу між розділами:

1) Home – повернення на головний екран;

2) Map – перегляд розташування доступних авто на карті;

3) Favourites – збережені автомобілі користувача;

4) Profile – особисті дані та налаштування акаунту.

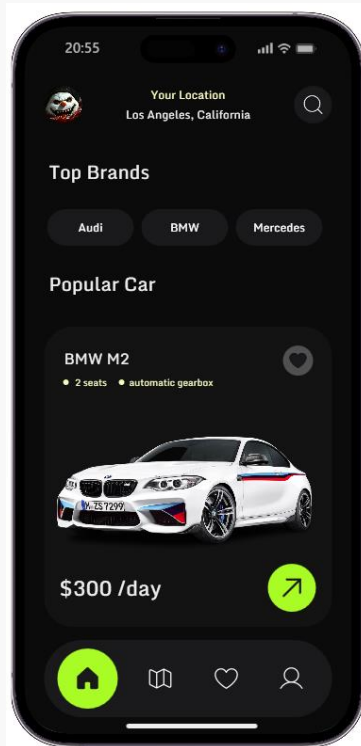


Рисунок 5.4 – Сторінка Home «Vroomie»

Сторінка Search Car (рисунок 5.5) дозволяє користувачу знайти потрібний автомобіль за його брендом або назвою. Інтерфейс сторінки розроблений так, щоб зробити процес пошуку максимально швидким та зручним.

На сторінці Search Car розташовані наступні елементи:

- кнопка «Back» – розташована у верхньому лівому куті. При натисканні повертає користувача назад на головну сторінку Home;
- поле для введення бренду автомобіля – тут користувач вводить назву бренду або моделі авто, яке хоче знайти. Ліворуч від поля розташована кнопка пошуку, натискання якої ініціює пошук за введеним запитом;
- блок «Recommendation» – містить автомобіль, який був відзначений як найпопулярніший за тиждень (цей самий автомобіль відображається на головній сторінці);
- список усіх доступних авто – спочатку відображає всі автомобілі, доступні для оренди. Якщо користувач скористається пошуком, список оновиться відповідно до запиту та покаже лише знайдені авто певного бренду

або конкретну модель;

– меню навігації – розташоване внизу сторінки та забезпечує швидкий доступ до основних розділів додатку (Home, Map, Favourites, Profile).

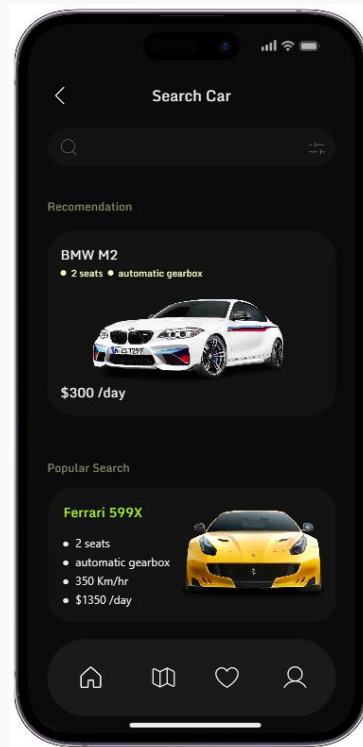


Рисунок 5.5 – Сторінка Search Car «Vroomie»

Сторінка All Cars (рисунок 5.6) відображає список усіх автомобілів обраного бренду. Вона відкривається після того, як користувач на головній сторінці Home натискає на один із брендів у списку «Top Brands».

На сторінці All Cars розташовані наступні елементи:

– кнопка «Back» – повертає користувача назад на головну сторінку Home;

– список автомобілів обраного бренду – відображає всі доступні для оренди моделі лише тієї марки, яку обрав користувач. Наприклад, якщо користувач натиснув на бренд Chevrolet, у списку будуть представлені лише автомобілі цього виробника;

– меню навігації – містить чотири кнопки для переходу між основними розділами додатку: Home, Map, Favourites, Profile.

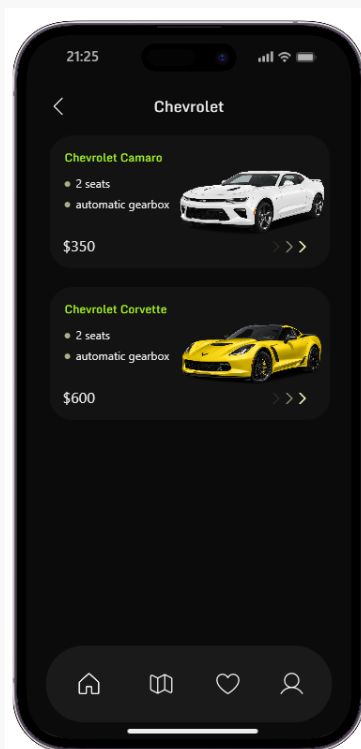


Рисунок 5.6 – Сторінка All Cars «Vroomie»

Сторінка Car Details (рисунок 5.7) надає користувачеві детальну інформацію про обраний автомобіль. У верхній частині розташована кнопка «Back», яка дозволяє повернутися до попереднього екрану вибору авто. Поруч із нею знаходиться кнопка «Favourites», яка додає автомобіль у список обраних користувачем транспортних засобів.

На сторінці розташовано:

- а) фото автомобіля спереду;
- б) назва авто – розташована зліва під фото;
- в) оцінка авто та кількість відгуків – розміщені справа під фото;
- г) текстовий напис «Specification» – позначає блок із характеристиками авто;
- д) блок характеристик авто, що включає:
 - 1) тип палива;
 - 2) об'єм двигуна;
 - 3) тип приводу (передній, задній, повний);
 - 4) максимальна швидкість авто;

- е) кнопка «Select Location» – при свайпі переводить користувача до наступного кроку бронювання авто;
- ж) меню навігації (Home, Map, Favourites, Profile).

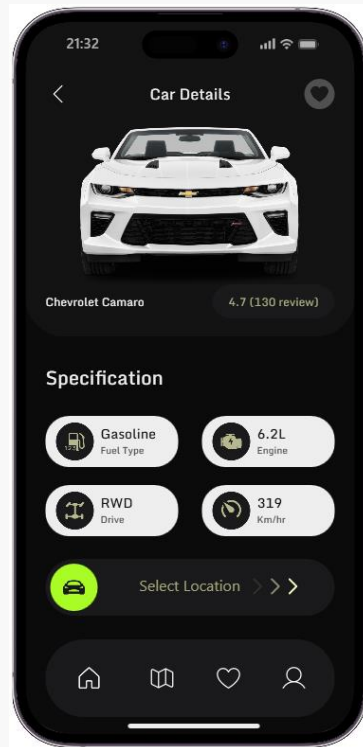


Рисунок 5.7 – Сторінка Car Details «Vroomie»

Сторінка Select Location (рисунок 5.8), на якій користувач обирає деталі оренди, зокрема місце отримання та повернення авто, дати, час і формат оренди. Цей крок є важливим, оскільки на його основі розраховується вартість прокату та доступність транспортного засобу.

На сторінці розташовано:

- кнопка «Back» – повертає користувача на крок назад;
- текстові написи – вказують, звідки користувач забирає та куди повертає авто (Vermont Square);
- вибір дати та часу початку оренди – користувач обирає дату й час отримання авто;
- вибір дати та часу завершення оренди – користувач зазначає, коли планує повернути авто;

- тип оренди – користувач обирає варіант оренди: з водієм або без нього;
- кнопка «Review Summary» – переводить користувача до наступного етапу бронювання;
- меню навігації (Home, Map, Favourites, Profile).

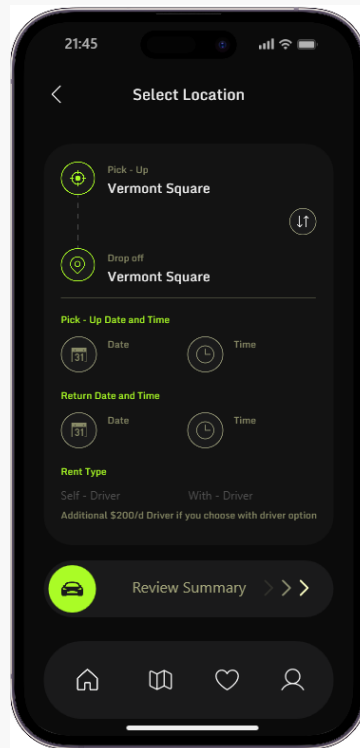


Рисунок 5.8 – Сторінка Select Location «Vroomie»

Сторінка Review Summary (рисунок 5.9), на якій користувач переглядає дані про своє бронювання та підсумкову інформацію, розташовані такі елементи:

- а) кнопка «Back» – повертає користувача на крок назад до попередніх етапів;
- б) фото авто спереду – зображення обраного транспортного засобу;
- в) назва авто – текст, який містить назву обраного авто;
- г) блок з інформацією, яку користувач обрав на попередній сторінці:
 - 1) початок оренди;
 - 2) завершення оренди;

- 3) місцезнаходження авто;
- 4) тип оренди (з водієм або без);
- 5) ціна оренди за день;

д) кнопка «Book Now», яка містить кінцеву суму оренди обраного авто, та дозволяє користувачу підтвердити бронювання;

е) меню навігації (Home, Map, Favourites, Profile).

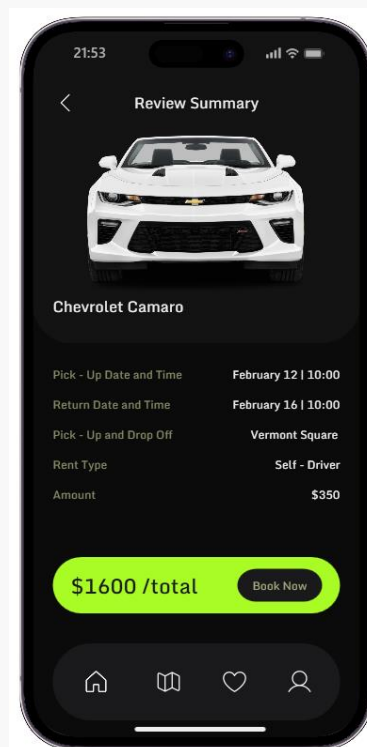


Рисунок 5.9 – Сторінка Review Summary «Vroomie»

Сторінка Payment Methods (рисунок 5.10), на якій користувач обирає метод оплати для завершення бронювання, розташовані такі елементи:

- а) кнопка «Back» – повертає користувача на попередній крок;
- б) вибір методів оплати – доступні два варіанти:
 - 1) Apple Pay;
 - 2) додавання власної картки – користувач обирає цей варіант для введення даних своєї картки;
- в) кнопка кінцевої суми оренди – відображається загальна сума, яку користувач має сплатити за оренду;

г) меню навігації (home, map, favourites, profile).

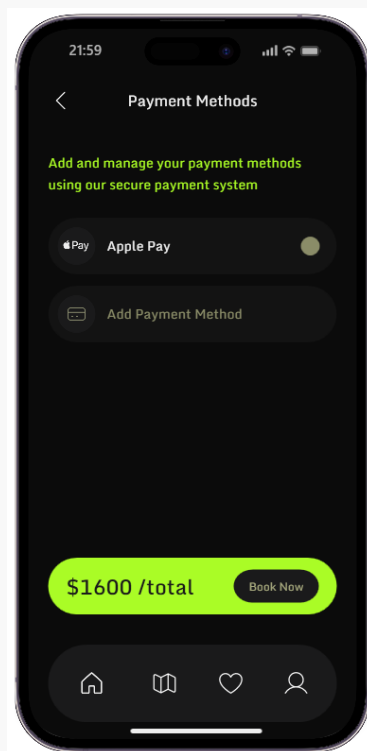


Рисунок 5.10 – Сторінка Payment Methods «Vroomie»

Сторінка Add Payment Methods (рисунок 5.11), на якій користувач вводить дані для оплати оренди, розташовані такі елементи:

а) кнопка «Back» – повертає користувача на попередній крок (сторінку Payment Methods);

б) поля для вводу даних користувача – включають:

- 1) номер карти – поле для введення номера платіжної картки;
- 2) термін закінчення карти – поле для введення дати завершення терміну дії картки;
- 3) CVV – поле для введення тризначного коду CVV картки;
- 4) ПІБ власника карти – поле для введення імені та прізвища власника картки;

в) кнопка «Payment» – після натискання цієї кнопки, при правильному введенні даних, користувач успішно здійснить оплату оренди і побачить нове вікно з підтвердженням оплати.

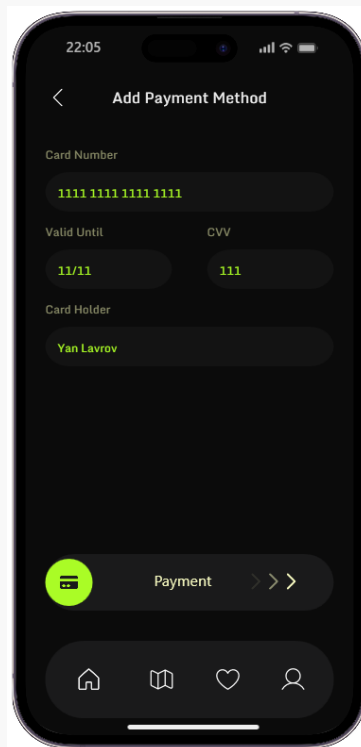


Рисунок 5.11 – Сторінка Add Payment Methods «Vroomie»

Після успішної оренди, користувач бачить вікно успішної оплати (рисунок 5.12), на якому розташовані такі елементи:

а) галочка – символізує успішне завершення операції;

б) текстові написи:

1) «Rental Success!» – повідомлення про успішну оренду;

2) «The receipt has been sent to your phone number» – підтвердження, що чек з орендою надіслано на номер телефону користувача;

в) кнопка «Back To Main» – при натисканні на цю кнопку, користувача перенаправляє на головну сторінку додатку.

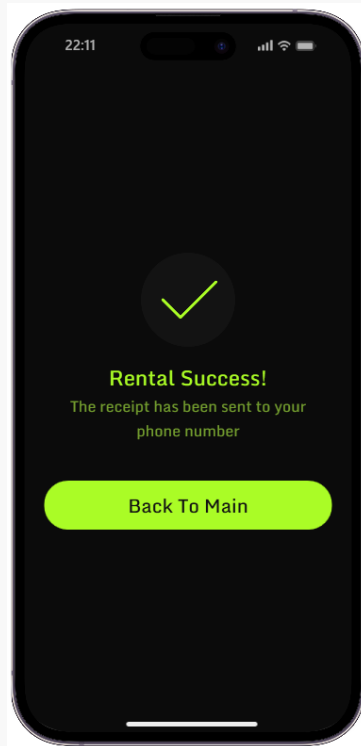


Рисунок 5.12 – Сторінка Rental Success «Vroomie»

На сторінці Map (рисунок 5.13) користувач переглядає власну геолокацію та бачить автомобілі, які знаходяться поруч і доступні для оренди. Ця функція допомагає швидко знайти найближче авто та зробити вибір відповідно до локації.

На сторінці розташовано:

- геолокація користувача – відображає його поточне місцезнаходження;
- мітки авто – відображають автомобілі, які знаходяться поруч та доступні для оренди;
- меню навігації (Home, Map, Favourites, Profile).

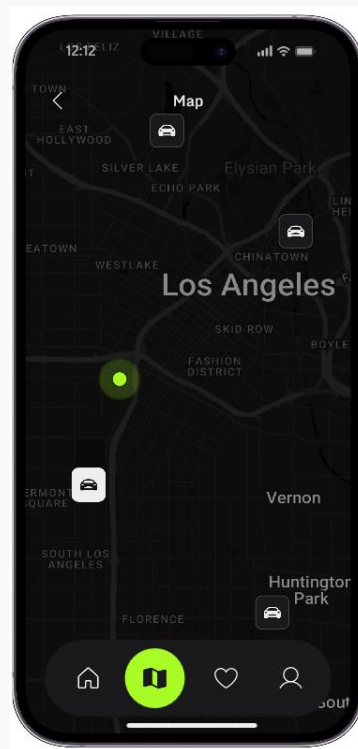


Рисунок 5.13 – Сторінка Map «Vroomie»

На сторінці Favourites (рисунок 5.14) користувач переглядає список автомобілів, які було додано до обраного. Це дозволяє швидко знайти вподобані авто без необхідності повторного пошуку. Зі списку користувач може одразу перейти до детального опису авто та розпочати процес бронювання.

На сторінці розташовано:

- список улюблених авто – відображає всі автомобілі, які користувач додав до обраного;
- можливість відкриття опису авто – при натисканні на будь-яке авто зі списку відкривається сторінка Car Details з детальною інформацією про транспортний засіб;
- меню навігації (Home, Map, Favourites, Profile).

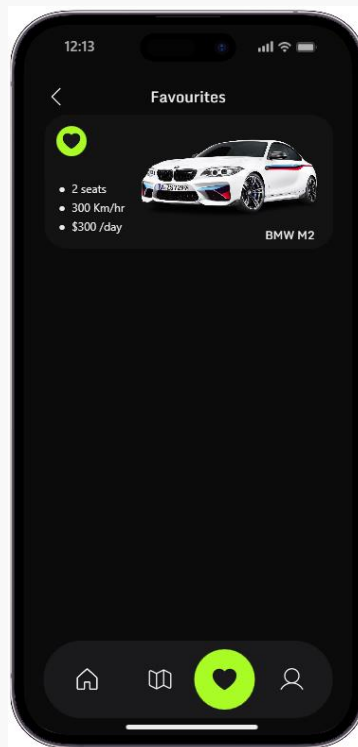


Рисунок 5.14 – Сторінка Favourites «Vroomie»

На сторінці Profile (рисунок 5.15) користувач може переглянути та змінити свої персональні дані, а також отримати доступ до основних функцій, пов'язаних з акаунтом та додатком. Тут він може оновити фото профілю, перевірити список орендованих авто, переглянути платіжні методи та отримати підтримку в разі потреби.

На сторінці розташовано:

- ім'я користувача та номер телефону – відображаються у верхній частині сторінки;
- фото профілю – користувач може змінити його, натиснувши на іконку зміни.

Два основні блоки.

Account Info:

- personal information (рисунок 5.16) – кнопка для переходу на сторінку зміни особистих даних;
- payment methods (рисунок 5.17) – кнопка для перегляду збережених карт, що використовуються для оплати;

– rented cars (рисунок 5.18) – кнопка для перегляду списку успішно орендованих авто.

App Info:

– settings (рисунок 5.19) – кнопка для переходу на сторінку налаштувань додатку;

– help (рисунок 5.20) – кнопка для переходу на сторінку підтримки (онлайн-чат з адміністратором для вирішення питань).

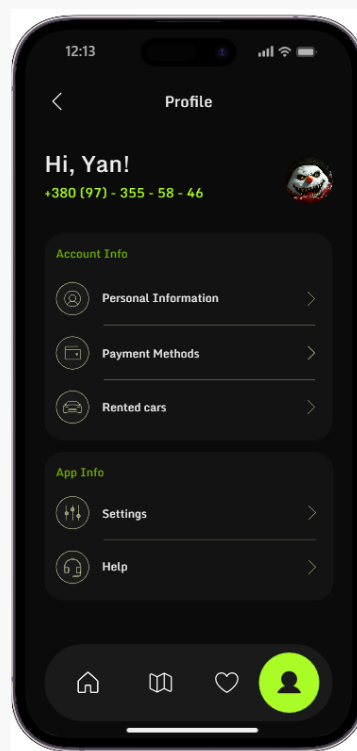


Рисунок 5.15 – Сторінка Profile «Vroomie»

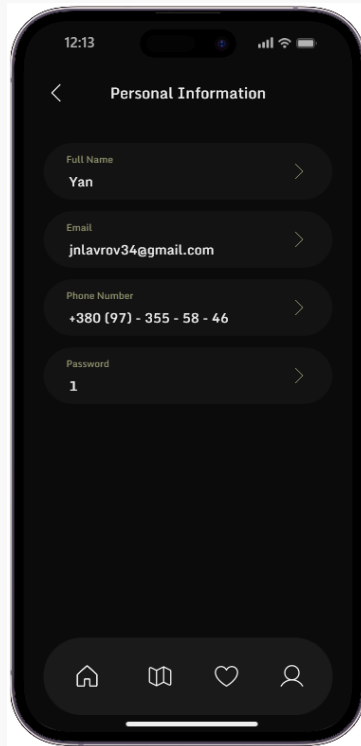


Рисунок 5.16 – Сторінка Personal Information «Vroomie»

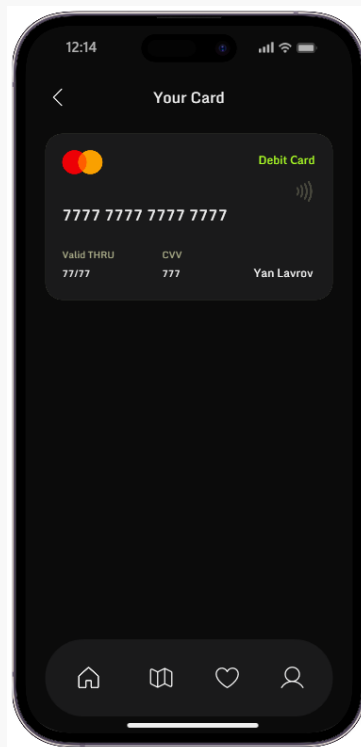


Рисунок 5.17 – Сторінка Payment Methods «Vroomie»

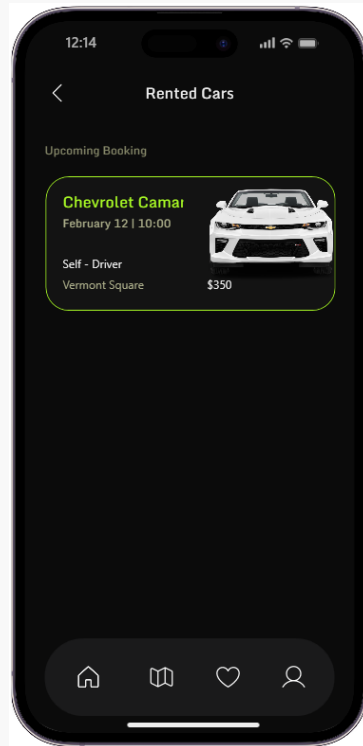


Рисунок 5.18 – Сторінка Rented Cars «Vroomie»

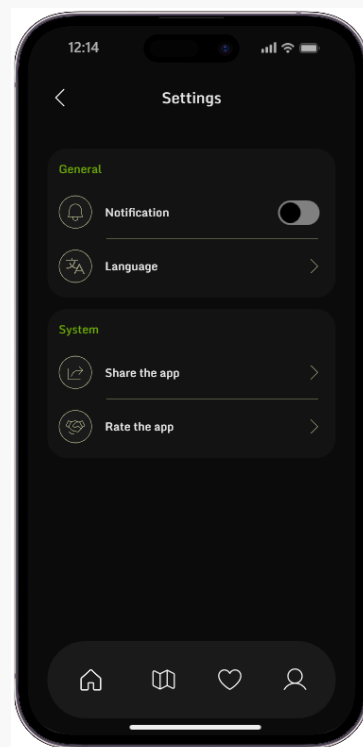


Рисунок 5.19 – Сторінка Settings «Vroomie»

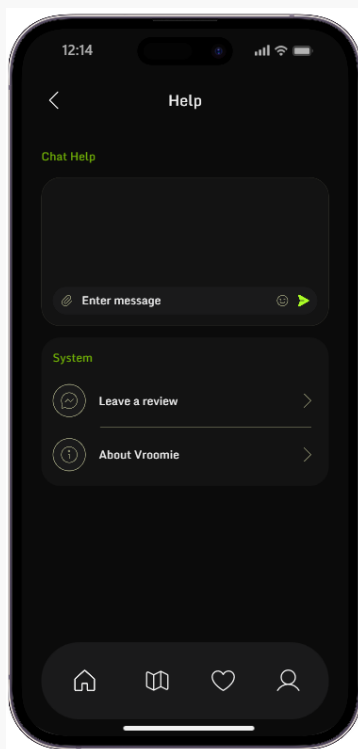


Рисунок 5.20 – Сторінка Help «Vroomie»

На сторінці Rate the App (рисунок 5.21) користувач має змогу оцінити додаток та залишити відгук про його роботу. Це дозволяє збирати зворотний зв'язок, який допоможе покращити функціональність та якість сервісу. Потрапити на цю сторінку можна через Settings або Help.

На сторінці розташовано:

- 5-зіркову систему оцінки – користувач може натискати на зірочки та обрати оцінку додатку;
- поле для введення відгуку – користувач може залишити коментар або запитання щодо роботи додатку;
- кнопку відправлення – після натискання відгук або запитання надсилається адміністратору, який переглядає їх у своїй панелі.

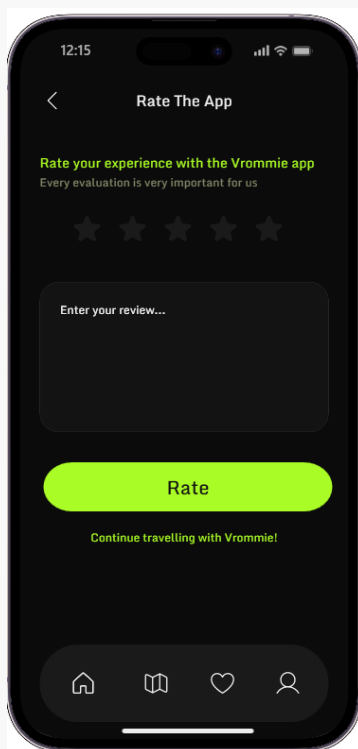


Рисунок 5.21 – Сторінка Rate the App «Vroomie»

На сторінці About Vroomie (рисунок 5.22) користувач може ознайомитися з основною інформацією про додаток, автомобілі, що доступні для оренди, а також про розробника. Потрапити на цю сторінку можна, натиснувши кнопку About Vroomie у розділі Help.

На сторінці розташовані такі елементи:

- блок «Інформація про Vroomie» (рисунок 5.23) – містить загальний опис додатку та його можливості;
- блок «Інформація про авто» (рисунок 5.24) – включає детальну інформацію про автомобілі, доступні для оренди;
- блок «Інформація про розробника» (рисунок 5.25) – надає дані про автора проєкту;
- три кнопки соцмереж «Vroomie» – посилання на офіційні сторінки у соціальних мережах.

При натисканні на перші три блоки вони відкриваються на весь екран для детального ознайомлення. Кнопки соціальних мереж не відкривають додаткові вікна, а просто ведуть на відповідні платформи.

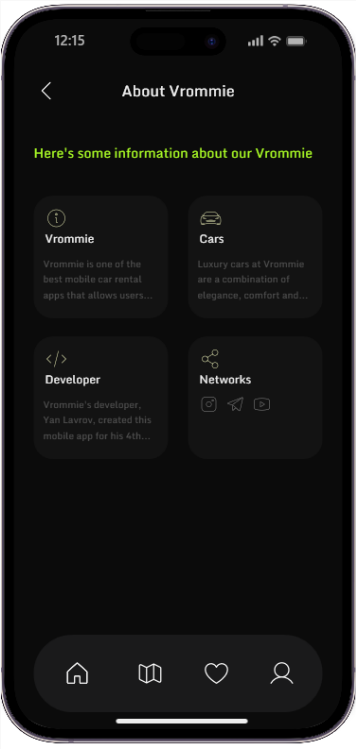


Рисунок 5.22 – Сторінка About Vroomie «Vroomie»

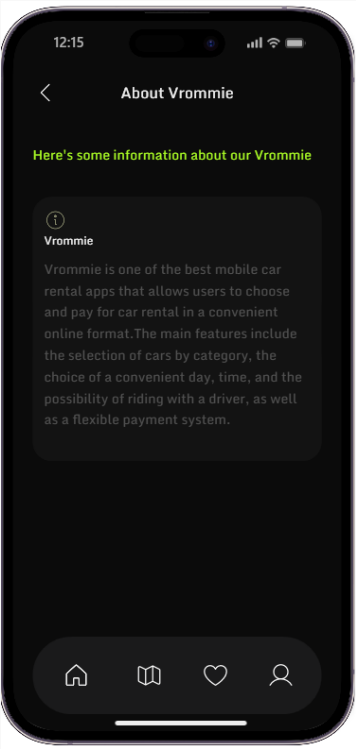


Рисунок 5.23 – Сторінка About Info Vroomie «Vroomie»

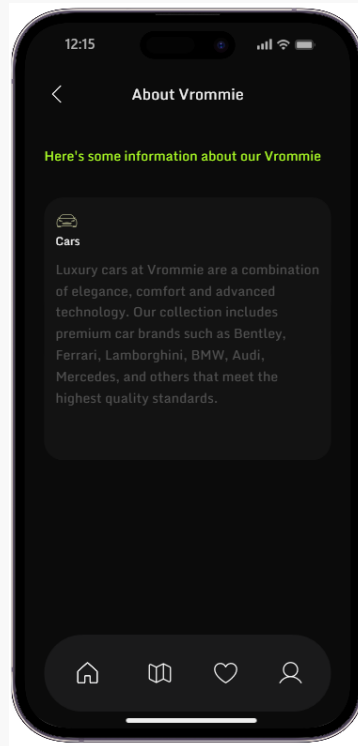


Рисунок 5.24 – Сторінка About Vroomie «Vroomie»

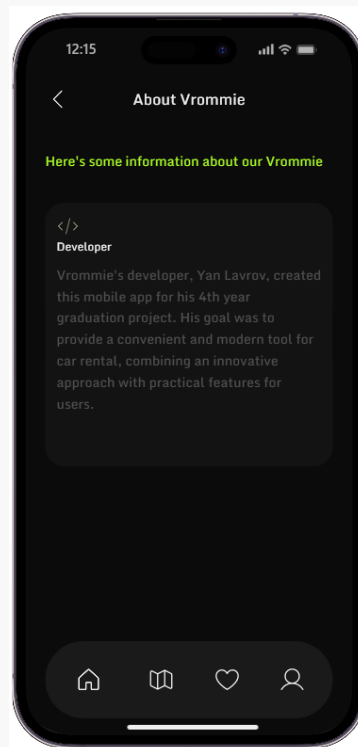


Рисунок 5.25 – Сторінка About Vroomie «Vroomie»

На сторінці Add Car (рисунок 5.26) адміністратор має можливість додати новий автомобіль до бази даних, заповнюючи відповідні поля інформацією про авто. Інтерфейс містить форму для введення всіх необхідних даних, таких як назва, фото, ціна оренди тощо. Також на сторінці доступне меню навігації, яке дозволяє перемикатися між основними функціями адміністратора: Add Car, Delete Car, Answer Question, Profile.

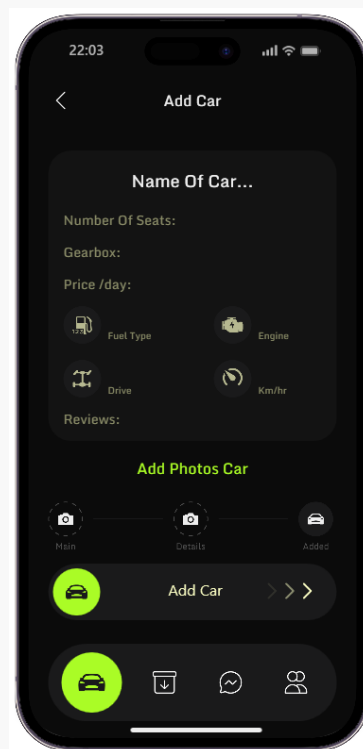


Рисунок 5.26 – Сторінка Add Car «Vroomie»

На сторінці Delete Car (рисунок 5.27) адміністратор може обрати бренд автомобіля з горизонтального списку, після чого нижче відображається список усіх авто, що належать до цього бренду. Для видалення автомобіля адміністратор натискає на нього, після чого з'являється вікно з успіхом операції.

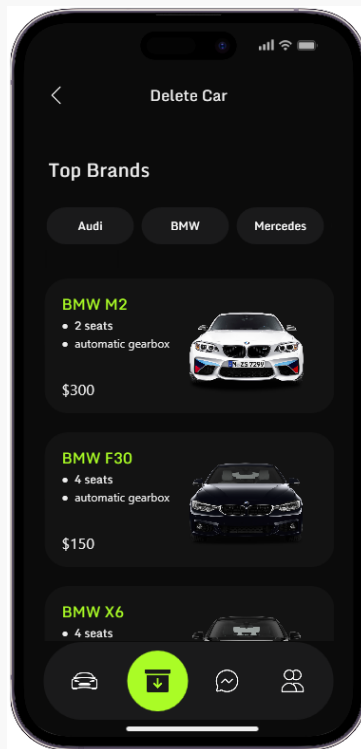


Рисунок 5.27 – Сторінка Delete Car «Vroomie»

На сторінці User Review (рисунок 5.28) відображається список запитань від користувачів. Адміністратор може обрати будь-яке запитання, натиснувши на нього, після чого відкриється нова сторінка для надання відповіді.

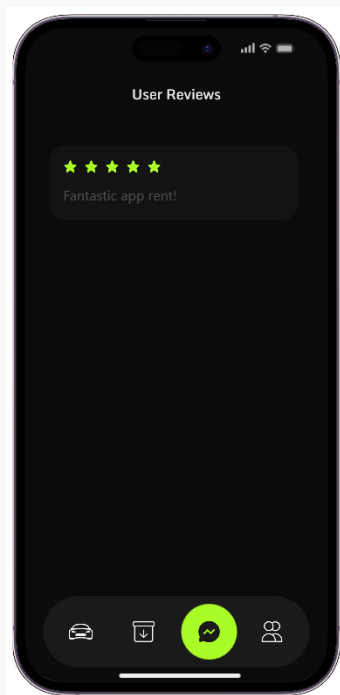


Рисунок 5.28 – Сторінка User Review «Vroomie»

На сторінці Reply to Review (рисунок 5.29) адміністратор бачить відгук або запитання від користувача та має можливість дати на нього відповідь. Тут відображаються оцінка користувача у вигляді зірок, залишене повідомлення, а також поле для введення відповіді. Після натискання кнопки відправлення відповідь надсилається, а адміністратор автоматично повертається на сторінку зі списком запитань, де оновлюється список і вже оброблене запитання зникає.

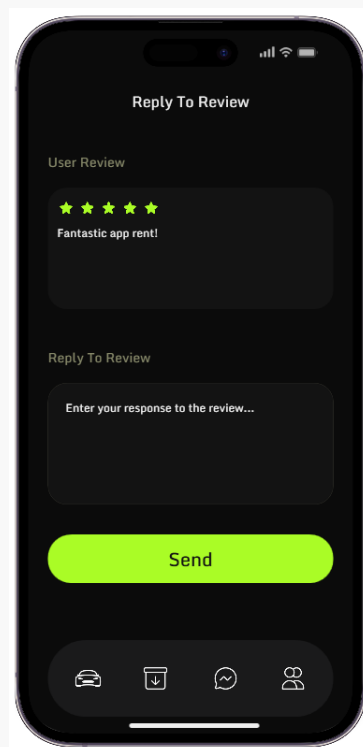


Рисунок 5.29 – Сторінка Reply to Review «Vroomie»

На сторінці Profile (рисунок 5.30) адміністратор бачить інформацію, яка вказує, що це його акаунт, а також короткий опис його обов'язків: додавання та видалення автомобілів, а також відповіді на запитання користувачів. Також на сторінці розташована кнопка виходу з акаунта, яка дозволяє адміністратору завершити сеанс і повернутися на екран входу.

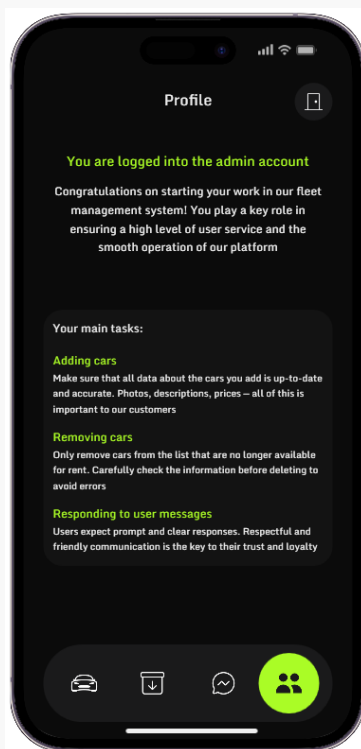


Рисунок 5.30 – Сторінка Profile «Vroomie»

5.4 Системні повідомлення для користувача

У додатку «Vroomie» використовуються різні типи системних повідомлень для інформування користувачів про важливі події, попередження щодо заповнення необхідних полів та повідомлення про помилки, які можуть виникати під час використання додатку. Ці повідомлення забезпечують коректну взаємодію з додатком, допомагають користувачеві орієнтуватися у процесі та усувають можливі проблеми, які можуть виникнути під час виконання операцій, таких як авторизація, бронювання авто чи оплата. Нижче наведено основні типи повідомлень, які можуть бути представлені користувачеві в процесі роботи з додатком.

Інформаційні повідомлення:

- успішне оформлення оренди – текст повідомлення: «Автомобіль успішно орендовано!»;
- успішна зміна даних профілю – текст повідомлення: «Дані успішно оновлено!».

Попереджувальні повідомлення:

- незаповнені поля при бронюванні авто – текст повідомлення: «Будь ласка, заповніть усі обов'язкові поля перед підтвердженням бронювання!»;
- незаповнені поля при додаванні картки – текст повідомлення: «Не всі поля заповнені».

Помилки:

- невірні дані при авторизації – текст повідомлення: «Користувача не знайдено або неправильний пароль»;
- помилка при пошуку автомобіля – текст повідомлення: «Автомобіль не знайдено»;
- помилка при оформленні оренди – текст повідомлення: «Не всі поля заповнені»;
- помилка при додаванні картки – текст повідомлення: «Не вдалося додати картку».

5.5 Висновки за розділом 5

У цьому розділі було детально розглянуто керівництво оператора для роботи з додатком «Vroomie». Було визначено його основне призначення, а також умови, необхідні для коректної роботи, зокрема апаратні вимоги та вимоги до користувача.

Окремо було описано процес взаємодії користувача з додатком, починаючи від перегляду доступних авто та їх бронювання до здійснення оплати та перегляду інформації у профілі. Також було розглянуто роль адміністратора, його можливості щодо керування автомобілями в базі даних та відповідей на запитання користувачів.

Крім того, було наведено системні повідомлення, які допомагають користувачам орієнтуватися в роботі додатка та отримувати зворотний зв'язок

щодо своїх дій.

Таким чином, цей розділ містить усю необхідну інформацію для користувачів і адміністраторів «Vroomie», що забезпечує комфортну та зрозумілу взаємодію з додатком.

6 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

6.1 Загальні положення

Цей розділ містить опис процесу експлуатації, тестування та експериментального дослідження додатку «Vroomie». У ньому розглядаються основні етапи перевірки працездатності додатку, оцінювання його функціональності, відповідності заявленим вимогам та зручності використання.

Експлуатація охоплює аналіз роботи додатку у реальних умовах, тестування спрямоване на виявлення та усунення можливих помилок, а експериментальне дослідження допомагає визначити ефективність впроваджених рішень. Сукупність цих заходів забезпечує стабільність, продуктивність і високу якість роботи додатку.

6.2 Підготовка до експлуатації додатку

Перед початком використання мобільного додатку «Vroomie» необхідно виконати ряд підготовчих дій, які забезпечать його коректну роботу:

а) підключення до інтернету. Для коректного функціонування додатку необхідне активне підключення до інтернету (Wi-Fi або мобільна мережа);

б) завантаження додатку. Користувач повинен завантажити додаток «Vroomie» на свій мобільний пристрій через офіційні магазини додатків:

- App Store – для пристроїв на базі iOS;
- Google Play – для пристроїв на базі Android.

в) запуск та реєстрація. Після встановлення користувач відкриває додаток та проходить процес реєстрації, ввівши необхідні дані (ім'я, номер телефону, електронну пошту тощо).

Після успішного виконання цих дій додаток готовий до використання.

6.3 Розробка тестової документації

Для перевірки коректної роботи мобільного додатку «Vroomie» було складено чек-лист тестування (таблиця 6.1). У ньому перераховані основні функції додатку, які підлягають перевірці на двох платформах: Android та iOS.

Тестування проводилося з метою виявлення можливих дефектів, перевірки стабільності роботи додатку та забезпечення відповідності функціональності заявленим вимогам.

Таблиця 6.1 – Чек-лист тестування додатку «Vroomie»

Перевірка	Android 10.0	IOS 17.8
Користувач		
1	2	3
Додаток завантажується та відкривається	Passed	Passed
Реєстрація нового користувача	Passed	Passed
Вхід у додаток із зареєстрованими даними	Passed	Passed
Відображення головної сторінки	Passed	Passed
Перегляд списку автомобілів	Passed	Passed
Перехід до деталей автомобіля	Passed	Passed
Вибір дати оренди	Passed	Passed
Вибір опції «з водієм» чи «без водія»	Passed	Passed
Перехід на сторінку оплати	Passed	Passed
Вибір методу оплати (Apple Pay/картка)	Passed	Passed
Завантаження орендованого авто до списку	Passed	Passed
Додавання авто до обраного	Passed	Passed

Продовження таблиці 6.1

1	2	3
Видалення авто з обраного	Passed	Passed
Надсилання відгуку про додаток	Passed	Passed
Відображення карти з локаціями авто	Passed	Passed
Перегляд профілю користувача	Passed	Passed
Оновлення фото профілю	Passed	Passed
Вихід із профілю	Passed	Passed
Адміністратор		
Вхід до профілю адміністратора	Passed	Passed
Додавання нового автомобіля	Passed	Passed
Видалення автомобіля	Passed	Passed
Відображення списку запитань від користувачів	Passed	Passed
Надсилання відповіді користувачу	Passed	Passed
Перехід до профілю адміністратора	Passed	Passed
Вихід із акаунта адміністратора	Passed	Passed

Всі перевірені функції працюють коректно на обох платформах, що свідчить про стабільну роботу додатку.

6.4 Методологія тестування та процес перевірки

Для забезпечення якісної роботи мобільного додатку «Vroomie» було застосовано системний підхід до тестування, який включає кілька рівнів перевірки функціональності, продуктивності та стабільності.

Методологія тестування.

В процесі тестування використовувалися наступні методи:

- функціональне тестування – перевірка відповідності функцій додатку заявленим вимогам;
- кросплатформне тестування – перевірка коректної роботи додатку на операційних системах Android та iOS;
- UI/UX тестування – перевірка коректного відображення інтерфейсу, логічності взаємодії елементів;
- тестування безпеки – перевірка механізмів входу та виходу, роботи з особистими даними;
- тестування навантаження – оцінка продуктивності додатку при високому навантаженні;
- тестування сумісності – перевірка стабільності додатку на різних пристроях з різними версіями ОС.

Процес перевірки.

Підготовка тестового середовища:

- встановлення додатку на тестові пристрої (Android, iOS);
- налаштування тестових акаунтів (користувач, адміністратор);
- перевірка підключення до мережі Інтернет.

Виконання тестування:

- послідовне виконання тестових сценаріїв згідно з чек-листом;
- перевірка кожного елемента функціоналу та взаємодії користувача з інтерфейсом;
- виявлення та фіксація дефектів.

Аналіз результатів тестування:

- оцінка відповідності функціоналу додатку до технічного завдання;
- усунення виявлених дефектів та повторне тестування (регресія);
- узагальнення результатів для висновків щодо якості продукту.

Використання комплексного підходу до тестування забезпечило стабільну та надійну роботу додатку «Vroomie» на всіх етапах експлуатації.

6.5 Аналіз результатів тестування

За результатами тестування мобільного додатку «Vroomie» на платформах Android та iOS, було підтверджено його стабільність, коректну роботу всіх функцій та зручність використання для кінцевого користувача.

Основні переваги додатку:

- стабільність роботи – додаток працює без збоїв, не викликає критичних помилок і швидко реагує на дії користувачів;
- зручність використання – інтерфейс інтуїтивно зрозумілий, що дозволяє користувачам легко орієнтуватися у функціоналі;
- кросплатформність – усі функції додатку повністю сумісні з операційними системами Android та iOS;
- швидкість завантаження – додаток швидко відкривається, сторінки відображаються без затримок;
- функціональність – коректна робота основних модулів: реєстрації, бронювання автомобілів, платежів, перегляду карт та взаємодії з адміністративною панеллю;
- сучасний дизайн – приємне візуальне оформлення та зручне розташування елементів інтерфейсу;
- ефективність роботи адміністративної панелі – адміністратори можуть легко додавати та видаляти автомобілі, а також відповідати на питання користувачів.

Таким чином, тестування підтвердило високу якість розробленого програмного забезпечення. Мобільний додаток «Vroomie» готовий до використання і забезпечує комфортний, надійний та функціональний сервіс для користувачів та адміністраторів.

6.6 Висновки за розділом 6

У цьому розділі було проведено аналіз експлуатації, тестування та експериментального дослідження мобільного додатку «Vroomie». Спочатку було визначено загальні положення щодо експлуатації додатку та необхідні умови для його використання, зокрема завантаження на смартфон, наявність інтернет-з'єднання та проходження реєстрації.

Далі було розроблено тестову документацію, що включала чек-лист для перевірки коректної роботи основних функцій програми на платформах Android та iOS. Усі критично важливі функції, такі як реєстрація користувачів, вибір та бронювання авто, оплата, робота з профілем та взаємодія адміністратора з системою, були протестовані й успішно підтвердили свою працездатність.

У процесі тестування було визначено, що додаток працює стабільно, без збоїв та критичних помилок. Функціонал відповідає заявленим вимогам, забезпечує зручність використання та швидку реакцію на дії користувачів. Адміністративна панель дозволяє ефективно взаємодіяти з автомобілями та обробляти запити користувачів.

Підсумовуючи, можна зробити висновок, що мобільний додаток «Vroomie» успішно пройшов тестування, відповідає вимогам експлуатації та готовий до повноцінного використання кінцевими користувачами та адміністраторами.

ВИСНОВКИ

У ході виконання даної роботи було проведено повний цикл розробки мобільного додатку «Vroomie» для оренди автомобілів. Спочатку було виконано аналіз предметної області, досліджено особливості онлайн-оренди авто, розглянуто існуючі аналоги додатків та сформульовано вимоги до майбутнього програмного продукту.

На основі проведеного аналізу було обґрунтовано вибір середовища розробки, архітектури, бази даних та мови програмування, що дозволило забезпечити оптимальну продуктивність, масштабованість та зручність використання додатку. Розроблено архітектурну схему програми, визначено принципи її роботи, створено концептуальну модель бази даних та реалізовано взаємодію між модулями системи.

Описано керівництво програміста, що включає інструкції з використання додатку, запуску, введення та виведення даних, а також системні повідомлення. Крім того, розроблено керівництво користувача, в якому описано вимоги до апаратного забезпечення, особливості роботи додатку, взаємодію з його функціоналом та інтерфейсом.

Проведено тестування додатку, яке включало підготовку до експлуатації, розробку тестової документації, опис методології перевірки та аналіз результатів. Випробування підтвердили коректну роботу усіх функцій додатку, стабільність системи, відсутність критичних помилок та відповідність заявленим вимогам.

Таким чином, розроблений мобільний додаток «Vroomie» повністю готовий до експлуатації. Він забезпечує зручність використання, високу швидкодію та інтуїтивно зрозумілий інтерфейс для кінцевих користувачів, а також надійний функціонал для адміністраторів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Uber Rent [Electronic resource]. – Access mode: <https://apps.apple.com/us/app/uber-request-a-ride/id368677368>.
2. Turo [Electronic resource]. – Access mode: <https://apps.apple.com/us/app/turo-car-rental-marketplace/id555063314>.
3. Getaround [Electronic resource]. – Access mode: <https://apps.apple.com/ua/app/getaround/id492238016?l=ru>.
4. Visual Studio [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio.
5. Mono Develop [Electronic resource]. – Access mode: <https://www.monodevelop.com/>.
6. Code::Blocks [Electronic resource]. – Access mode: <https://www.codeblocks.org/>.
7. JetBrains [Electronic resource]. – Access mode: <https://www.jetbrains.com/>.
8. MVVM (Model-View-ViewModel) [Електрон. ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Model-View-ViewModel>.
9. MVP (Model-View-Presenter) [Електрон. ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Model-View-Presenter>.
10. MVC (Model-View-Controller) [Електрон. ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-%D0%B2%D0%B8%D0%B4-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80>.
11. Clear architecture [Електрон. ресурс]. – Режим доступу: <https://habr.com/ua/companies/otus/articles/732178/>.
12. MongoDB [Electronic resource]. – Access mode: <https://www.mongodb.com/>.

13. MySQL [Electronic resource]. – Access mode:
<https://www.mysql.com/>.
14. PostgreSQL [Electronic resource]. – Access mode:
<https://www.postgresql.org/>.
15. Cassandra [Электрон. ресурс]. – Режим доступа:
https://uk.wikipedia.org/wiki/Apache_Cassandra
16. C# [Электрон. ресурс]. – Режим доступа:
https://uk.wikipedia.org/wiki/C_Sharp.
17. Java [Электрон. ресурс]. – Режим доступа:
<https://uk.wikipedia.org/wiki/Java>.
18. Python [Электрон. ресурс]. – Режим доступа:
<https://uk.wikipedia.org/wiki/Python>.
19. C++ [Электрон. ресурс]. – Режим доступа:
<https://uk.wikipedia.org/wiki/C%2B%2B>.

ДОДАТОК А
Технічне завдання

A.1 Вступ

Оренда автомобілів є важливою складовою сучасної транспортної системи, оскільки забезпечує користувачам зручність пересування без необхідності купівлі власного транспорту. Завдяки цифровим технологіям процес бронювання авто стає швидшим і доступнішим, а мобільні додатки відіграють ключову роль у спрощенні цього процесу.

Метою роботи є розробка мобільного додатку для оренди автомобілів, який дозволить користувачам легко знаходити, бронювати транспортні засоби та керувати своїми замовленнями. Для цього було проведено аналіз існуючих рішень, розроблено функціональну структуру та реалізовано основні можливості, зокрема авторизацію, пошук авто, оформлення бронювання, вибір оплати та управління профілем.

Результатом є створення зручного та безпечного мобільного додатку, який автоматизує процес оренди, скорочує час на оформлення замовлення та покращує взаємодію між компанією та користувачами. Це актуальне рішення, яке сприяє підвищенню доступності транспортних послуг та робить процес оренди швидким і комфортним.

A.2 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Мобільний додаток оренди авто «Vroomie», затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 р.

A.3 Ціль розробки

Метою розробки мобільного додатка «Vroomie» є автоматизація процесу оренди автомобілів, забезпечення зручного доступу до транспортних

засобів та покращення взаємодії між користувачами та сервісом прокату. Додаток дозволяє швидко знаходити, бронювати та керувати орендою авто без необхідності відвідування фізичних офісів компаній.

Основні цілі розробки:

- спрощення процесу оренди – надання користувачам можливості швидко та зручно знаходити авто відповідно до їхніх потреб;
- доступність 24/7 – забезпечення безперервного доступу до інформації про автомобілі, їх наявність та умови оренди;
- інтерактивність та персоналізація – можливість налаштовувати пошук авто за фільтрами, переглядати історію бронювань та керувати профілем;
- інтеграція карт та геолокації – відображення доступних авто на карті, вибір пункту видачі та повернення;
- гнучкість оплати – підтримка різних методів оплати, зокрема Apple Pay та карткових платежів;
- додаток «Vroomie» сприяє покращенню користувацького досвіду, економії часу та оптимізації роботи сервісів оренди автомобілів.

A.4 Характеристики та вимоги до програмного забезпечення

Мобільний додаток «Vroomie» повинен відповідати сучасним стандартам якості, забезпечуючи стабільну роботу, високу продуктивність і зручність у використанні. Додаток має бути розроблений з урахуванням функціональних та нефункціональних вимог, а також враховувати певні технічні обмеження.

A.4.1 Функціональні можливості

Функціональні вимоги визначають основні можливості додатка, які забезпечують його коректну роботу та відповідають потребам користувачів.

Додаток «Vroomie» має такі вимоги:

- реєстрація та авторизація користувачів із можливістю входу через e-mail або соціальні мережі;
- пошук автомобілів назвою та брендом;
- відображення карт із доступними авто та пунктами прокату;
- бронювання автомобіля з вибором дати, місця отримання та повернення;
- можливість вибору додаткових опцій (оренда з водієм);
- перегляд детальної інформації про авто (опис, фото, характеристики);
- оплата оренди через Apple Pay або банківську картку;
- управління профілем користувача, включаючи особисті дані та історію бронювань;
- функція оцінки та відгуків про автомобілі та сервіс;
- підтримка адміністративної панелі для управління базою автомобілів.

A.4.2 Нефункціональні характеристики

Нефункціональні вимоги визначають якісні характеристики системи, що впливають на її продуктивність, безпеку та зручність використання.

Додаток «Vroomie» має такі вимоги:

- інтуїтивно зрозумілий інтерфейс для зручної навігації;
- високий рівень безпеки персональних даних і платіжної інформації;
- стабільна робота без критичних збоїв;

- оптимізоване завантаження даних для швидкого відгуку інтерфейсу;
- сумісність із сучасними мобільними платформами (IOS, Android);
- мінімальне використання ресурсів пристрою.

A.4.3 Обмеження та обсяг застосування

Для забезпечення стабільної роботи та оптимізації продуктивності додатка слід враховувати певні технічні та операційні обмеження. Додаток «Vroomie» має такі обмеження:

- доступність програми лише на мобільних пристроях;
- необхідність активного інтернет-з'єднання для роботи з сервером;
- обмеження за мінімальною версією ос (IOS 15+, Android 8.0+).

A.5 Вимоги до супровідної документації

Для ефективного використання, адміністрування та підтримки мобільного додатка «Vroomie» необхідно розробити комплект супровідної документації. Вона повинна містити детальні інструкції щодо встановлення, налаштування, використання та технічного обслуговування програмного забезпечення.

До основного комплекту документації входять:

- керівництво користувача – опис можливостей додатка, покрокові інструкції щодо реєстрації, бронювання авто, оплати, управління профілем та використання додаткових функцій;
- технічна документація – структура додатка, архітектурні рішення, технологічні особливості, вимоги до серверної частини та безпеки;
- керівництво адміністратора – інструкції з управління базою даних авто та обробки запитів користувачів;

- документація для тестування – тест-кейси, звіти про тестування, сценарії перевірки функціональності та безпеки;
- вся документація повинна бути структурованою, зрозумілою та актуальною, що дозволить забезпечити ефективне використання додатка, його технічну підтримку та подальший розвиток.

A.6 Оцінка ефективності

Розробка мобільного додатка «Vroomie» спрямована на оптимізацію процесу оренди автомобілів, що дозволяє підвищити рівень зручності для користувачів та покращити ефективність управління автопарком.

Ефективність додатка оцінюється за такими ключовими параметрами:

- простота використання – інтуїтивний інтерфейс та зручна система навігації дозволяють користувачам швидко знаходити та бронювати автомобілі;
- автоматизація процесів – зменшення часу на оформлення оренди завдяки інтеграції з базою автомобілів, платіжними сервісами та картографічними системами;
- зменшення операційних витрат – зниження витрат на обслуговування клієнтів через автоматизацію більшості процесів;
- безпека та надійність – реалізація сучасних механізмів захисту персональних даних та фінансових операцій.

A.7 Процедура перевірки та затвердження

Процедура перевірки та затвердження розробленого додатка «Vroomie» включає кілька етапів, що забезпечують повну перевірку відповідності вимогам та характеристикам, зазначеним у технічному завданні. На кожному етапі перевіряються функціональні можливості додатка, а також його стабільність та ефективність в реальних умовах експлуатації.

A.7.1 Етапи перевірки

Процес перевірки складається з наступних етапів:

- попередня перевірка – перевірка виконання основних вимог до програми на етапі її розробки. Це включає перевірку коду, інтерфейсу, бази даних та інтеграцій з іншими сервісами;
- тестування функціональності – проведення тестування за допомогою різноманітних методів, зокрема функціонального тестування, юзабіліті тестування, тестування на сумісність із різними операційними системами та пристроями;
- тестування безпеки – перевірка механізмів захисту персональних даних, фінансових транзакцій та інших критичних функцій додатка;
- перевірка на відповідність вимогам – фінальна перевірка того, чи відповідає розроблений додаток вимогам, описаним у технічному завданні.

A.7.2 Умови та критерії затвердження

Для прийняття додатка необхідно виконати наступні умови:

- всі функціональні можливості повинні бути реалізовані згідно з технічним завданням;
- всі системи безпеки повинні забезпечувати належний рівень захисту даних користувачів;
- інтерфейс додатка має бути інтуїтивно зрозумілим та зручним для користувача;
- додаток має стабільно працювати на основних мобільних платформах (Android, iOS) без значних помилок та збоїв;
- всі тести повинні бути пройдені без критичних помилок.

A.7.3 Перелік документації для затвердження

Для завершення процедури приймання розробки необхідно надати наступні документи:

- повний звіт про тестування додатка, включаючи результати всіх проведених тестів;
- технічну документацію, що містить опис архітектури додатка, використаних технологій та алгоритмів;
- документацію щодо безпеки, яка підтверджує належний рівень захисту персональних даних і фінансових операцій;
- інструкцію для користувачів, що описує основні функції додатка та процеси взаємодії з ним.

A.7.4 Процес приймання та оцінки

Процедура приймання додатка включає наступні кроки:

- оформлення акта приймання-передачі, що підтверджує завершення розробки та готовність до використання додатка;
- перевірка всіх компонентів додатка на відповідність технічним вимогам та критеріям приймання;
- ознайомлення замовника з результатами тестування та документацією;
- підписання акта приймання, що підтверджує готовність додатка до запуску в експлуатацію.

Таким чином, після виконання всіх етапів перевірки та затвердження, додаток «Vroomie» буде готовий до подальшої експлуатації та використання кінцевими користувачами.

ДОДАТОК Б
Текст програми

Б.1 – Логіка роботи користувача

Б.1.1 – AboutVroomie.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace Vroomie.View
{
    /// <summary>
    /// Логіка взаємодії для AboutVroomie.xaml
    /// </summary>
    public partial class AboutVroomie : Page
    {
        public AboutVroomie()
        {
            InitializeComponent();

            //Time
            DateTime currentTime = DateTime.Now;
            string formattedTime = currentTime.ToString("HH:mm");
            Time.Content = formattedTime;
        }

        private void AboutVroomie_Click(object sender, RoutedEventArgs e)
        {
            GridAllInfo.Visibility = Visibility.Visible;
            allinfoPhoto.Source = new BitmapImage(new Uri("D:/Универ/4
курс/Diplom/Vroomie/Vroomie/Photo/Vroomie.png", UriKind.Absolute));
            allinfoTitle.Text = "Vroomie";
            allinfoText.Text = "Vroomie is one of the best mobile car rental apps that allows
users to choose and pay for car rental in a convenient online format." +
"The main features include the selection of cars by category, the
choice of a convenient day, time, " +
"and the possibility of riding with a driver, as well as a flexible
payment system.";
        }

        private void Cars_Click(object sender, RoutedEventArgs e)

```

```

        {
            GridAllInfo.Visibility = Visibility.Visible;
            allinfoPhoto.Source = new BitmapImage(new Uri("D:/Универ/4
курс/Diplom/Vroomie/Vroomie/Photo/Cars.png", UriKind.Absolute));
            allinfoTitle.Text = "Cars";
            allinfoText.Text = "Luxury cars at Vroomie are a combination of elegance,
comfort and advanced technology. " +
                "Our collection includes premium car brands such as Bentley, Ferrari,
Lamborghini, BMW, Audi, Mercedes, and others that meet the highest quality standards.";
        }

        private void Developer_Click(object sender, RoutedEventArgs e)
        {
            GridAllInfo.Visibility = Visibility.Visible;
            allinfoPhoto.Source = new BitmapImage(new Uri("D:/Универ/4
курс/Diplom/Vroomie/Vroomie/Photo/Developer.png", UriKind.Absolute));
            allinfoTitle.Text = "Developer";
            allinfoText.Text = "Vroomie's developer, Yan Lavrov, created this mobile app
for his 4th year graduation project. " +
                "His goal was to provide a convenient and modern tool for car rental,
combining an innovative approach with practical features for users.";
        }

        private void HideGrid_Click(object sender, RoutedEventArgs e)
        {
            GridAllInfo.Visibility = Visibility.Collapsed;
            allinfoTitle.Text = "";
            allinfoText.Text = "";
        }
    }
}

```

Б.1.2 – AllCars.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Runtime.ConstrainedExecution;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;

```

```

using System.Windows.Shapes;
using Vroomie.Models;

namespace Vroomie.View
{
    /// <summary>
    /// Логіка взаємодії для AllCars.xaml
    /// </summary>
    public partial class AllCars : Page
    {
        DataBase _data;
        public string SelectedBrand { get; set; }
        public ObservableCollection<Cars> FilteredCars { get; set; }

        public AllCars(string brand)
        {
            InitializeComponent();
            _data = new DataBase();
            SelectedBrand = brand;
            FilteredCars = new ObservableCollection<Cars>();
            DataContext = this;
            LoadCars();

            //Time
            DateTime currentTime = DateTime.Now;
            string formattedTime = currentTime.ToString("HH:mm");
            Time.Content = formattedTime;

            NameCar.Text = brand;
        }

        private async void LoadCars()
        {
            var carsFromDb = await _data.GetAllCars();

            foreach (var car in carsFromDb.Where(c => c.Name.StartsWith(SelectedBrand)))
            {
                FilteredCars.Add(car);
            }
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            Home home = new Home();
            Frame mainFrame = Application.Current.MainWindow.Content as Frame;
            mainFrame.NavigatonService.Navigate(home);
        }

        private void CardButton_Click(object sender, RoutedEventArgs e)
        {
            var button = sender as Button;
            var car = button?.Tag as Cars;

```

```

        if (car != null)
        {
            CarDetails carDetails = new CarDetails(car, car.PriceDay);
            Frame mainFrame = Application.Current.MainWindow.Content as Frame;
            mainFrame.NavigationService.Navigate(carDetails);
        }
    }
}

```

B.1.3 – CarDetails.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.ConstrainedExecution;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Vroomie.Models;
using static Vroomie.View.Login;

namespace Vroomie.View
{
    /// <summary>
    /// Interaction logic for CarDetails.xaml
    /// </summary>
    public partial class CarDetails : Page
    {
        DataBase _dataBase;
        private double _startX;
        private bool _isDragging;
        private Cars _selectedCar;
        public CarDetails(Cars selectedCar, string priceDay)
        {
            InitializeComponent();
            _selectedCar = selectedCar;
            _dataBase = new DataBase();

            //Time
            DateTime currentTime = DateTime.Now;

```

```

        string formattedTime = currentTime.ToString("HH:mm");
        Time.Content = formattedTime;
        Time.FontFamily = new FontFamily("Monda");

        SwipeButtonContainer.MouseLeftButtonDown +=
SwipeButton_MouseLeftButtonDown;
        SwipeButtonContainer.MouseMove += SwipeButton_MouseMove;
        SwipeButtonContainer.MouseLeftButtonUp +=
SwipeButton_MouseLeftButtonUp;

        imageCar.Source = new BitmapImage(new Uri(_selectedCar.PhotoCarOne,
UriKind.Absolute));
        NameCar.Text = _selectedCar.Name;
        Reviews.Text = _selectedCar.Review;
        fuelType.Text = _selectedCar.FuelType;
        engine.Text = _selectedCar.Motor;
        drive.Text = _selectedCar.Drive;
        kmHr.Text = _selectedCar.MaxSpeed;
        priceDayText.Text = priceDay;
    }

    private void SwipeButton_MouseLeftButtonDown(object sender,
MouseEventArgs e)
    {
        _isDragging = true;
        _startX = e.GetPosition(SwipeArea).X;
        SwipeButtonContainer.CaptureMouse();
        SwipeText.Visibility = Visibility.Hidden;
        SwipeImage.Visibility = Visibility.Hidden;
    }

    private void SwipeButton_MouseMove(object sender, MouseEventArgs e)
    {
        if (_isDragging)
        {
            var posX = e.GetPosition(SwipeArea).X;

            if (posX >= 0 && posX <= SwipeArea.ActualWidth -
SwipeButtonContainer.Width)
            {
                Canvas.SetLeft(SwipeButtonContainer, posX);
            }
        }
    }

    private void SwipeButton_MouseLeftButtonUp(object sender,
MouseEventArgs e)
    {
        _isDragging = false;
        SwipeButtonContainer.ReleaseMouseCapture();
    }

```

```

        if (Canvas.GetLeft(SwipeButtonContainer) >= SwipeArea.ActualWidth -
SwipeButtonContainer.Width - 10)
        {
            SelectLocation selectLocation = new SelectLocation(priceDayText.Text,
_selectedCar.PhotoCarOne, NameCar.Text);
            Frame mainFrame = Application.Current.MainWindow.Content as Frame;
            mainFrame.NavigationService.Navigate(selectLocation);
        }
        else
        {
            var animation = new DoubleAnimation
            {
                From = Canvas.GetLeft(SwipeButtonContainer),
                To = 0,
                Duration = TimeSpan.FromMilliseconds(300)
            };

            SwipeButtonContainer.BeginAnimation(Canvas.LeftProperty, animation);
        }
    }

    private async void AddToFavourites_Click(object sender, RoutedEventArgs e)
    {
        string userId = UserSession.UserId;
        FavouriteCar favouriteCar = new FavouriteCar(_selectedCar.Name,
_selectedCar.PhotoCarTwo, _selectedCar.NumberSeats, _selectedCar.MaxSpeed,
_selectedCar.PriceDay, userId);
        await _dataBase.AddToFavourites(favouriteCar);

        messageMotification.Text = $"Car added to favourites";
        BorderNotification.Visibility = Visibility.Visible;
        gridNotification.Visibility = Visibility.Visible;
    }

    private void Button_Click_2(object sender, RoutedEventArgs e)
    {
        BorderNotification.Visibility = Visibility.Collapsed;
        gridNotification.Visibility = Visibility.Collapsed;
    }
}
}

```

B.1.4 – Favourites.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;

```

```

using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Vroomie.Models;

namespace Vroomie.View
{
    /// <summary>
    /// Логіка взаємодії для Favourites.xaml
    /// </summary>
    public partial class Favourites : Page
    {
        DataBase _data;
        public Favourites()
        {
            InitializeComponent();
            _data = new DataBase();

            //Time
            DateTime currentTime = DateTime.Now;
            string formattedTime = currentTime.ToString("HH:mm");
            Time.Content = formattedTime;

            LoadFavourites();
        }
        public void LoadFavourites()
        {
            var favourites = _data.GetFavourites();
            FavouritesListBox.ItemsSource = favourites;
        }

        private async void DeleteCar_Click(object sender, RoutedEventArgs e)
        {
            var button = sender as Button;
            var carName = button?.Tag as string;

            if (!string.IsNullOrEmpty(carName))
            {
                await _data.RemoveFromFavourites(carName);

                LoadFavourites();

                MessageBox.Show($"Автомобиль '{carName}' был удалён из
избранного!");
            }
            else
            {

```

```

        MessageBox.Show("Ошибка: Не удалось определить выбранный
автомобиль.");
    }
}

private void Button_Click(object sender, RoutedEventArgs e)
{
}
}
}

```

Б.1.5 – Help.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace Vroomie.View
{
    /// <summary>
    /// Логіка взаємодії для Help.xaml
    /// </summary>
    public partial class Help : Page
    {
        public Help()
        {
            InitializeComponent();
            //Time
            DateTime currentTime = DateTime.Now;
            string formattedTime = currentTime.ToString("HH:mm");
            Time.Content = formattedTime;
        }

        private void UserInput_TextChanged(object sender, TextChangedEventArgs e)
        {
            userMessages.Content = userMessageInput.Text;
        }
    }
}

```

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    gridUserMessage.Visibility = Visibility.Visible;
}

private void ShareApp_Click(object sender, RoutedEventArgs e)
{
}

private void RateApp_Click(object sender, RoutedEventArgs e)
{
}
}
```

ДОДАТОК В
Слайди презентації

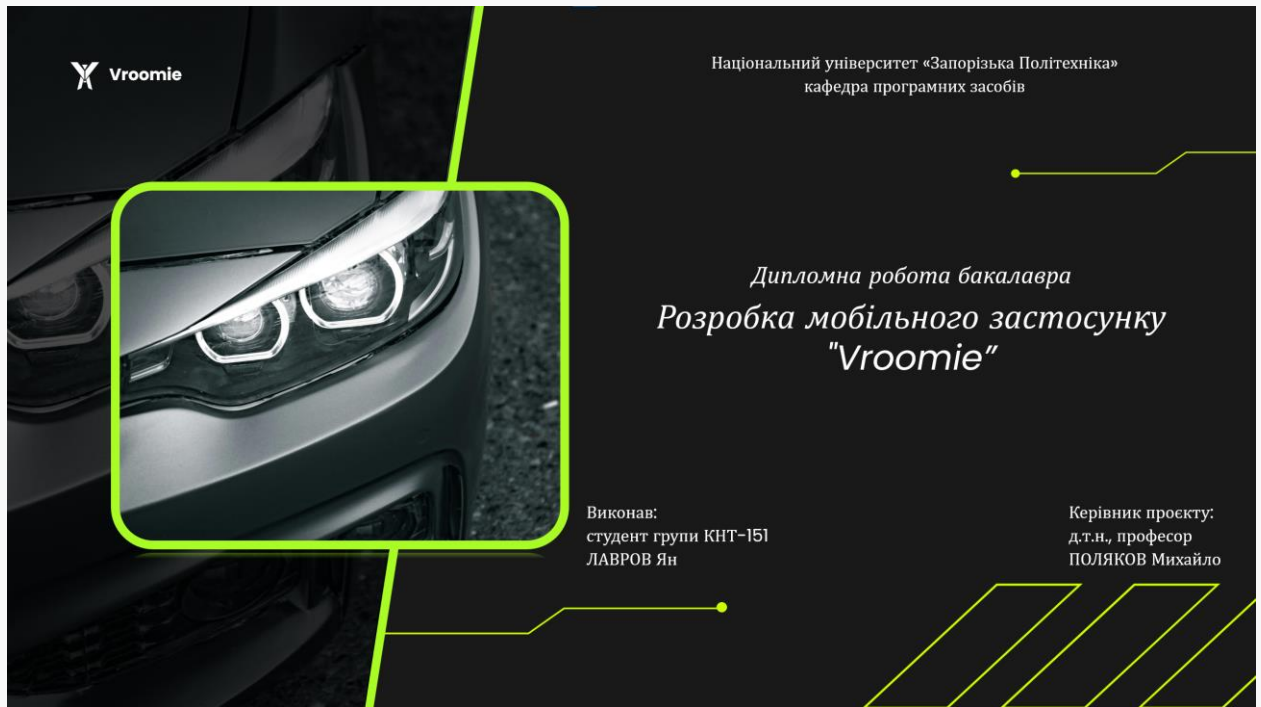


Рисунок В.1 – Слайд №1

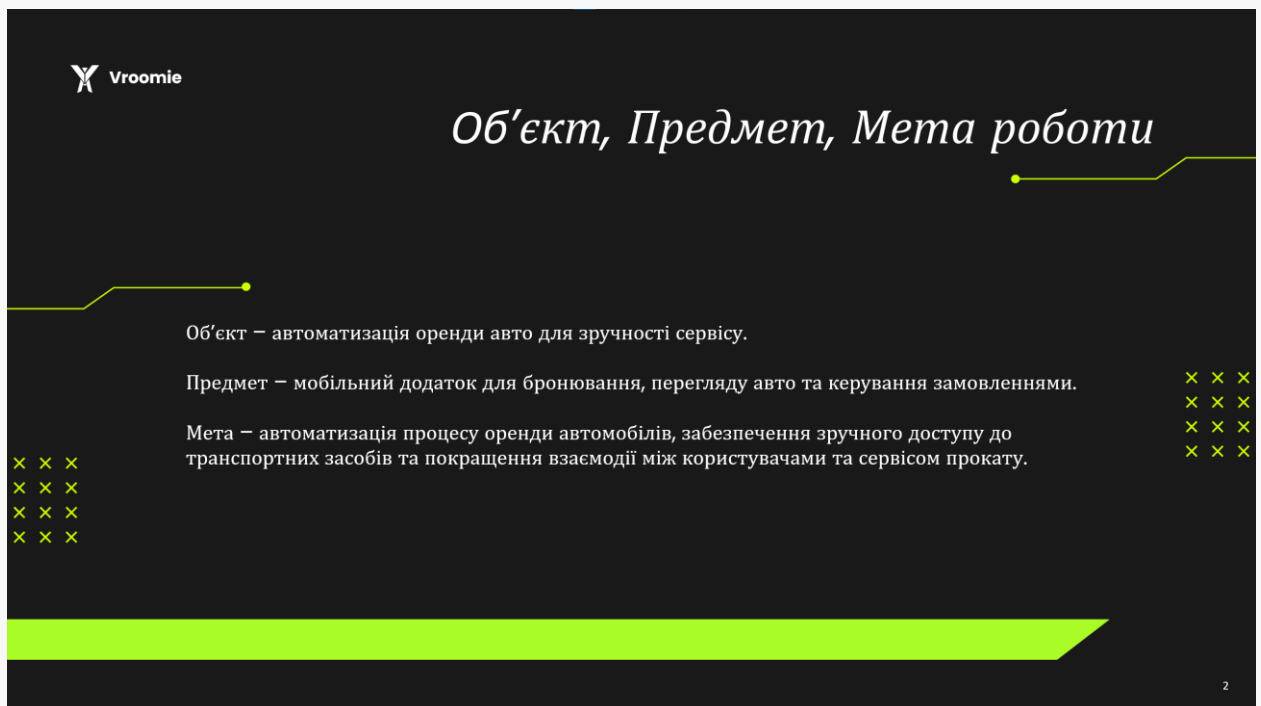


Рисунок В.2 – Слайд №2

 **Vroomie**

Завдання до роботи

Для досягнення поставленої мети необхідно провести аналіз сучасних рішень у сфері оренди автомобілів, розробити функціональну структуру додатку, реалізувати основні можливості, такі як:

- ✓ авторизація користувачів;
- ✓ пошук авто;
- ✓ оформлення бронювання;
- ✓ вибір способу оплати та управління профілем користувача.

Також важливо забезпечити високий рівень безпеки даних та зручність інтерфейсу.

3

Рисунок В.3 – Слайд №3

 **Vroomie**

Аналоги програмного продукту

Критерії	Vroomie	Uber Rent	Turo	Getaround
Вибір автомобілів різних брендів	+	+	+	-
Можливість додавання авто до улюблених	+	-	-	+
Інтерактивна мапа	+	-	-	-
Можливість вибору авто з водієм	+	-	-	+
Персоналізоване управління бронюванням	+	-	-	-
Офлайн-доступ до бронювань	+	-	+	-
Зв'язок з технічною підтримкою прямо у додатку	+	-	-	-

4

Рисунок В.4 – Слайд №4

Аналоги середовищ розробки

Критерії	Visual Studio	Mono Develop	Code :: Blocks	JetBrains Rider
Підтримка C# та .NET	+	+	-	+
Кросплатформеність	+	+	-	-
Інструменти налагодження	+	-	+	+
Підтримка GUI - розробки	+	+	-	+
Підтримка багатомовності	+	-	-	+
Інтеграція з Git	+	+	-	+
Розширюваність (плагіни)	+	+	-	-

5

Рисунок В.5 – Слайд №5

Аналоги архітектур проєктування

Критерії	MVVM	MVP	MVC	Clean Architecture
Простота реалізації	+	+	+	-
Прив'язка даних	+	-	-	+
Тестованість	+	+	-	-
Гнучкість змін інтерфейсу	+	-	-	+
Складність підтримки	+	+	-	-
Ізоляція компонентів	+	+	-	+
Масштабованість	+	+	+	-

6

Рисунок В.6 – Слайд №6

Vroomie

Аналоги баз даних

Критерії	MongoDB	MySQL	PostgreSQL	Cassandra
Масштабованість	+	–	–	+
Продуктивність	+	+	+	–
Підтримка транзакцій	+	+	–	+
Гнучкість запитів	+	+	–	+
Сумісність з JSON	+	–	+	+
Інтеграція з фреймворками	+	+	–	+
Керування та адміністрування	+	+	–	–

7

Рисунок В.7 – Слайд №7

Vroomie

Аналоги мов програмування

Критерії	C# (WPF)	Java	Python	C++
Простота синтаксису	+	–	+	–
Продуктивність	+	+	–	+
Розробка інтерфейсу	+	–	–	–
Мобільна розробка	+	+	–	–
Підтримка багатозадачності	+	+	–	+
Платформонезалежність	+	+	+	–

8

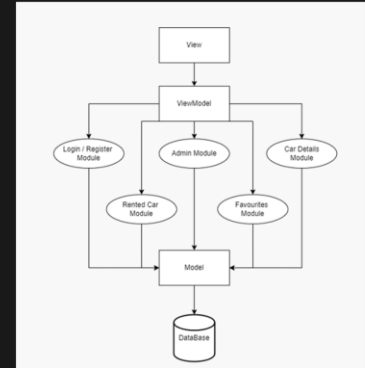
Рисунок В.8 – Слайд №8

Архітектура MVVM додатку

Розроблена система «Vroomie» побудована на основі архітектурного патерну MVVM (Model-View-ViewModel), що дозволяє розділити логіку, забезпечуючи модульність та зручність розширення.

Основні компоненти системи:

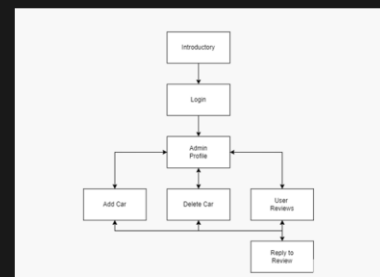
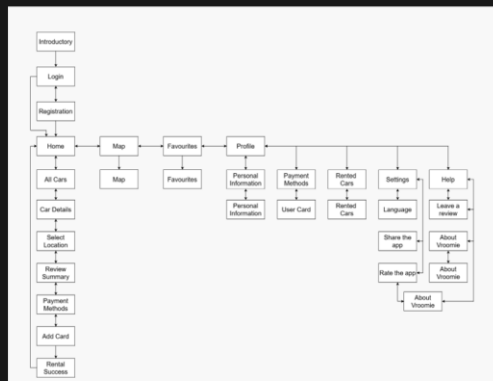
- ✓ View;
- ✓ View-Model;
- ✓ Model.



9

Рисунок В.9 – Слайд №9

Схеми функціонування додатку



10

Рисунок В.10 – Слайд №10

Діаграма класів додатку

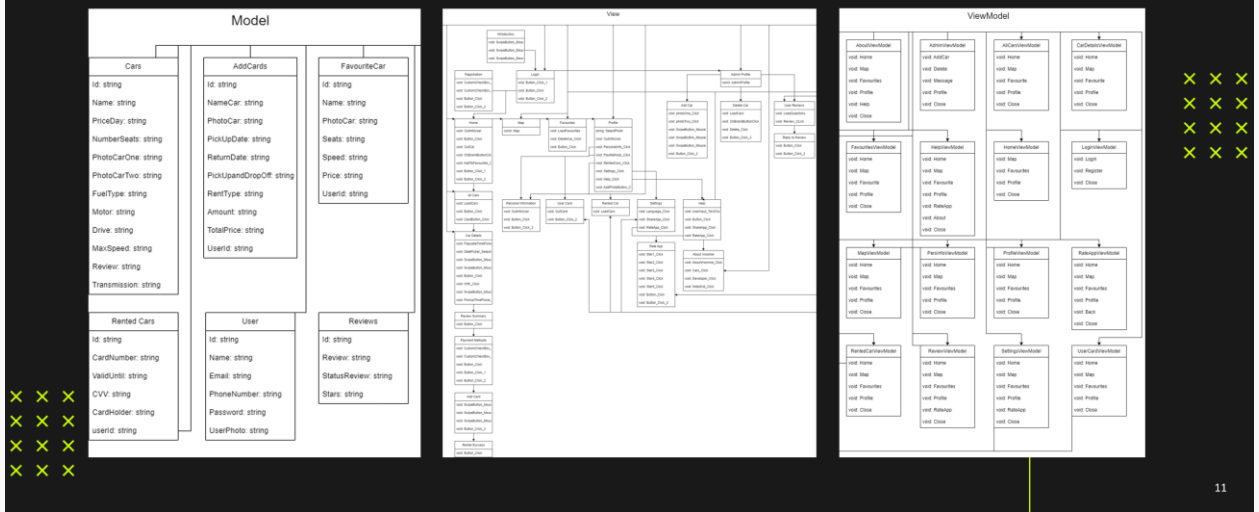


Рисунок В.11 – Слайд №11

Діаграма візуалізації компонентів

Візуалізація компонентів системи є важливою для розуміння структури та взаємодії основних частин програми. Вона демонструє, як окремі модулі та компоненти, такі як інтерфейс користувача, база даних, платіжні системи та серверна частина, взаємодіють між собою для забезпечення безперебійної роботи додатку оренди авто «Vroomie».

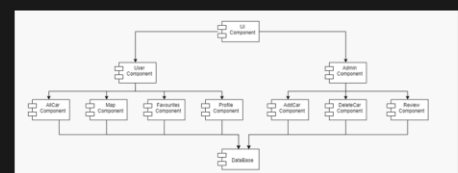
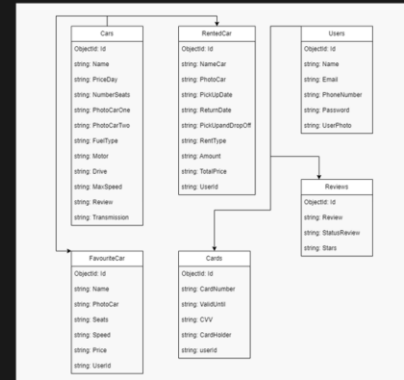


Рисунок В.12 – Слайд №12

Концептуальна модель БД

Концептуальна модель бази даних є основою для проектування та організації зберігання даних у системі оренди авто «Vroomie».

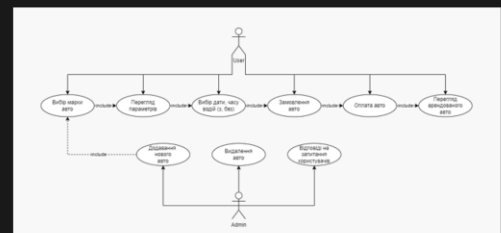


13

Рисунок В.13 – Слайд №13

Діаграма випадків використання

Діаграма випадків використання є важливим інструментом для візуалізації основних сценаріїв взаємодії користувачів з системою оренди авто «Vroomie».



14

Рисунок В.14 – Слайд №14

 **Vroomie**

Методологія тестування та процес перевірки

Для забезпечення якісної роботи мобільного додатку «Vroomie» було застосовано системний підхід до тестування, який включає кілька рівнів перевірки функціональності, продуктивності та стабільності.

Методологія тестування

В процесі тестування використовувалися наступні методи:

- ✓ функціональне тестування – перевірка відповідності функцій додатку заявленим вимогам;
- ✓ кросплатформне тестування – перевірка коректної роботи додатку на операційних системах Android та iOS;
- ✓ UI/UX тестування – перевірка коректного відображення інтерфейсу, логічності взаємодії елементів.



15

Рисунок В.15 – Слайд №15

 **Vroomie**

× × ×
× × ×
× × ×
× × ×



Стартове вікно додатку «Vroomie»

× × × ×
× × × ×
× × × ×
× × × ×



16

Рисунок В.16 – Слайд №16

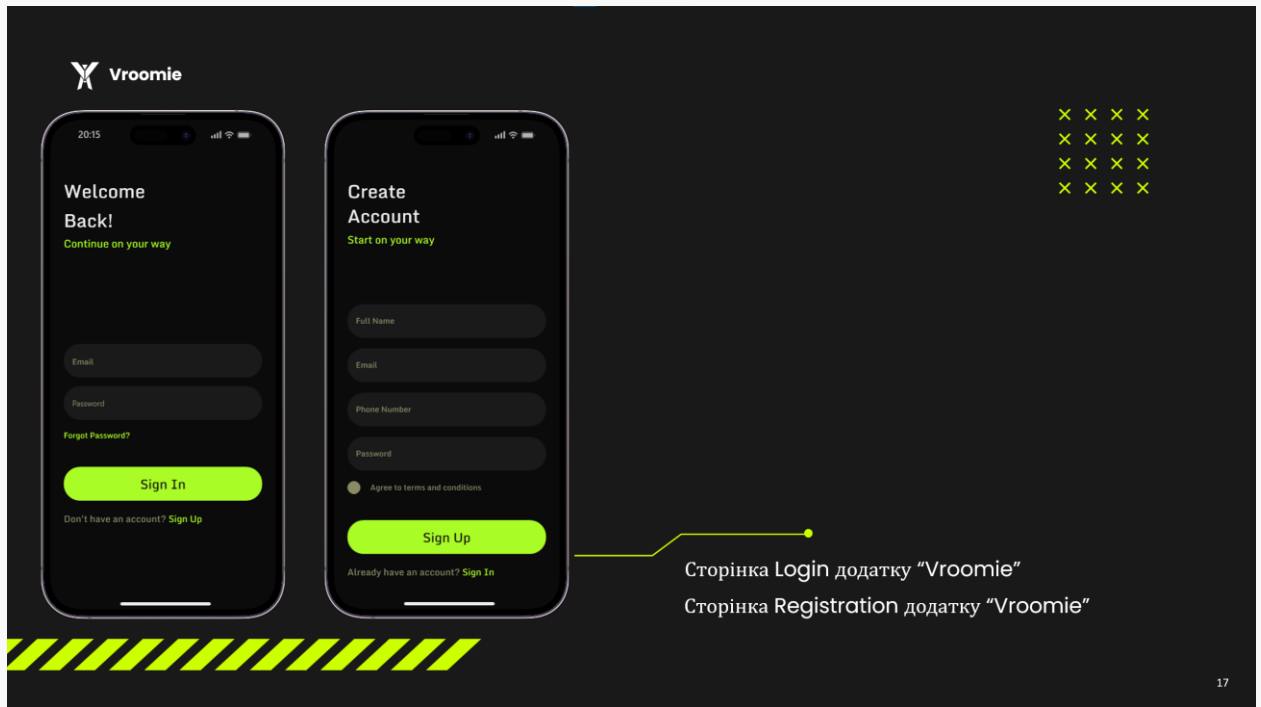


Рисунок В.17 – Слайд №17

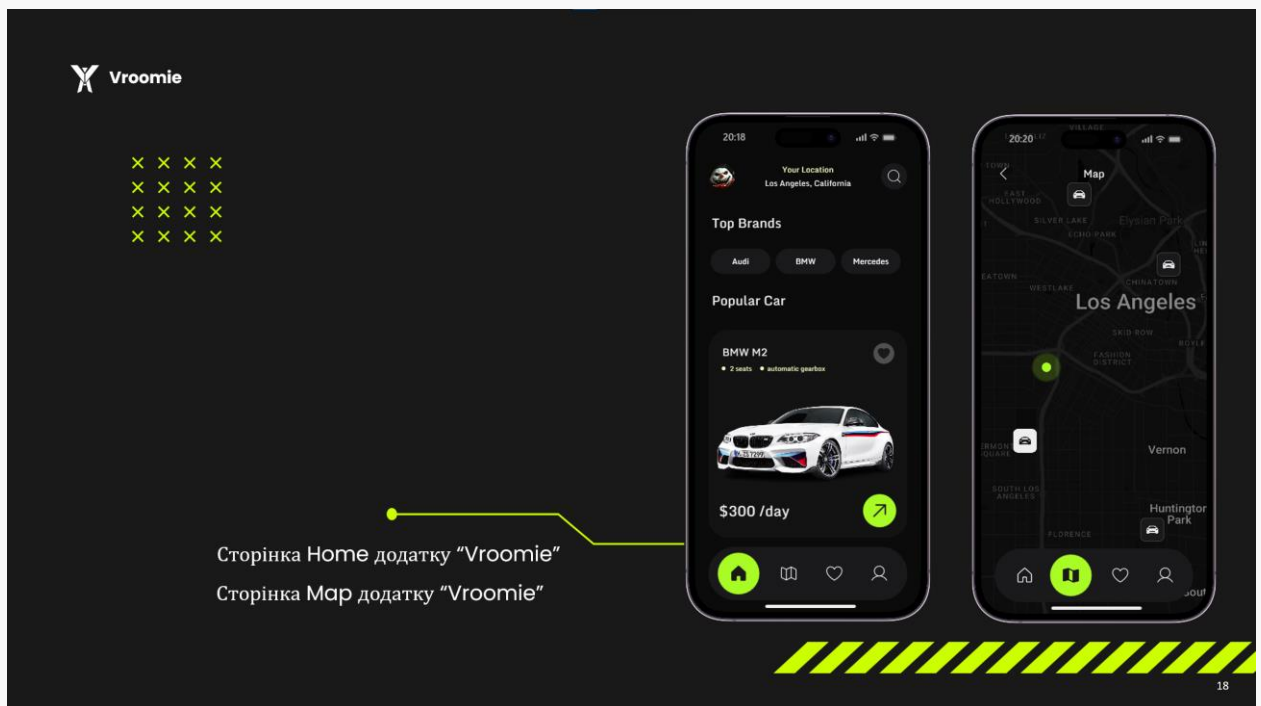


Рисунок В.18 – Слайд №18



Висновки

У результаті виконання роботи над додатком «Vroomie» було:

- ✓ проведено аналіз аналогічних застосунків, мов програмування, СУБД, середовищ розробки та архітектур;
- ✓ визначено об'єкт, предмет і мету розробки додатку для оренди авто;
- ✓ сформульовано основні завдання, що охоплюють функціональність і зручність користування ;
- ✓ побудовано діаграми: класів, компонентів, випадків використання, а також схеми функціонування;
- ✓ створено інтерфейсні екрани, орієнтовані на зручність користувача.

19

Рисунок В.19 – Слайд №19