

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Розробка сервісу каршерінгу»

Виконав студент групи ПЗк-21-1
СПАСЬОНОВ Артем Русланович

Керівник роботи:
КУБРАК Юрій Олександрович

Консультант:
ЄФРЕМОВ Юрій Миколайович

Рецензент:
ТОЛСТОЙ Ігор Анатолійович

Житомир – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
26 лютого 2024р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **Спасьонов Артем Русланович**

Керівник роботи: КУБРАК Юрій Олександрович

Тема роботи: **«Розробка сервісу каршерінгу»**, затверджена Наказом закладу вищої освіти №74/с від 23.02.2024р.

Вихідні дані для роботи: розробка сервісу каршерінгу

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Кубрак Ю.О., Єфремов Ю.М.	27.02.2024р.	27.02.2024р.
2	Кубрак Ю.О., Єфремов Ю.М.	27.02.2024р.	27.02.2024р.
3	Кубрак Ю.О., Єфремов Ю.М.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з сервісу каршерінгу та пояснювальної записки.

Текстова частина викладена на 71 сторінках друкованого тексту. Пояснювальна записка має 13 сторінок додатків. Список використаних джерел містить 20 найменувань і займає 2 сторінки. У роботі наведено 50 рисунків та 9 таблиць. Загальний обсяг роботи – 71 сторінок.

У першому розділі було детально проаналізовано потреби цільової аудиторії та визначено оптимальні засоби для реалізації розробки сервісу каршерінгу. В результаті цього аналізу було вибрано ключові технології.

У другому розділі проведено детальне проєктування архітектури сервісу. Була розроблена об'єктно-орієнтована модель сервісу, в якій описано взаємодію основних компонентів та їх функціональні можливості.

Третій розділ був присвячений фактичній реалізації сервісу каршерінгу. Розроблено інтерфейс користувача, який включає ключові елементи для зручного користування сервісом, такі як пошук автомобілів, бронювання та оплата послуг. Було проведено функціональне тестування для перевірки працездатності сервісу та його відповідності вимогам.

У висновку підкреслено, що розроблений сервіс каршерінгу є сучасним, функціональним і відповідає стандартам якості.

У додатку представлені лістинги коду, які ілюструють структуру та ключові аспекти реалізації сервісу.

Ключові слова: REACT, PRISMA, WEB-САЙТ, MVC, СЕРВЕР, АДМІН, АВТОРИЗАЦІЯ, ДАНІ.

					ІПЗ.КР.Б – 121 – 24 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		А.Р. Спасьонов			Розробка сервісу каршерінгу	Літ.		Арк.	Аркушів
Керівник		Ю.О. Кубрак						3	71
Консультант		Ю.М. Єфремов				Житомирська політехніка, група ІПЗк-21-1			
Рецензент		І.А. Толстой							
Зав. каф.		Т.А. Вакалюк							

ABSTRACT

The qualification graduation thesis consists of a car-sharing service and an explanatory note.

The text part is set out on 71 pages of printed text. The explanatory note has 13 pages of appendices. The list of references includes 20 titles and occupies 2 pages. The paper contains 50 figures and 9 tables. The total volume of the paper is 71 pages.

In the first chapter, the needs of the target audience were analyzed in detail and the optimal means for implementing the development of a car-sharing service were determined. As a result of this analysis, key technologies were selected.

In the second section, we carried out a detailed design of the system architecture. We developed an object-oriented service model that describes the interaction of the main components and their functionality.

The third section was devoted to the actual implementation of the car sharing service. A user interface has been developed that includes key elements for convenient use of the service, such as car search, booking, and payment. Functional testing was conducted to verify the system's performance and compliance with the requirements.

The conclusion emphasizes that the developed car sharing service is modern, functional and meets quality standards.

The appendix contains code listings that illustrate the structure and key aspects of the service implementation.

Keywords: REACT, PRISMA, WEBSITE, MVC, SERVER, ADMIN, AUTHORIZATION, DATA.

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	4
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП	7
РОЗДІЛ 1. ПОСТАНОВКА ТА АНАЛІЗ ЗАДАЧІ	9
1.1. Постановка задачі	9
1.2. Аналіз існуючих аналогів	10
1.3. Обґрунтування вибору інструментальних засобів	15
1.4. Технічне завдання на розробку сервісу	17
Висновки до першого розділу	18
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА СЕРВІСУ	19
2.1. Аналіз вимог до сервісу	19
2.2. Алгоритм роботи та стани сервісу	21
2.3. Об'єктно-орієнтована модель сервісу	23
2.4. Взаємодія об'єктів сервісу	25
2.5. Розробка бази даних сервісу	26
Висновки до другого розділу	29
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СЕРВІСОМ	30
3.1. Порядок встановлення та налаштування сервісу	30
3.2. Розробка основних функцій сервісу	32
3.3. Опис інструкції користувача	42
3.4. Тестування роботи сервісу	52
Висновки до третього розділу	55
ВИСНОВКИ	56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	57
ДОДАТКИ	59

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

TS – TypeScript.

БД – База даних.

ПЗ – Програмне забезпечення.

ПК – Персональний комп'ютер.

СУБД – Система управління базами даних.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Сучасне життя та процес урбанізації кардинально змінили наші пріоритети, змусивши нас цінувати час більше, ніж будь-коли раніше. У зв'язку з цим, багато хто віддає перевагу автомобілю перед громадським транспортом як найшвидшому засобу пересування в умовах міста. Відсутність власного автомобіля більше не є перешкодою для комфортних поїздок, оскільки оренда автомобіля стала повсякденною практикою для тих, хто часто подорожує у справах, у туристичних цілях або просто звик до зручності особистого транспорту. Проте останнім часом все більше популярності набуває каршерінг – інноваційна послуга, яка відрізняється від традиційної оренди автомобіля своїми перевагами та доступністю. Каршерінг дозволяє швидко та зручно орендувати автомобіль на короткий період, що особливо важливо в умовах сучасного ритму життя [1].

Розробка сучасного сервісу каршерінгу з використанням новітніх технологій, таких як Next.js та React.js є актуальною задачею. Ці технології дозволяють створювати високопродуктивні, безпечні та зручні у використанні сервіси, що відповідають сучасним вимогам до веброзробки і користувацького досвіду. Також важливою є інтеграція з платіжними системами для забезпечення швидкої та безпечної оплати орендованих послуг.

Мета даної кваліфікаційної роботи є розробка функціонального та зручного сервісу каршерінгу, який забезпечить користувачам зручний доступ до послуг оренди автомобілів, дозволить швидко бронювати та оплачувати оренду, а також надасть адміністраторам інструменти для управління автопарком та користувачами. Для досягнення цієї мети сформовані наступні завдання:

1. Аналіз існуючих аналогів.
2. Формування вимог до функціоналу сайту згідно потреб користувачів.
3. Створення інтуїтивно зрозумілого інтерфейсу користувача, який дозволяє швидко знаходити та бронювати доступні автомобілі.
4. Інтеграцію з платіжною системою Stripe для забезпечення безпечних та зручних платежів.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

5. Використання сучасних технологій та фреймворків для забезпечення високої продуктивності та масштабованості сервісу.
6. Розробку адміністративної панелі для управління автопарком, користувачами та бронюваннями.

Об’єктом дослідження є сервіси каршерінгу, їх функціональні можливості, користувацький інтерфейс та досвід, а також технології та методи, які використовуються для їх створення. В рамках цього дослідження буде проведено аналіз існуючих рішень на ринку каршерінгу, їх переваг та недоліків, а також тенденцій розвитку даної галузі. Особлива увага приділятиметься технологічним аспектам, які дозволяють створювати сучасні, швидкі та безпечні сервіси.

Предметом дослідження є процес розробки сервісу каршерінгу з використанням таких технологій, як Next.js 13, React.js, TypeScript, Tailwind, Prisma та Stripe [2-4]. В рамках цього процесу буде детально розглянуто:

1. Вибір та обґрунтування технологій для реалізації сервісу.
2. Проектування архітектури сервісу, включаючи фронтенд та бекенд.
3. Інтеграцію з платіжною системою Stripe для забезпечення безпечних транзакцій.
4. Використання Tailwind для створення адаптивного та привабливого інтерфейсу користувача.
5. Реалізацію системи управління даними за допомогою Prisma.
6. Тестування, налагодження та оптимізацію продуктивності сервісу.

Цей підхід дозволить створити зручний сервіс каршерінгу.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. ПОСТАНОВКА ТА АНАЛІЗ ЗАДАЧІ

1.1. Постановка задачі

Сервіс каршерінгу – це платформа, яка дозволяє користувачам орендувати автомобілі на короткі періоди через браузер. Сервіс каршерінгу працює через Інтернет та використовує інтерфейс браузера, розміщуючи всю необхідну інформацію на віддалених серверах.

Використання сервісу каршерінгу надає користувачам можливість швидко знаходити та бронювати автомобілі, роблячи процес оренди більш інтерактивним, зручним і доступним. Оскільки сервіс не потребує встановлення на ПК, користувачі можуть орендувати автомобілі з будь-якого пристрою, в будь-який час і в будь-якому місці. Це забезпечує гнучкість у плануванні поїздок і ефективне використання часу.

Для функціонування сервісу каршерінгу необхідні вебсервер і база даних. Вебсервер відповідає за обробку клієнтських запитів, включаючи приймання запитів на оренду автомобілів, їх обробку та відправку відповідей користувачам. Він забезпечує безперебійну роботу сервісу, обробляючи запити в режимі реального часу, що дозволяє користувачам отримувати актуальну інформацію про наявні автомобілі, тарифи та інші важливі деталі.

База даних є основою для зберігання всієї необхідної інформації, що використовується сервісом каршерінгу. Вона забезпечує ефективне управління даними, дозволяючи зберігати великі обсяги інформації з високою швидкістю та надійністю. У базі даних зберігаються відомості про автомобілі, користувачів, бронювання, тарифи та інші важливі дані.

Сервіс каршерінгу надає можливість багатьом користувачам одночасно отримувати доступ у браузері, що усуває необхідність встановлення сервісу на кожному пристрої. Це значно спрощує процес оновлення та обслуговування сервісу, оскільки всі зміни проводяться на серверній стороні і одразу стають доступними для всіх користувачів.

Сервіс спрямований на надання користувачам можливості орендувати автомобілі на короткий період, що дозволяє економити на витратах, пов'язаних з

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		9

володінням власним автомобілем. Крім того, сервіс забезпечує доступність транспортних засобів у будь-який час, що підвищує гнучкість і зручність пересування для користувачів.

Впровадження сервісу каршерінгу забезпечить користувачам простоту бронювання автомобілів через браузер, надаючи доступ до різноманітних моделей транспортних засобів. Завдяки інтерактивному інтерфейсу, користувачі зможуть швидко знаходити і бронювати доступні автомобілі, що зробить процес оренди легким та зручним.

1.2. Аналіз існуючих аналогів

Сучасний сегмент сервісів каршерінгу приваблює все більше користувачів у великих містах по всьому світу завдяки своїй високій доступності й зручності. Проведений аналіз існуючих платформ розкриває ключові напрямки й особливості їхнього розвитку. Розглянемо кілька популярних сервісів каршерінгу, щоб краще зрозуміти їх функціональні можливості та переваги, які вони пропонують сучасним користувачам.

Getmancar - <https://getmancar.com.ua>

Getmancar (рис. 1.1) є сервісом каршерінгу, який пропонує зручне та ефективне рішення для оренди автомобілів у різних містах України.

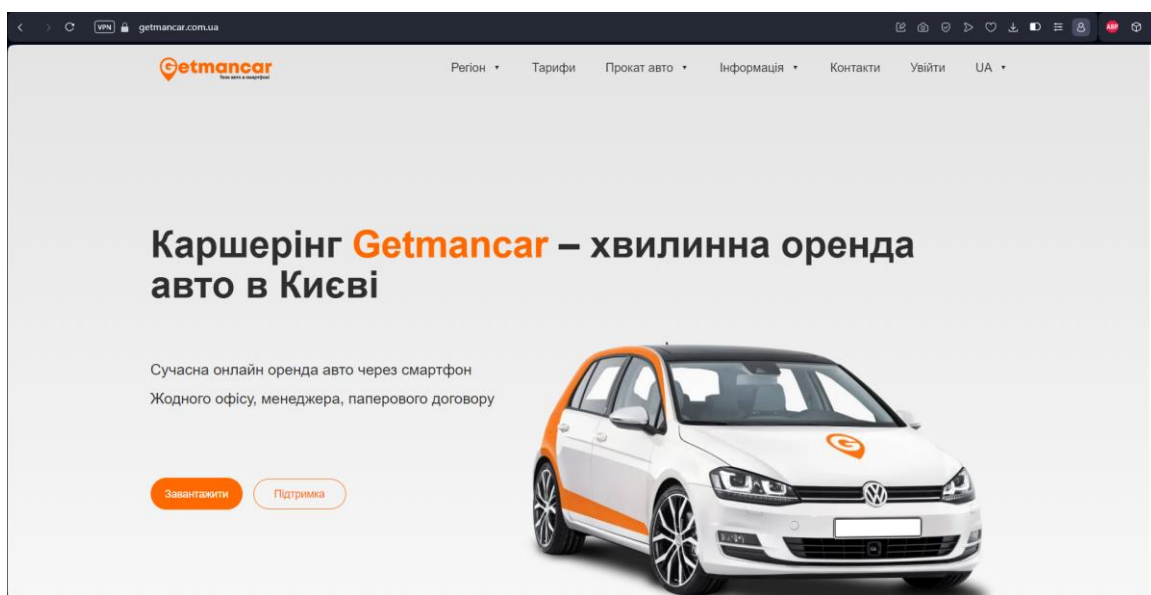


Рисунок 1.1 – Головна сторінка сервісу Getmancar

У сервісі Getmancar користувачі можуть швидко здійснити бронювання автомобіля, вибрати підходящий клас авто від економічного до преміального,

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

оцінити вартість і тривалість оренди (рис. 1.2). Сервіс також надає можливість вибору додаткових опцій, таких як страхування, GPS-навігація та інші сервіси, що полегшують використання автомобіля.

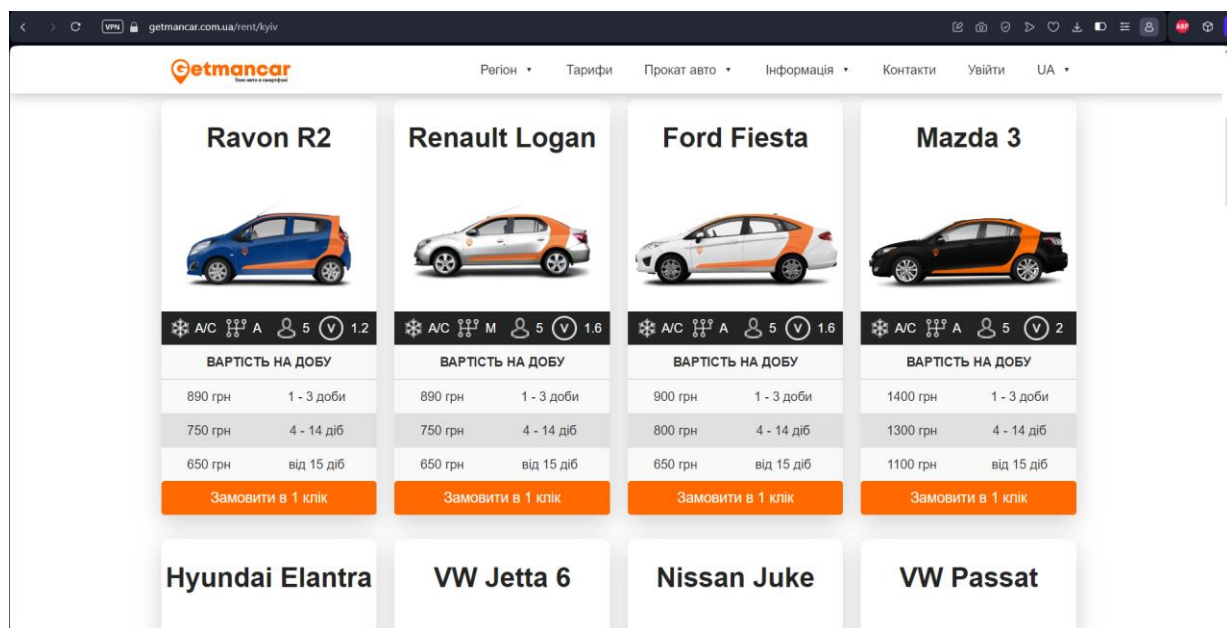


Рисунок 1.2 – Вибір автомобіля на Getmancar

Сервіс каршерінгу Getmancar пропонує інтерфейс, який легко навігується і адаптований для різних типів користувачів, включаючи як частних клієнтів, так і підприємства. Кожен автомобіль на сайті супроводжується детальною інформацією про його технічний стан, фотографіями та умовами оренди, що дозволяє користувачам зробити обміркований вибір.

NarsCars - <https://narscars.com.ua/ua>

NarsCars (рис. 1.3) - це український сервіс каршерінгу, що набув значної популярності завдяки своєму фокусу на забезпеченні зручних та доступних послуг оренди автомобілів.

У сервісі NarsCars користувачі можуть з легкістю та швидкістю орендувати автомобіль, який відповідає їхнім індивідуальним вимогам і бюджету (рис. 1.4). Платформа пропонує різноманітні класи авто - від економічних до преміальних, що задовольняє різноманітні потреби користувачів. Інформація про кожен автомобіль на сайті включає його технічний стан, вартість оренди та умови використання, що спрощує процес вибору для клієнтів.

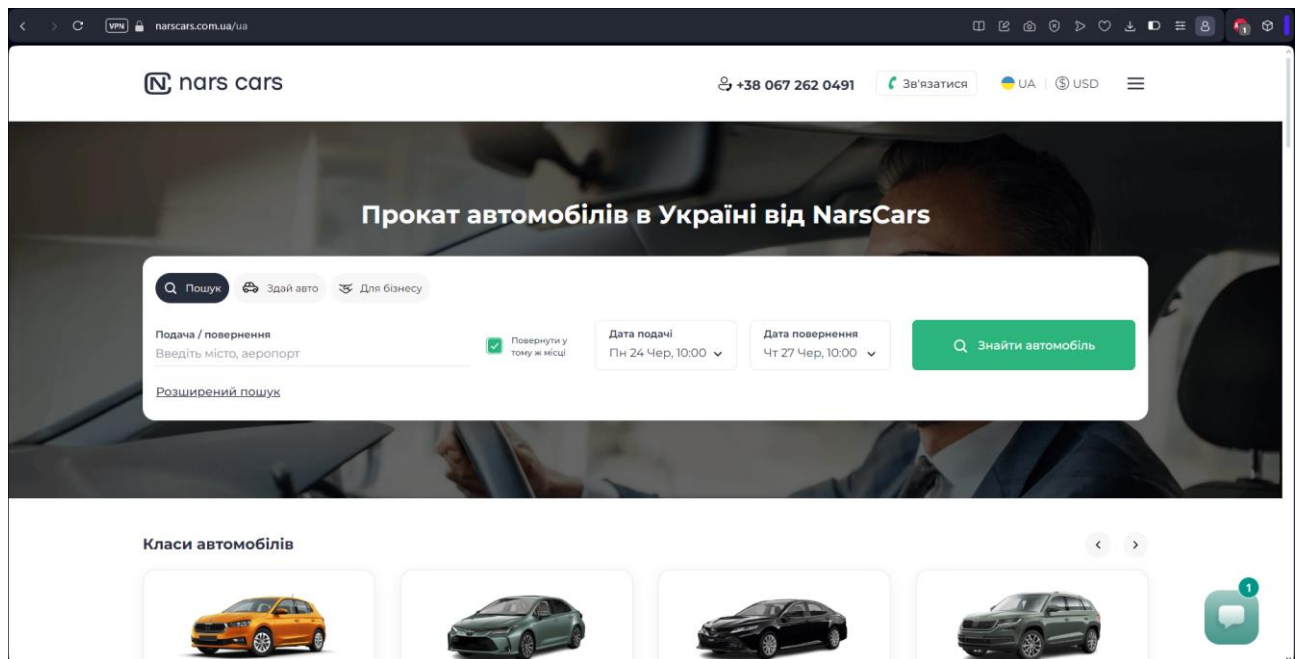


Рисунок 1.3 – Головна сторінка сервісу NarsCars

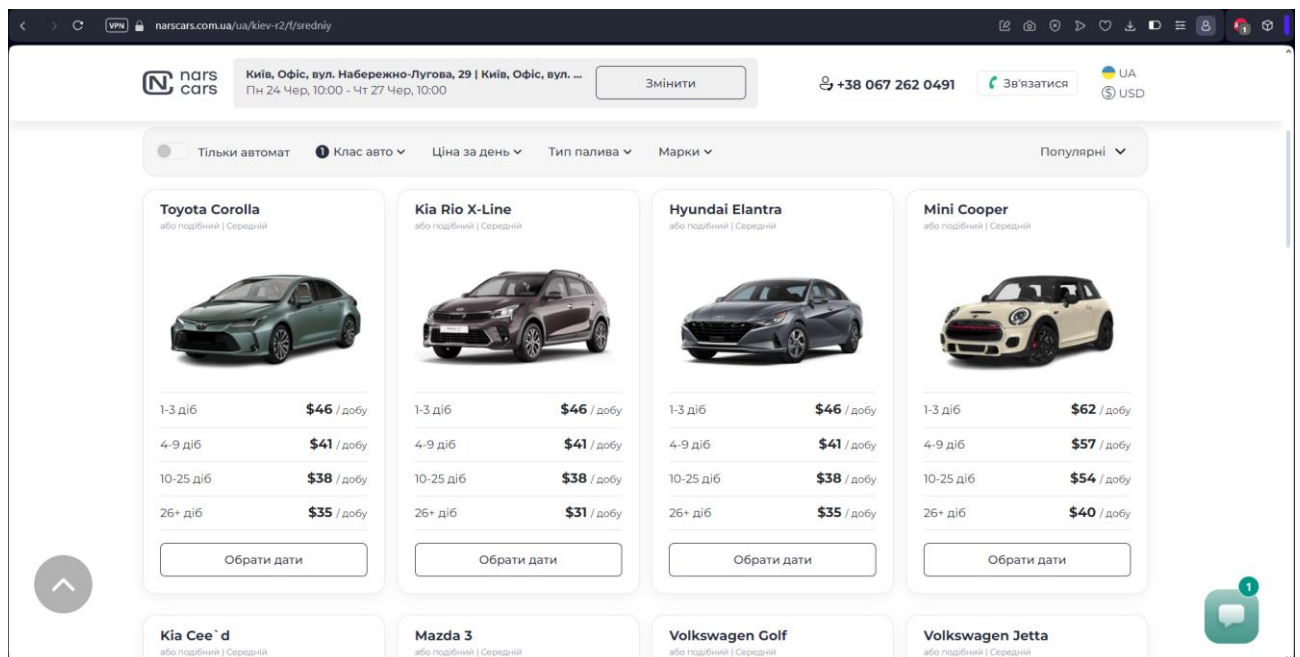


Рисунок 1.4 – Вибір автомобіля на NarsCars

Zipcar - <https://www.zipcar.com>

Zipcar (рис. 1.5) - це міжнародний сервіс каршерінгу, який пропонує можливість користуватися автомобілями на короткострокові терміни. Основний акцент платформи полягає на наданні доступу до автомобілів без необхідності власності або повноцінної оренди.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

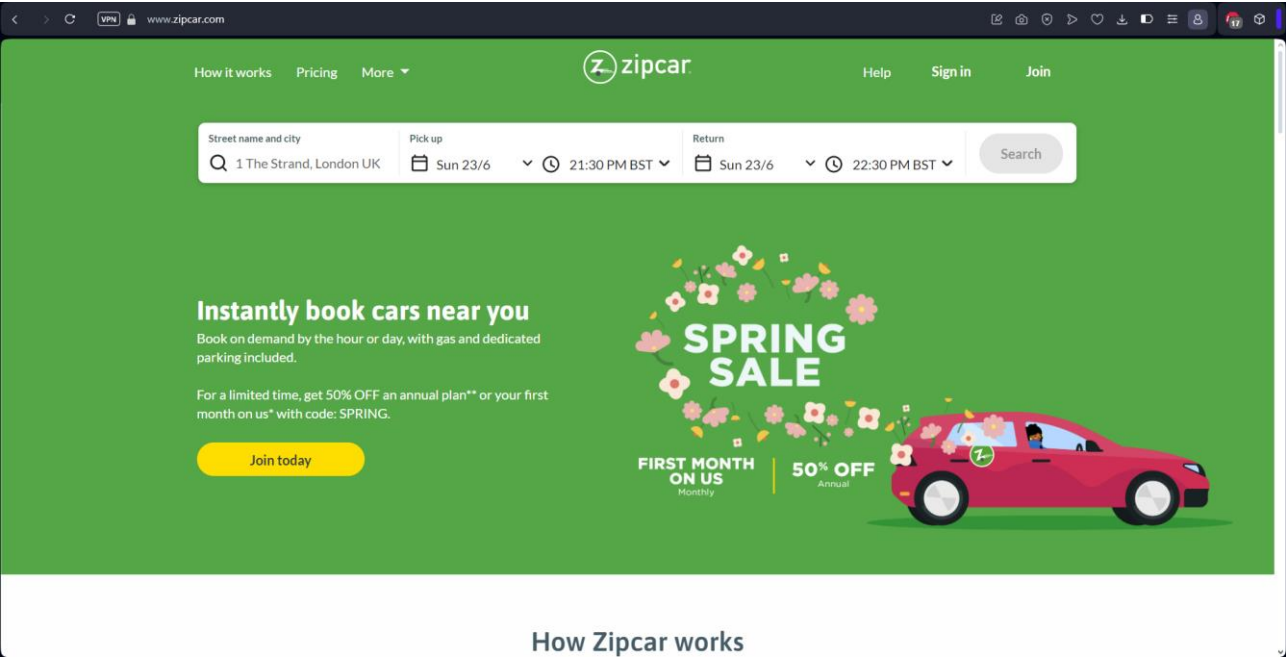


Рисунок 1.5 – Головна сторінка сервісу Zipcar

Zipcar відомий своєю гнучкістю і зручністю: він надає можливість використовувати автомобілі для кількох годин або навіть для цілого дня, що робить його ідеальним вибором для короткострокових поїздки (рис. 1.6).

Сервіс пропонує широкий вибір автомобілів різних класів та моделей, що включає економічні авто, кросовери та електричні транспортні засоби. Кожен автомобіль на сайті супроводжується детальною інформацією про його технічний стан і умови використання, що дозволяє клієнтам зробити інформований вибір.

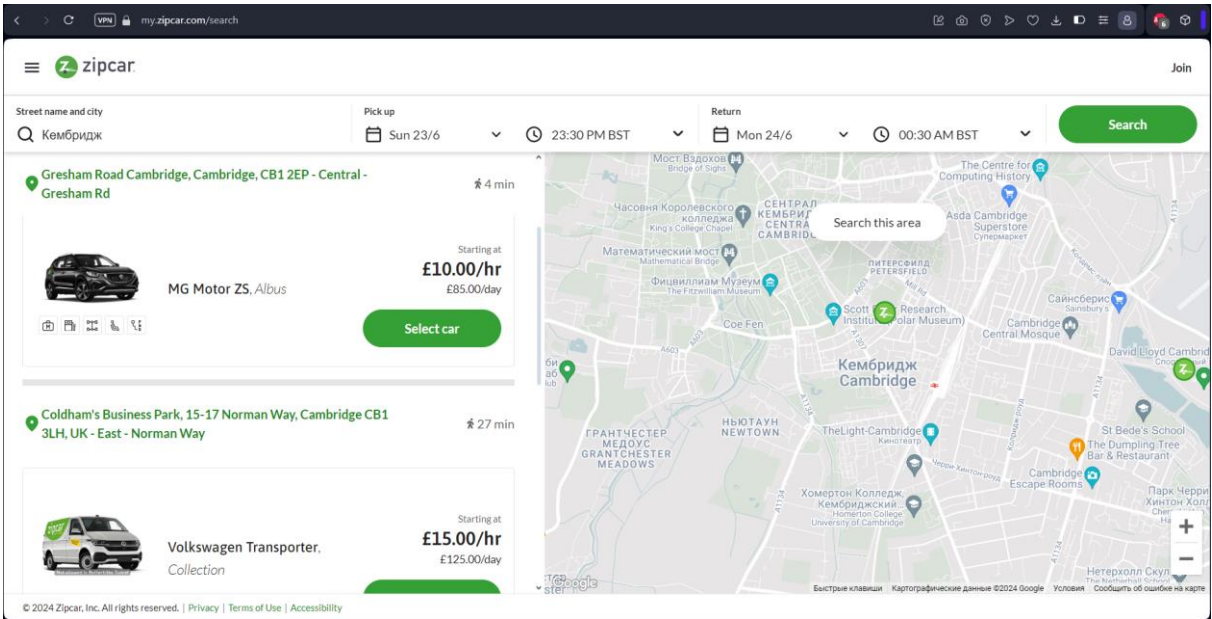


Рисунок 1.6 – Вибір автомобіля на Zipcar

Ці три сервіси каршерінгу - Zipcar, NarsCars і Getmancar - кожен відзначається своїми унікальними особливостями і пропозиціями для користувачів.

Zipcar, міжнародно відомий сервіс, спеціалізується на короткостроковій оренді автомобілів у містах, пропонуючи широкий вибір моделей, включаючи електричні транспортні засоби. NarsCars і Getmancar, вітчизняні сервіси, орієнтовані на український ринок, пропонують різноманіття автомобілів від економ до преміум класів. Обидва сервіси відрізняються зручністю онлайн-бронювання та детальним описом кожного автомобіля.

Таблиця 1.1

Порівняння існуючих аналогів

Критерії	Getmancar	NarsCars	Zipcar
Типи автомобілів	Електричні, стандартні, преміум	Електричні, стандартні, преміум	Електричні, стандартні, преміум
Тривалість оренди	Короткострокова (дні)	Короткострокова (дні)	Короткострокова (години, дні)
Географічне покриття	Україна	Україна	Міжнародне
Додаткові послуги	Страховання, GPS, інші опції	Страховання, додаткові опції	GPS, страховання, паливо включено
Особливості	Локальний сервіс, вигідні умови оренди	Український ринок, локальна підтримка	Гнучкі умови оренди, великий вибір моделей

Було вирішено розробити власний сервіс каршерінгу, з увагою на ключові аспекти, що забезпечать зручність користувачам:

- необхідно створити зручний і простий інтерфейс, який дозволить користувачам легко орендувати автомобілі та швидко знаходити потрібні опції;
- забезпечити чіткість, послідовність та повноту процесу оренди автомобілів, де кожен крок буде чітко структурований і логічно поєднаний з наступним, щоб забезпечити зручність користувачам у бронюванні та доступі до необхідної інформації.

Таким чином, сервіс каршерінгу буде спрямований на максимальне спрощення процесу оренди та надання користувачам зручного і ефективного інструменту для оренди автомобілів.

1.3. Обґрунтування вибору інструментальних засобів

Під час вибору технологій для розробки сервісу каршерінгу були ураховані ключові технічні, функціональні та естетичні аспекти. Вибір кожного інструменту був обґрунтований з урахуванням високих стандартів якості, ефективності та надійності сервісу. Кожна технологія відіграє ключову роль у забезпеченні оптимальної продуктивності, безпеки та зручності для користувачів сервісу каршерінгу.

Таблиця 1.2

Технологія	Обґрунтування вибору
React.js + Next.js	Використання React.js разом з Next.js забезпечує високу продуктивність і швидкодію завдяки підтримці серверного та статичного відтворення сторінок. Це особливо важливо для створення динамічних інтерфейсів та ефективної маршрутизації, що дозволяє користувачам легко взаємодіяти з сервісом.
TypeScript	TypeScript був обраний для додавання строгої типізації до коду, що підвищує його надійність та спрощує управління сервісом, особливо у великих масштабах. Це робить

	розробку більш прогнозованою і допомагає уникнути помилок на етапі виконання.
Tailwind CSS	Tailwind CSS використовується для швидкого створення і стилізації компонентів інтерфейсу. Його готові утиліти і компоненти дозволяють ефективно працювати з дизайном без необхідності написання власних CSS-стилів.
Prisma	Prisma обрано для автоматизації рутинних операцій з базою даних, таких як створення, читання, оновлення та видалення записів. Його зручний інтерфейс і гнучкість управління дозволяють підтримувати високий рівень надійності та швидкість обробки даних.
Stripe	Для прийому платежів обрано Stripe, який відомий своєю безпекою та зручністю інтеграції з різноманітними платіжними методами. Це забезпечує користувачам безпечний і зручний процес оплати за оренду автомобілів.
Firebase	Firebase використовується для забезпечення серверної інфраструктури а саме збереження фотографій авто у хмарному сховищі. Це дозволяє легко інтегрувати рішення для реального часу [5].
MongoDB	MongoDB як гнучка NoSQL БД для зберігання та обробки структурованих даних про автомобілі, користувачів та операції. Вона забезпечує швидкий доступ до даних і зручність у розширенні масштабу сервісу.

Ці інструменти були обрані з метою забезпечення високої якості, продуктивності та надійності сервісу каршерінгу. Вони сприяють ефективній розробці, підтримці і поліпшенню користувацького досвіду, роблячи процес оренди автомобілів зручним і безпечним для всіх користувачів.

1.4. Технічне завдання на розробку сервісу

- Назва: Сервіс каршерінгу.

Повне найменування: Сервіс каршерінгу.

Коротке найменування: Сервіс.

- Призначення:

Сервіс розроблений для забезпечення зручного та ефективного процесу оренди автомобілів користувачами. Основна мета полягає в створенні платформи, що дозволяє швидко і просто здійснювати оренду автомобілів на визначений термін. Адміністративна панель надає можливість управління автомобілями, обробку бронювань і моніторингу статистики використання сервісу.

- Вимоги до сервісу:

Сервіс має забезпечувати можливість реєстрації та аутентифікації користувачів. Користувачам необхідно мати можливість зареєструватися на платформі та виконувати вхід до сервісу.

Пошук і бронювання автомобілів повинні бути простими і зручними, з можливістю отримання детальної інформації про автомобілі, включаючи опис та фотографії. Користувачам також має бути доступна можливість обрати термін оренди та здійснити оплату за послуги.

Адміністративні функції сервісу повинні включати можливість адміністраторам додавати, редагувати та видаляти автомобілі, а також переглядати та керувати бронюваннями користувачів. Крім того, адміністратори повинні мати можливість моніторити статистику використання сервісу.

Сервіс повинен інтегрувати безпечні методи оплати для здійснення фінансових транзакцій з використанням захисту особистих даних користувачів та фінансової інформації. Надійність сервісу повинна гарантувати її безперебійну доступність 24/7 з мінімальними простоями.

- Вимоги до інтерфейсу:

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

Інтерфейс сервісу має бути легким у сприйнятті і забезпечувати зручну навігацію для різних категорій користувачів. Користувачам повинно бути просто знаходити інформацію про наявні автомобілі, включаючи їх характеристики та фотографії. Адміністраторський інтерфейс має надавати зручні інструменти для управління автомобілями та бронюваннями, включаючи можливість перегляду та зміни інформації.

Висновки до першого розділу

У першому розділі було сформульовано завдання сервісу, визначено основні вимоги до сервісу каршерінгу та проведено аналіз існуючих технологій для їх подальшого використання. Проведено детальний аналіз технічного завдання для розробки сервісу каршерінгу

Результати цього дослідження дозволили обрати HTML, CSS, JS, React.js, TypeScript, Tailwind CSS, Prisma, Firebase, MongoDB і Stripe як основні технології для розробки сервісу [6-8]. Цей вибір був зроблений з урахуванням їхніх переваг і можливостей, які забезпечать високу якість, ефективність та надійність сервісу каршерінгу. Такий підхід дозволить створити платформу, що відповідає сучасним вимогам користувачів і забезпечить комфортний і безпечний досвід користувачів під час оренди автомобілів.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА СЕРВІСУ

2.1. Аналіз вимог до сервісу

Сервіс каршерінгу має відповідати вимогам, що забезпечують зручність, ефективність і надійність для користувачів.

Вимоги користувачів вимагають можливості швидкої реєстрації та аутентифікації, що забезпечує безпеку та зручність у використанні сервісу. Основною вимогою є простота інтерфейсу для пошуку і бронювання автомобілів, з можливістю швидкого пошуку необхідної інформації про транспортні засоби, включаючи їх характеристики і фотографії. Користувачі мають мати зручну можливість здійснювати бронювання авто та проведення оплати, що забезпечує їхню задоволеність та комфорт в процесі використання сервісу.

Бізнес-вимоги включають, що сервіс каршерінгу повинен забезпечувати ефективне управління автомобілями та оптимізацію процесів бронювання, що сприяє збільшенню оборотів автомобілів та підвищенню доходів. Основною бізнес-вимогою є здатність залучати нових клієнтів завдяки простому і швидкому інтерфейсу, що дозволяє зручно використовувати сервіс.

Функціональні вимоги сервісу включають створення інтуїтивно зрозумілого інтерфейсу користувача, який спрощує навігацію та забезпечує зручність взаємодії. Для частини адміністратора важливо мати зручні інструменти для управління автомобілями, бронюваннями та статистикою, що дозволяє ефективно контролювати роботу сервісу.

Нефункціональні вимоги націлені на забезпечення високої продуктивності сервісу, мінімізацію часу простою і забезпечення безпеки і надійності у вимогах постійної доступності.

Вимоги до сервісу враховують підтримку різних браузерів, що забезпечує можливість використання в різних умовах експлуатації без обмежень. Вимоги для вебсервера включають наявність операційної системи, яка підтримує Node.js, встановлення та налаштування додаткових компонентів, таких як Prisma для роботи з базою даних і інтеграція з Stripe для обробки платежів.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

Діаграма варіантів використання є важливим візуальним зображенням, яке показує, як різні актори, такі як користувачі, взаємодіють з сервісом для досягнення конкретних цілей. Вона визначає функціональні вимоги сервісу з погляду користувача. Шляхом ілюстрації акторів, варіантів використання та їх взаємозв'язків ця діаграма надає загальний огляд поведінки та функціоналу сервісу, що допомагає в аналізі вимог, проектуванні сервісу та валідації взаємодії користувачів [9].

Діаграма варіантів використання для сервісу каршерінгу (рис. 2.1) описує ролі гостя, авторизованого користувача і адміністратора, кожна з яких має свої унікальні функціональні можливості та доступ до ресурсів сервісу.

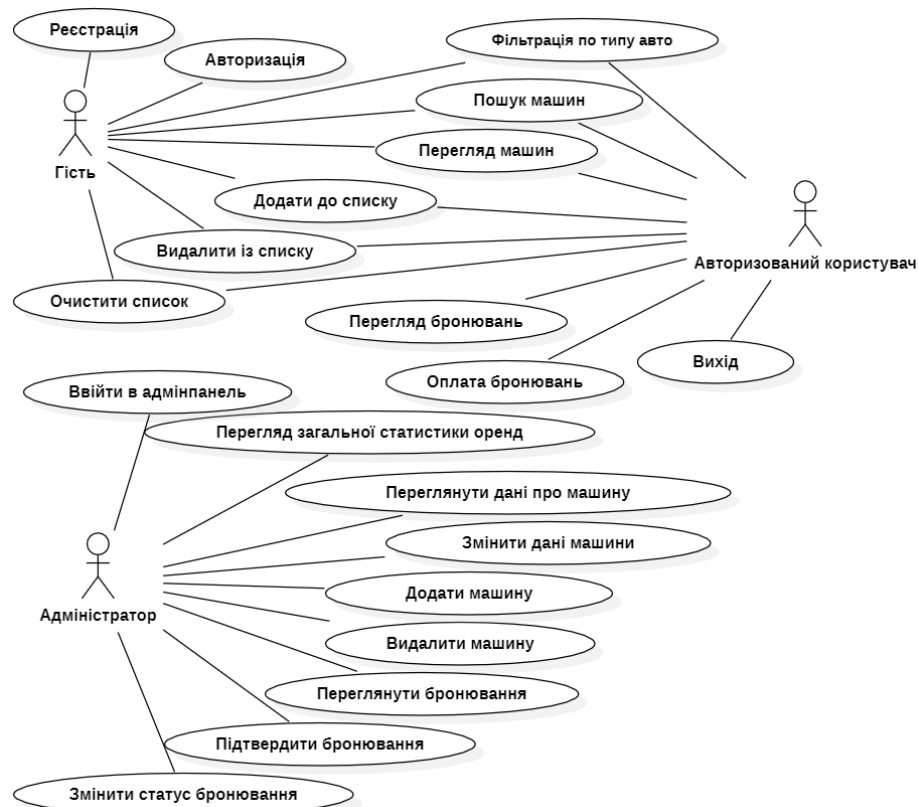


Рисунок 2.1 – Діаграма варіантів використання

Гість має можливість зареєструватись у сервісі або авторизуватись, якщо вже має обліковий запис. Він може фільтрувати доступні автомобілі за їх типом та проводити пошук автомобілів за різними критеріями. Гість може переглядати детальну інформацію про автомобілі, таку як характеристики та фотографії, а також керувати списком бажань, додавати, видаляти автомобілі або очищати весь список.

Авторизований користувач включає всі можливості гостя, а також може переглядати свої бронювання та оплачувати їх через інтеграцію з платіжною системою. Після завершення сеансу роботи користувач може вийти з облікового запису.

Адміністратор має доступ до адміністративної панелі сервісу, де він може переглядати загальну статистику оренд автомобілів і отримувати дані про кожен автомобіль, включаючи характеристики і фотографії. Він володіє повноваженнями для зміни інформації про автомобілі, їх додавання, редагування та видалення. Адміністратор також може переглядати список бронювань користувачів, підтверджувати їх і змінювати статуси.

2.2. Алгоритм роботи та стани сервісу

Діаграма активностей використовується для моделювання та візуалізації динамічних аспектів сервісу. Вона ілюструє потік управління та послідовність дій, необхідних для виконання використання, фокусуючись на умовах потоку та послідовності виконання [10].

Діаграма активностей для процесу пошуку та бронювання автомобіля в сервісі каршерінгу (рис. 2.2) починається з того, що користувач заходить на головну сторінку сервісу.

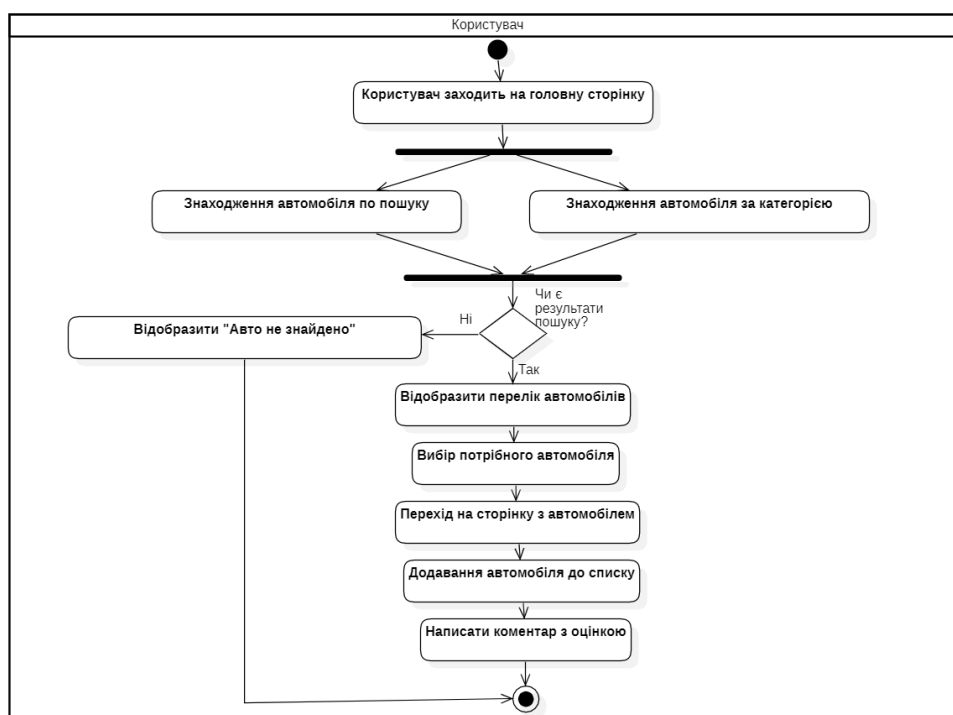


Рисунок 2.2 – Діаграма процесу пошуку автомобіля і бронювання

Він має дві альтернативи: перший варіант - знайти автомобіль за допомогою пошуку, де він вводить назву автомобіля, і другий варіант - знайти автомобіль за категорією, обравши відповідну категорію. Після цього перевіряється наявність результатів пошуку: якщо результатів немає, процес завершується. Якщо ж є результати, відображається перелік доступних автомобілів. Користувач вибирає потрібний автомобіль для бронювання і переходить на сторінку з деталями автомобіля. На цій сторінці він може додати автомобіль до свого списку, а також залишити коментар з оцінкою. Процес завершується після написання коментаря.

Діаграма процесу бронювання та його обробки в сервісі каршерінгу (рис. 2.3) розпочинається з того, що авторизований користувач додає автомобіль до свого списку для подальшої оренди. Після цього він заповнює необхідні дані для бронювання і проводить оплату оренди. Далі адміністратор переглядає всі бронювання і може змінювати їх статуси залежно від поточної ситуації. Після зміни статусу бронювання користувачу автоматично відображаються оновлені дані щодо статусу його бронювання. Процес завершується після того, як користувач отримує відповідні повідомлення про результати обробки його бронювання.

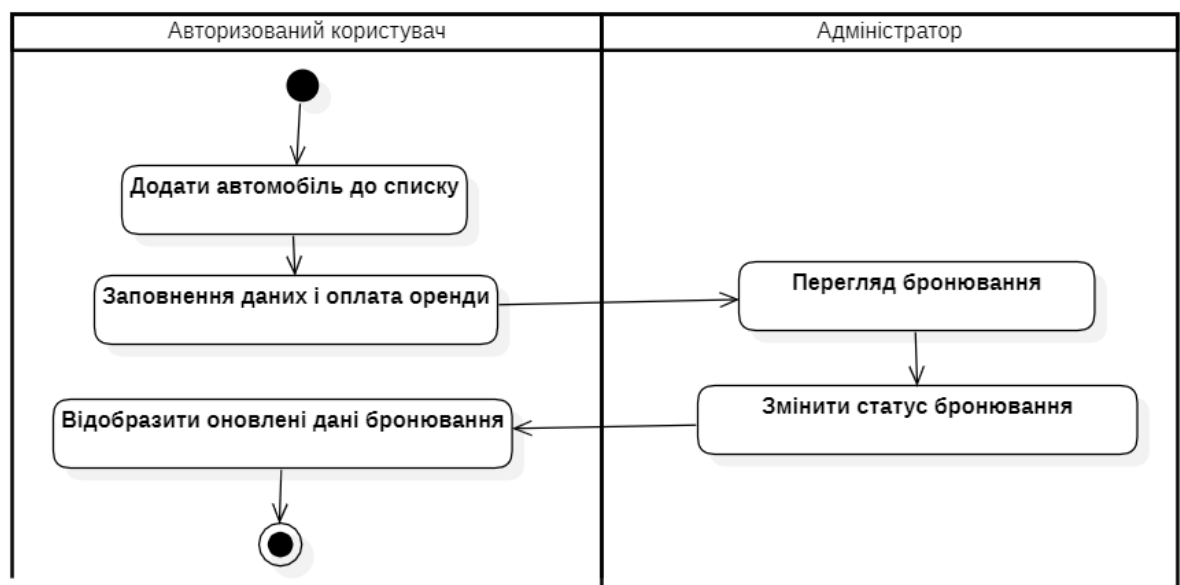


Рисунок 2.3 – Детальна діаграма процесу бронювання і його обробка

2.3. Об'єктно-орієнтована модель сервісу

Об'єктно-орієнтована модель сервісу зображена у вигляді діаграми класів (рис. 2.4). Ця діаграма відображає основні компоненти сервісу та їх взаємозв'язки, дозволяючи візуалізувати структуру та поведінку сервісу.

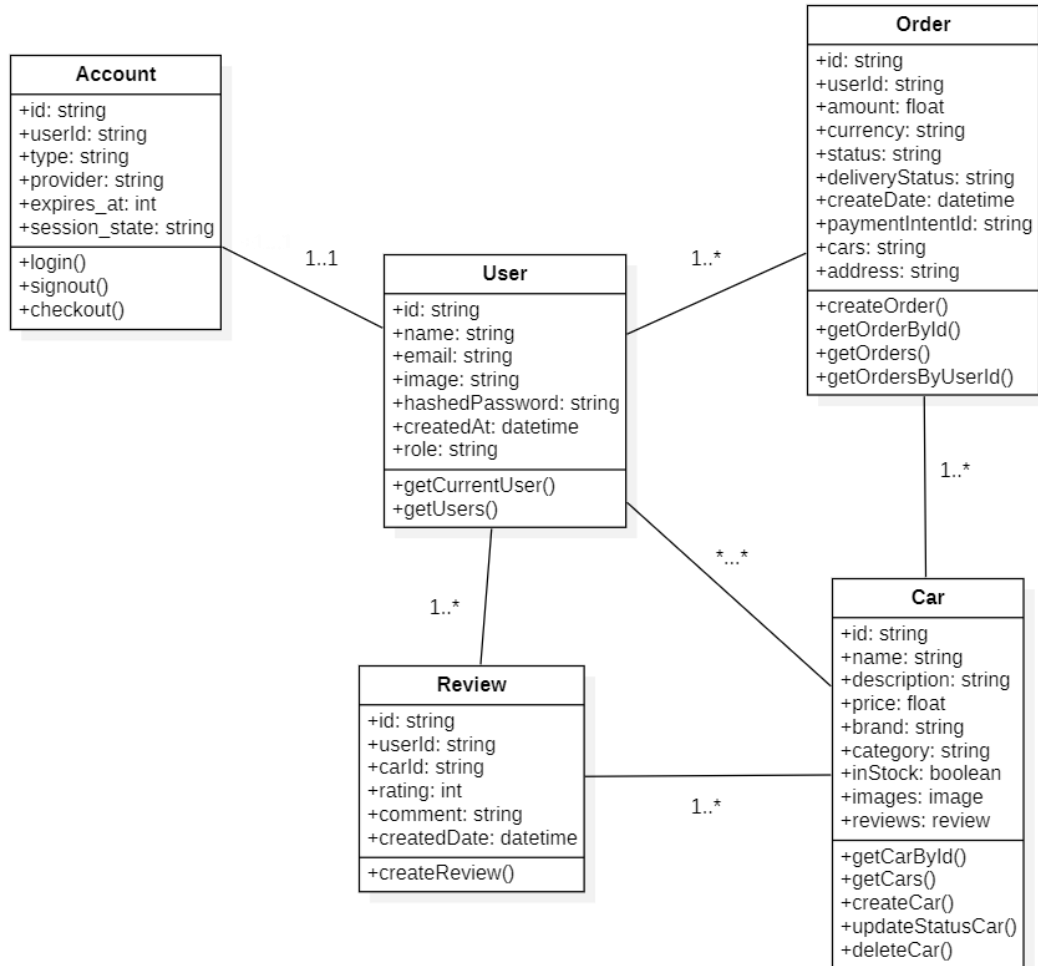


Рисунок 2.4 – Діаграма класів

Діаграма класів є типом UML діаграми, яка використовується для візуального представлення структури та взаємозв'язків класів у сервісі. UML – це стандартизована мова моделювання, яка допомагає в проектуванні та документуванні програмних сервісів. У діаграмах класів класи зображуються у вигляді прямокутників, кожен з яких містить три секції: назва класу, атрибути та методи. Лінії, що з'єднують класи, ілюструють асоціації, показуючи взаємозв'язки. Діаграми класів забезпечують високорівневе уявлення про дизайн сервісу, допомагаючи комунікувати та документувати структуру програмного забезпечення [11].

Класи, такі як Account, User, Car, Review та Order, демонструють об'єкти, їх атрибути і методи, а також зв'язки між ними, що ілюструють, як об'єкти взаємодіють один з одним для виконання різних функцій сервісу.

Діаграма класів для сервісу каршерінгу включає класи User, Account, Order, Car і Review. Кожен з цих класів має свої атрибути та методи, що відповідають різним аспектам сервісу. Клас User має атрибути id, name, email, image, hashedPassword, createdAt і role, а також методи getCurrentUser() і getUsers(). Цей клас зберігає інформацію про користувача, включаючи його ім'я, електронну пошту, зображення профілю та інші дані, а також дозволяє отримувати інформацію про поточного користувача та список всіх користувачів.

Клас Account має атрибути id, userId, type, provider, expires_at і session_state, а також методи login(), signout() і checkout(). Цей клас відповідає за управління обліковими записами користувачів, включаючи аутентифікацію.

Клас Order містить атрибути id, userId, amount, currency, status, deliveryStatus, createDate, paymentIntentId, cars і address, а також методи createOrder(), getOrderById(), getOrders() і getOrdersByUserId(). Цей клас використовується для управління замовленнями користувачів, включаючи створення нових замовлень, отримання замовлень за ідентифікатором та користувачем, а також отримання списку всіх замовлень.

Клас Car включає атрибути id, name, description, price, brand, category, inStock, images і reviews, а також методи getCarById(), getCars(), createCar(), updateStatusCar() і deleteCar(). Цей клас представляє автомобілі, доступні для оренди, і включає методи для отримання інформації про автомобілі, створення нових автомобілів, оновлення їх статусу та видалення.

Клас Review має атрибути id, userId, carId, rating, comment і createDate, а також метод createReview(). Цей клас використовується для управління відгуками користувачів про автомобілі, включаючи створення нових відгуків.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

2.4. Взаємодія об'єктів сервісу

Діаграма кооперації — це тип UML діаграми, що представляє функціональні вимоги сервісу у вигляді взаємодій між користувачами і сервісом для досягнення певної мети. Вона ілюструє сценарії, де сервіс взаємодіє із зовнішніми об'єктами, що дозволяє визначити, як сервіс повинен працювати з точки зору користувача. Основні компоненти діаграми варіантів використання включають акторів, варіанти використання, а також асоціації між ними [12].

Діаграма кооперації у випадку сервісу бронювання автомобілів (рис. 2.5) демонструє взаємодію між користувачем та сервісом для виконання різних завдань. Користувач може шукати автомобіль, після цього сервіс відображає список доступних автомобілів. Користувач може переглядати деталі вибраного автомобіля, такі як технічні характеристики, фотографії та доступність. Якщо автомобіль підходить, користувач заповнює необхідні дані для бронювання, включаючи особисту інформацію, дату та час оренди. Далі користувач здійснює бронювання, і сервіс опрацьовує його.



Рисунок 2.5 – Діаграма кооперації сервісу каршерінгу

Після завершення процесу оренди користувач має можливість залишити коментар та оцінку до автомобіля. Користувач взаємодіє з інтерфейсом сервісу каршерінгу та переходить на сторінку авто, яке він орендував раніше. Унизу сторінки після успішної оренди користувач бачить додаткове поле для написання коментаря про автомобіль. Користувач вводить коментар з усіма деталями які він бажає написати і ставить рейтинг автомобілю - від 1 до 5 зірочок. Далі браузер відправляє запит на сервер для збереження коментаря. Вебсервер отримує запит та обробляє його. Якщо коментар проходить валідацію, то вебсервер взаємодіє з базою даних і коментар записується у таблицю відгуків. Після цього вебсервер відправляє повідомлення у браузер про успішне збереження коментаря і користувач бачить його у верхній частині сторінки.

Таким чином користувач бачить, що його коментар був успішно збережений та може бути впевнений, що інші користувачі сервісу побачать його.

У іншому випадку, якщо коментар не пройшов валідацію, то вебсервер генерує помилку і відправляє повідомлення про неможливість збереження коментаря. Вся ця послідовність показана на діаграмі послідовності відправлення коментаря (рис. 2.6).

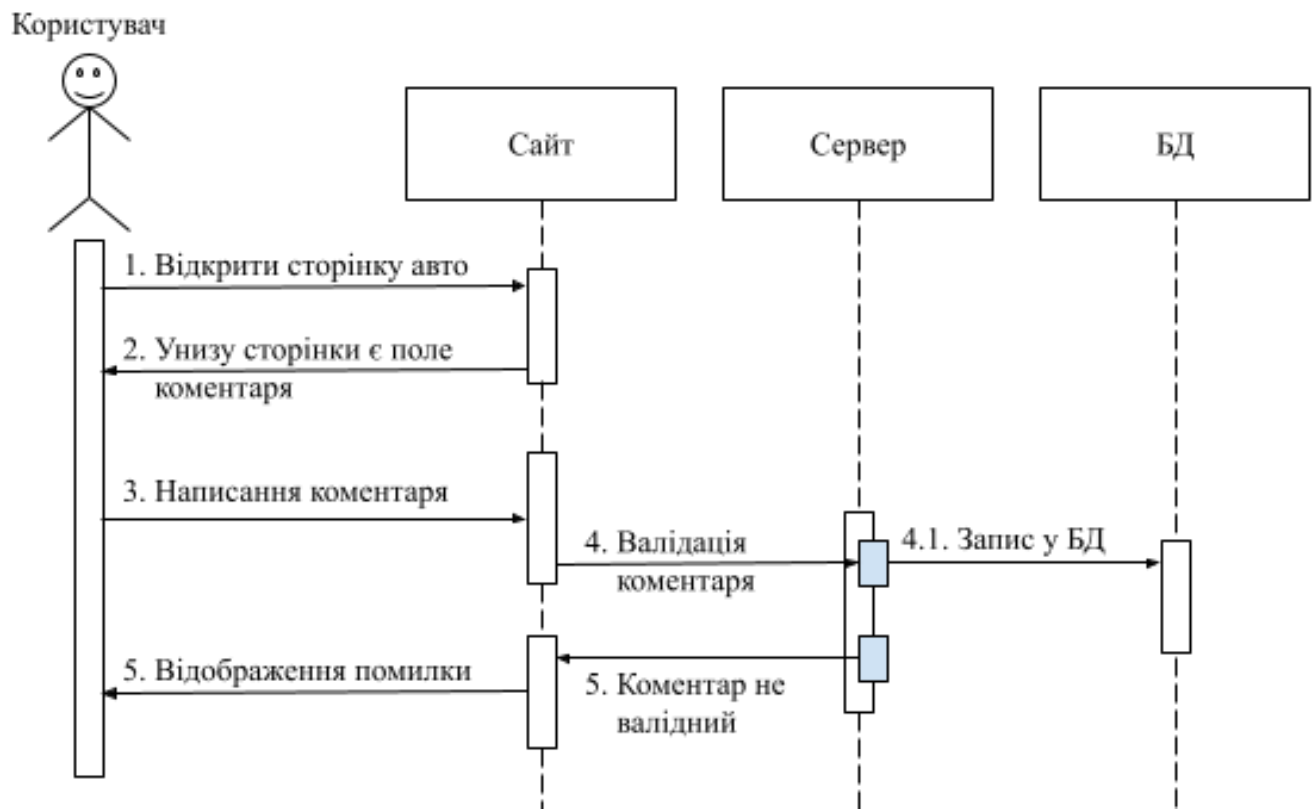


Рисунок 2.6 – Діаграма послідовності відправлення коментаря

2.5. Розробка бази даних сервісу

База даних сервісу каршерінгу реалізована за допомогою СУБД MongoDB та Firebase, що забезпечує надійне зберігання даних та ефективне управління ними. MongoDB, як документно-орієнтована БД NoSQL, надає гнучкість у зберіганні різноманітних даних, включаючи інформацію про користувачів, автомобілі, замовлення та відгуки. Вона дозволяє зберігати дані у форматі JSON-подібних документів, що спрощує процес зберігання складних даних та їх швидкий доступ.

Таблиця «Car» містить інформацію про автомобілі.

Таблиця 2.1

Опис полів таблиці «Car»

Назва	Тип даних	ПК	ЗК	Опис поля
id	string	+	–	Унікальний ідентифікатор автомобіля
name	string	–	–	Назва автомобіля
description	string	–	–	Опис автомобіля
price	float	–	–	Вартість оренди автомобіля
brand	string	–	–	Марка автомобіля
category	string	–	–	Категорія автомобіля
inStock	boolean	–	–	Категорія автомобіля
images	array	–	–	Зображення автомобіля
reviews	array	–	+	Відгуки про автомобіль

Таблиця «User» зберігає інформацію про користувачів сервісу каршерінгу.

Таблиця 2.2

Опис полів таблиці «User»

Назва	Тип даних	ПК	ЗК	Опис поля
id	string	+	–	Унікальний ідентифікатор користувача
name	string	–	–	Ім'я користувача
email	string	–	–	Електронна адреса користувача
image	array	–	–	Зображення профілю користувача
hashedPassword	string	–	–	Пароль користувача
createdAt	datetime	–	–	Дата і час створення акаунта
role	string	–	+	Роль користувача у сервісі

Таблиця «Order» зберігає інформацію про замовлення, здійснені користувачами сервісу каршерінгу.

Таблиця 2.3

Опис полів таблиці «Order»

Назва	Тип даних	ПК	ЗК	Опис поля
id	string	+	–	Унікальний ідентифікатор замовлення
userId	string	–	–	Ідентифікатор користувача, що замовив
amount	float	–	–	Сума замовлення
currency	string	–	–	Валюта замовлення
status	string	–	–	Статус замовлення
deliveryStatus	string	–	–	Статус доставки замовлення
createDate	datetime	–	–	Дата і час створення замовлення
payment	string	–	–	Ідентифікатор платежу
cars	array	–	+	Масив автомобілів, включених у замовлення
address	string	–	–	Адреса доставки замовлення

Таблиця «Account» зберігає інформацію про облікові записи користувачів, включаючи деталі аутентифікації.

Таблиця 2.4

Опис полів таблиці «Account»

Назва	Тип даних	ПК	ЗК	Опис поля
id	string	+	–	Унікальний ідентифікатор облікового запису
userId	string	–	–	Ідентифікатор користувача, до якого належить обліковий запис
type	string	–	–	Тип облікового запису
provider	string	–	–	Провайдер аутентифікації
expires_at	int	–	–	Час, коли закінчується сесія
session_state	string	–	–	Стан сесії

Таблиця «Review» відповідає за зберігання відгуків користувачів про автомобілі.

Таблиця 2.5

Опис полів таблиці «Review»

Назва	Тип даних	ПК	ЗК	Опис поля
id	string	+	–	Унікальний ідентифікатор відгуку
userId	string	–	+	Ідентифікатор користувача, який залишив відгук
carId	string	–	+	Ідентифікатор автомобіля, до якого відноситься відгук
rating	int	–	–	Оцінка автомобіля
comment	string	–	–	Текстовий коментар від користувача
createDate	datetime	–	–	Дата створення відгуку

Висновки до другого розділу

У другому розділі кваліфікаційної роботи проаналізовано функціональні і нефункціональні вимоги до сервісу. Сформовано алгоритм роботи сервісу та досліджено можливі її стани.

Була розроблена об'єктно-орієнтована модель сервісу, в якій детально описано основні об'єкти та їх взаємодію. Взаємодія між компонентами сервісу ілюструється за допомогою UML діаграм, які відображають послідовність та кооперацію під час виконання основних процесів. Також була розроблена база даних сервісу, було визначено структуру ключових таблиць та їх атрибути.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СЕРВІСОМ

3.1. Порядок встановлення та налаштування сервісу

Сервіс каршерінгу має багаторівневу архітектуру (рис. 3.1), яка забезпечує надійність, ефективність і зручність у використанні.

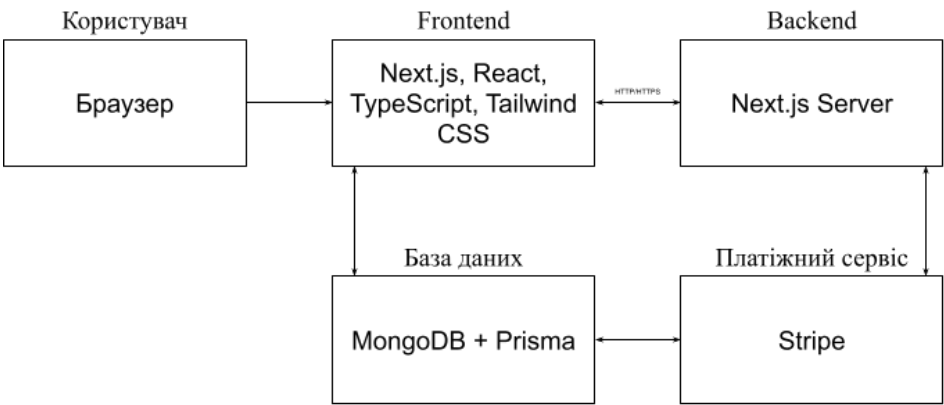


Рисунок 3.1 – Архітектура сервісу каршерінгу

Користувач взаємодіє із сервісом через браузер, де представлений інтуїтивно зрозумілий і адаптивний інтерфейс. Frontend частина сервісу реалізована за допомогою Next.js, React, TypeScript і Tailwind CSS, що забезпечує швидку та інтерактивну взаємодію з користувачем [13-16].

Backend частина сервісу побудована на Next.js, де серверна логіка обробляє запити від користувачів, виконує бізнес-логіку та забезпечує безперебійну роботу сервісу. Для управління платіжними операціями інтегровано платіжний сервіс Stripe, що дозволяє здійснювати безпечні та швидкі транзакції [17].

Дані користувачів і замовлень зберігаються в базі даних, реалізованій на MongoDB з використанням Prisma для зручного управління даними та забезпечення високої продуктивності. Така комбінація технологій дозволяє створити масштабований і надійний сервіс, який відповідає вимогам до якості і безпеки [18-19].

Файлова структура сервісу каршерінгу (рис. 3.2) організована для забезпечення легкої навігації та зручного доступу до різних частин коду.

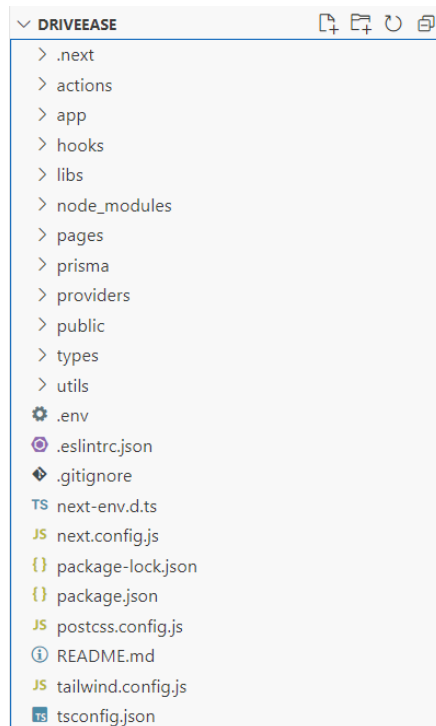


Рисунок 3.2 – Файлова структура сервісу

Папка actions містить функції або дії, які можуть бути викликані з різних частин сервісу, зокрема для виконання бізнес-логіки та взаємодії з бекендом.

У папці app розташовані компоненти, що стосуються структури та навігації сервісу, включаючи глобальні компоненти, макети сторінок та основні налаштування. Папка hooks призначена для користувацьких хуків React, які допомагають повторно використовувати логіку компонентів у різних частинах сервісу. Папка libs зберігає загальні бібліотеки та утиліти, які можуть бути використані в різних частинах сервісу, наприклад, функції для обробки даних або інтеграції зі сторонніми сервісами.

Папка pages містить всі сторінки Next.js, де кожен файл відповідає окремому маршруту у сервісі. У папці prisma зберігаються файли, що стосуються Prisma, включаючи схеми баз даних.

Папка providers містить файли, які надають контекст для сервісу через React Context API. Папка public містить статичні файли, які можуть бути доступні напряму з сервісу, наприклад, зображення. Папка types зберігає TypeScript типи, інтерфейси та типові декларації для забезпечення типізації в проєкті. Папка utils призначена для утилітарних функцій, які допомагають з різними аспектами

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

розробки, такими як обробка рядків, дат, формату даних. Файл .env зберігає конфіденційні змінні середовища, такі як API ключі, параметри підключення до бази даних та інші налаштування.

3.2. Розробка основних функцій сервісу

На рисунках 3.3 та 3.4 зображено компонент «LoginForm», який відповідає за реалізацію форми входу користувача на сайт за допомогою бібліотек Next.js та React. Компонент приймає проп «currentUser», який визначає, чи користувач вже авторизований. Якщо «currentUser» існує, користувач перенаправляється на сторінку «cart».

```
15 interface LoginFormProps {
16   currentUser: SafeUser | null;
17 }
18
19 const LoginForm: React.FC<LoginFormProps> = ({ currentUser }) => {
20   const [isLoading, setIsLoading] = useState(false);
21   const {
22     register,
23     handleSubmit,
24     formState: { errors },
25   } = useForm<FieldValues>({
26     defaultValues: {
27       email: "",
28       password: "",
29     },
30   });
31
32   const router = useRouter();
33
34   useEffect(() => {
35     if (currentUser) {
36       router.push("/cart");
37       router.refresh();
38     }
39   }, []);
40
41   const onSubmit: SubmitHandler<FieldValues> = (data) => {
42     setIsLoading(true);
43     signIn("credentials", {
44       ...data,
45       redirect: false,
46     }).then((callback) => {
47       setIsLoading(false);
48
49       if (callback?.ok) {
50         router.push("/cart");
51         router.refresh();
52         toast.success("Успішний вхід");
53       }
54
55       if (callback?.error) {
56         toast.error(callback.error);
57       }
58     });
59   };
60 }
```

Рисунок 3.3 – Компонент «LoginForm»

Функція «LoginForm» починається з імпорту необхідних залежностей, таких як React хуки, компоненти для побудови форми, функції для роботи з формами з «react-hook-form», функції для аутентифікації з «next-auth/react», і бібліотека для сповіщень «react-hot-toast».

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		32

```

61   if (currentUser) {
62     return <p className="text-center">Успішний вхід. Перенаправлення...</p>;
63   }
64
65   return (
66     <>
67       <Heading title="Увійти в DriveEase" />
68       <Button
69         outline
70         label="Продовжити з Google"
71         icon={AiOutlineGoogle}
72         onClick={() => {
73           signIn("google");
74         }}
75       />
76       <hr className="bg-slate-300 w-full h-px" />
77       <Input
78         id="email"
79         label="Email"
80         disabled={isLoading}
81         register={register}
82         errors={errors}
83         required
84       />
85       <Input
86         id="password"
87         label="Пароль"
88         disabled={isLoading}
89         register={register}
90         errors={errors}
91         required
92         type="password"
93       />
94       <Button
95         label={isLoading ? "Завантаження" : "Увійти"}
96         onClick={handleSubmit(onSubmit)}
97       />
98       <p className="text-sm">
99         Немає облікового запису?{" "}
100         <Link className="underline" href="/register">
101           Зареєструватися
102         </Link>
103       </p>
104     </>
105   );
106 };
107
108 export default LoginForm;

```

Рисунок 3.4 – Продовження компоненту «LoginForm»

Створюється стан «isLoading» для відстеження процесу завантаження під час обробки форми. Використовується «useForm» для керування станом форми, валідацією та обробкою помилок, з початковими значеннями полів «email» та «password».

Виконується побічний ефект за допомогою «useEffect», який перевіряє, чи користувач авторизований. Якщо так, він перенаправляється на сторінку «cart».

Функція «onSubmit» відповідає за обробку подання форми. Після відправки форми, встановлюється стан завантаження, і викликається функція

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

«signIn» для аутентифікації користувача. Якщо аутентифікація успішна, користувач перенаправляється на сторінку «cart» та відображається сповіщення про успішний вхід. У разі помилки, відображається відповідне повідомлення про помилку.

Якщо користувач вже авторизований, відображається повідомлення про успішний вхід та перенаправлення. Інакше відображається форма входу, яка містить заголовок, кнопку для входу з Google, поля для вводу «email» та «password», кнопку для відправки форми, а також посилання на сторінку реєстрації для нових користувачів.

На рисунках 3.5 та 3.6 зображено компонент «RegisterForm», який призначений для реєстрації нового користувача на сайті за допомогою бібліотек Next.js та React.

Компонент приймає проп «currentUser», який визначає, чи користувач вже авторизований. Якщо «currentUser» існує, користувач перенаправляється на сторінку «cart».

Функція «RegisterForm» починається з імпорту необхідних залежностей, таких як React хуки, компоненти для побудови форми, функції для роботи з формами з «react-hook-form», функції для аутентифікації з «next-auth/react», бібліотека для сповіщень «react-hot-toast», бібліотека для HTTP-запитів «axios» та інші необхідні компоненти.

Створюється стан «isLoading» для відстеження процесу завантаження під час обробки форми. Використовується «useForm» для керування станом форми, валідацією та обробкою помилок, з початковими значеннями полів «name», «email» та «password».

Виконується побічний ефект за допомогою «useEffect», який перевіряє, чи користувач авторизований. Якщо так, він перенаправляється на сторінку «cart».

Функція «onSubmit» відповідає за обробку подання форми. Після відправки форми, встановлюється стан завантаження, і надсилається POST-запит на сервер для створення нового акаунту. Якщо акаунт успішно створено, відображається сповіщення про успішну реєстрацію, і користувач автоматично

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		34

аутентифікується за допомогою «signIn». Якщо аутентифікація успішна, користувач перенаправляється на сторінку «cart» та відображається сповіщення про успішний вхід. У разі помилки, відображається відповідне повідомлення про помилку.

Якщо користувач вже авторизований, відображається повідомлення про успішний вхід та перенаправлення. Інакше відображається форма реєстрації, яка містить заголовок, кнопку для реєстрації через Google, поля для вводу «name», «email» та «password», кнопку для відправки форми, а також посилання на сторінку входу для користувачів, які вже мають обліковий запис.

```

16 interface RegisterFormProps {
17   | currentUser: SafeUser | null;
18 }
19
20 const RegisterForm: React.FC<RegisterFormProps> = ({ currentUser }) => {
21   const [isLoading, setIsLoading] = useState(false);
22   const {
23     register,
24     handleSubmit,
25     formState: { errors },
26   } = useForm<FieldValues>({
27     defaultValues: {
28       name: "",
29       email: "",
30       password: "",
31     },
32   });
33
34   const router = useRouter();
35
36   useEffect(() => {
37     if (currentUser) {
38       router.push("/cart");
39       router.refresh();
40     }
41   }, []);
42
43   const onSubmit: SubmitHandler<FieldValues> = (data) => {
44     setIsLoading(true);
45
46     axios
47       .post("/api/register", data)
48       .then(() => {
49         toast.success("Акаунт створено");
50
51         signIn("credentials", {
52           email: data.email,
53           password: data.password,
54           redirect: false,
55         }).then((callback) => {
56           if (callback?.ok) {
57             router.push("/cart");
58             router.refresh();
59             toast.success("Успішний вхід");
60           }
61
62           if (callback?.error) {
63             toast.error(callback.error);
64           }
65         });
66       })
67       .catch(() => toast.error("Щось пішло не так..."))
68       .finally(() => {
69         setIsLoading(false);
70       });
71   };

```

Рисунок 3.5 – Компонент «RegisterForm»

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

```

73   if (currentUser) {
74     return <p className="text-center">Успішний вхід. Перенаправлення...</p>;
75   }
76
77   return (
78     <>
79       <Heading title="Реєстрація в DriveEase" />
80       <Button
81         outline
82         label="Продовжити з Google"
83         icon={AiOutlineGoogle}
84         onClick={() => {
85           signIn("google");
86         }}
87       />
88       <hr className="bg-slate-300 w-full h-px" />
89       <Input
90         id="name"
91         label="Ім'я"
92         disabled={isLoading}
93         register={register}
94         errors={errors}
95         required
96       />
97       <Input
98         id="email"
99         label="Email"
100        disabled={isLoading}
101        register={register}
102        errors={errors}
103        required
104      />
105      <Input
106        id="password"
107        label="Пароль"
108        disabled={isLoading}
109        register={register}
110        errors={errors}
111        required
112        type="password"
113      />
114      <Button
115        label={isLoading ? "Завантаження" : "Зареєструватися"}
116        onClick={handleSubmit(onSubmit)}
117      />
118      <p className="text-sm">
119        Маєте акаунт?{" "}
120        <Link className="underline" href="/login">
121          Увійти
122        </Link>
123      </p>
124    </>
125  );
126 };
127
128 export default RegisterForm;

```

Рисунок 3.6 – Продовження компоненту «RegisterForm»

Зображено фрагмент коду (рис. 3.7), в якому описана функція керування списком автомобілів у кошику користувача. Ця функція, «CartContextProvider», відповідає за додавання, видалення, збільшення та зменшення кількості автомобілів у списку, а також за керування загальною кількістю та сумою.

Спочатку, створюється контекст за допомогою «createContext» та визначається його тип «CartContextType», який містить методи та властивості для роботи з кошиком. Потім створюються стани для збереження кількості товарів, загальної суми, списку автомобілів та ідентифікатора оплати.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

При першому завантаженні компоненту виконується побічний ефект за допомогою «useEffect», який отримує збережені в локальному сховищі дані про список автомобілів та ідентифікатор оплати, і встановлює їх в стани компоненту.

Ще один побічний ефект обчислює загальну кількість товарів та суму в списку при зміні списку автомобілів, використовуючи метод «reduce».

```

11 type CartContextType = {
12   cartTotalQty: number;
13   cartTotalAmount: number;
14   cartCars: CartCarType[] | null;
15   handleAddCarToCart: (car: CartCarType) => void;
16   handleRemoveCarFromCart: (car: CartCarType) => void;
17   handleCartQtyIncrease: (car: CartCarType) => void;
18   handleCartQtyDecrease: (car: CartCarType) => void;
19   handleClearCart: () => void;
20   paymentIntent: string | null;
21   handleSetPaymentIntent: (val: string | null) => void;
22 };
23
24 export const CartContext = createContext<CartContextType | null>(null);
25
26 interface Props {
27   [propName: string]: any;
28 }
29
30 export const CartContextProvider = (props: Props) => {
31   const [cartTotalQty, setCartTotalQty] = useState(0);
32   const [cartTotalAmount, setCartTotalAmount] = useState(0);
33   const [cartCars, setCartCars] = useState<CartCarType[] | null>(null);
34   const [paymentIntent, setPaymentIntent] = useState<string | null>(null);
35
36   useEffect(() => {
37     const cartItems: any = localStorage.getItem("driveEaseCartItems");
38     const cCars: CartCarType[] | null = JSON.parse(cartItems);
39     const driveEasePaymentIntent: any = localStorage.getItem(
40       "driveEasePaymentIntent"
41     );
42     const paymentIntent: string | null = JSON.parse(driveEasePaymentIntent);
43
44     setCartCars(cCars);
45     setPaymentIntent(paymentIntent);
46   }, []);
47
48   useEffect(() => {
49     const getTotals = () => {
50       if (cartCars) {
51         const { total, qty } = cartCars?.reduce(
52           (acc, item) => {
53             const itemTotal = item.price * item.quantity;
54
55             acc.total += itemTotal;
56             acc.qty += item.quantity;
57
58             return acc;
59           },
60           {
61             total: 0,
62             qty: 0,
63           }
64         );
65
66         setCartTotalQty(qty);
67         setCartTotalAmount(total);
68       }
69     };

```

Рисунок 3.7 – Функція «CartContextProvider»

Метод «handleAddCarToCart» (рис. 3.8) відповідає за додавання автомобіля до списку. Він додає новий автомобіль до поточного списку, зберігає оновлений список в локальному сховищі та відображає сповіщення про успішне додавання.

```

74     const handleAddCarToCart = useCallback((car: CartCarType) => {
75       setCartCars((prev) => {
76         let updatedCart;
77
78         if (prev) {
79           updatedCart = [...prev, car];
80         } else {
81           updatedCart = [car];
82         }
83
84         toast.success("Авто додане до списку");
85         localStorage.setItem("driveEaseCartItems", JSON.stringify(updatedCart));
86         return updatedCart;
87       });
88     }, []);

```

Рисунок 3.8 – Метод «handleAddCarToCart»

Метод «handleRemoveCarFromCart» (рис. 3.9) видаляє автомобіль зі списку за його ідентифікатором, оновлює список в стані та локальному сховищі, і відображає сповіщення про успішне видалення.

```

90     const handleRemoveCarFromCart = useCallback(
91       (car: CartCarType) => {
92         if (cartCars) {
93           const filteredCars = cartCars.filter((item) => {
94             return item.id !== car.id;
95           });
96
97           setCartCars(filteredCars);
98           toast.success("Авто видалене зі списку");
99           localStorage.setItem(
100             "driveEaseCartItems",
101             JSON.stringify(filteredCars)
102           );
103         }
104       },
105       [cartCars]
106     );

```

Рисунок 3.9 – Метод «handleRemoveCarFromCart»

Методи «handleCartQtyIncrease» (рис. 3.10) та «handleCartQtyDecrease» (рис. 3.11) відповідають за збільшення та зменшення кількості годин оренди автомобіля. Вони перевіряють граничні значення (120 годин та 1 година відповідно) і відповідно оновлюють кількість автомобілів у списку та зберігають оновлений список в локальному сховищі.

```

108     const handleCartQtyIncrease = useCallback(
109       (car: CartCarType) => {
110         let updatedCart;
111
112         if (car.quantity === 120) {
113           return toast.error("Максимально 120 годин");
114         }
115
116         if (cartCars) {
117           updatedCart = [...cartCars];
118
119           const existingIndex = cartCars.findIndex((item) => item.id === car.id);
120
121           if (existingIndex > -1) {
122             updatedCart[existingIndex].quantity = ++updatedCart[existingIndex]
123               .quantity;
124           }
125
126           setCartCars(updatedCart);
127           localStorage.setItem("driveEaseCartItems", JSON.stringify(updatedCart));
128         }
129       },
130       [cartCars]
131     );

```

Рисунок 3.10 – Метод «handleCartQtyIncrease»

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		38

```

133     const handleCartQtyDecrease = useCallback(
134       (car: CartCarType) => {
135         let updatedCart;
136
137         if (car.quantity === 1) {
138           return toast.error("Мінімально 1 година");
139         }
140
141         if (cartCars) {
142           updatedCart = [...cartCars];
143
144           const existingIndex = cartCars.findIndex((item) => item.id === car.id);
145
146           if (existingIndex > -1) {
147             updatedCart[existingIndex].quantity = --updatedCart[existingIndex]
148               .quantity;
149           }
150
151           setCartCars(updatedCart);
152           localStorage.setItem("driveEaseCartItems", JSON.stringify(updatedCart));
153         }
154       },
155       [cartCars]
156     );

```

Рисунок 3.11 – Метод «handleCartQtyDecrease»

Метод «handleClearCart» (рис. 3.12) очищує список автомобілів та скидає загальну кількість товарів, а також видаляє дані з локального сховища.

```

158     const handleClearCart = useCallback(() => {
159       setCartCars(null);
160       setCartTotalQty(0);
161       localStorage.setItem("driveEaseCartItems", JSON.stringify(null));
162     }, [cartCars]);

```

Рисунок 3.12 – Метод «handleClearCart»

Метод «handleSetPaymentIntent» (рис. 3.13) встановлює ідентифікатор оплати та зберігає його в локальному сховищі.

```

164     const handleSetPaymentIntent = useCallback(
165       (val: string | null) => {
166         setPaymentIntent(val);
167         localStorage.setItem("driveEasePaymentIntent", JSON.stringify(val));
168       },
169       [paymentIntent]
170     );

```

Рисунок 3.13 – Метод «handleSetPaymentIntent»

У кінці, «CartContextProvider» повертає провайдер контексту (рис. 3.14), який надає всі методи та властивості для роботи зі списком автомобілів. Для доступу до цього контексту використовується хук «useCart», який перевіряє наявність контексту та повертає його значення.

```

185     return <CartContext.Provider value={value} {...props} />;
186   };
187
188   export const useCart = () => {
189     const context = useContext(CartContext);
190
191     if (context === null) {
192       throw new Error(
193         "useCart має використовуватися в межах CartContextProvider"
194       );
195     }
196
197     return context;
198   };

```

Рисунок 3.14 – Повертання провайдеру контексту

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

На рисунку 3.15 зображено компонент «AddRating», який призначений для додавання користувачем оцінки та відгуку на автомобіль.

Компонент приймає пропси «car» та «user». Об'єкт «car» містить інформацію про автомобіль та його відгуки, а об'єкт «user» містить інформацію про користувача та його замовлення.

Функція «AddRating» починається з визначення стану «isLoading», який використовується для відстеження процесу завантаження під час відправки форми. Використовується хук «useRouter» для керування маршрутизацією.

```
15 interface AddRatingProps {
16   car: Car & {
17     reviews: Review[];
18   };
19   user:
20     | (SafeUser & {
21       orders: Order[];
22     })
23     | null;
24 }
25
26 const AddRating: React.FC<AddRatingProps> = ({ car, user }) => {
27   const [isLoading, setIsLoading] = useState(false);
28   const router = useRouter();
29
30   const {
31     register,
32     handleSubmit,
33     setValue,
34     reset,
35     formState: { errors },
36   } = useForm<FieldValues>({
37     defaultValues: {
38       comment: "",
39       rating: 0,
40     },
41   });
42
43   const setCustomValue = (id: string, value: any) => {
44     setValue(id, value, {
45       shouldTouch: true,
46       shouldDirty: true,
47       shouldValidate: true,
48     });
49   };
50
51   const onSubmit: SubmitHandler<FieldValues> = async (data) => {
52     setIsLoading(true);
53     if (data.rating === 0) {
54       setIsLoading(false);
55       return toast.error("No rating selected");
56     }
57     const ratingData = { ...data, userId: user?.id, car: car };
58
59     axios
60       .post("/api/rating", ratingData)
61       .then(() => {
62         toast.success("Rating submitted");
63         router.refresh();
64         reset();
65       })
66       .catch((error) => {
67         console.log(error);
68         toast.error("Something went wrong");
69       })
70       .finally(() => {
71         setIsLoading(false);
72       });
73 }
```

Рисунок 3.15 – Компонент «AddRating»

Використовується хук «useForm» для керування станом форми, валідацією та обробкою помилок, з початковими значеннями полів «comment» та «rating».

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

Функція «setCustomValue» використовується для встановлення значення полів форми з урахуванням їх стану (торкання, брудність, валідація).

Функція «onSubmit» відповідає за обробку подання форми. Після відправки форми встановлюється стан завантаження, і якщо рейтинг не вибрано, відображається повідомлення про помилку. В іншому випадку, дані форми надсилаються на сервер за допомогою «axios». У разі успіху відображається сповіщення про успішне додавання рейтингу, сторінка оновлюється, і форма скидається. У разі помилки відображається відповідне повідомлення.

Якщо користувач або автомобіль відсутні, компонент не відображається. Перевіряється, чи користувач вже залишив відгук або чи виконано доставку замовлення з цим автомобілем. Якщо користувач вже залишив відгук або доставка не виконана, компонент не відображається.

Якщо всі перевірки пройдено успішно, відображається форма для додавання оцінки та відгуку, яка містить заголовок, компонент для вибору рейтингу, поле для вводу коментаря та кнопку для відправки форми.

```
73   };
74
75   if (!user || !car) return null;
76
77   const deliveredOrder = user?.orders.some(
78     (order) =>
79       order.cars.find((item) => item.id === car.id) &&
80       order.deliveryStatus === "виконано"
81   );
82
83   const userReview = car?.reviews.find((review: Review) => {
84     return review.userId === user.id;
85   });
86
87   if (userReview || !deliveredOrder) return null;
88
89   return (
90     <div className="flex flex-col gap-2 max-w-[500px]">
91       <Heading title="Ваша оцінка" />
92       <Rating
93         onChange={(event, newValue) => {
94           setCustomValue("rating", newValue);
95         }}
96       />
97       <Input
98         id="comment"
99         label="Ваш коментар..."
100         disabled={isLoading}
101         register={register}
102         errors={errors}
103         required
104       />
105       <Button
106         label={isLoading ? "Завантаження" : "Надіслати відгук"}
107         onClick={handleSubmit(onSubmit)}
108       />
109     </div>
110   );
111 };
112
113 export default AddRating;
```

Рисунок 3.15 – Продовження компоненту «AddRating»

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3. Опис інструкції користувача

На головній сторінці сервісу каршерінгу (рис. 3.16) розміщено основне навігаційне меню, яке забезпечує зручний доступ до всіх основних розділів сайту. Крім того, на сторінці знаходиться банер з актуальними оголошеннями про знижки та спеціальні пропозиції, що привертають увагу користувачів. Нижче розташований перелік усіх доступних для оренди автомобілів, представлений у вигляді зручного та інтуїтивно зрозумілого каталогу.

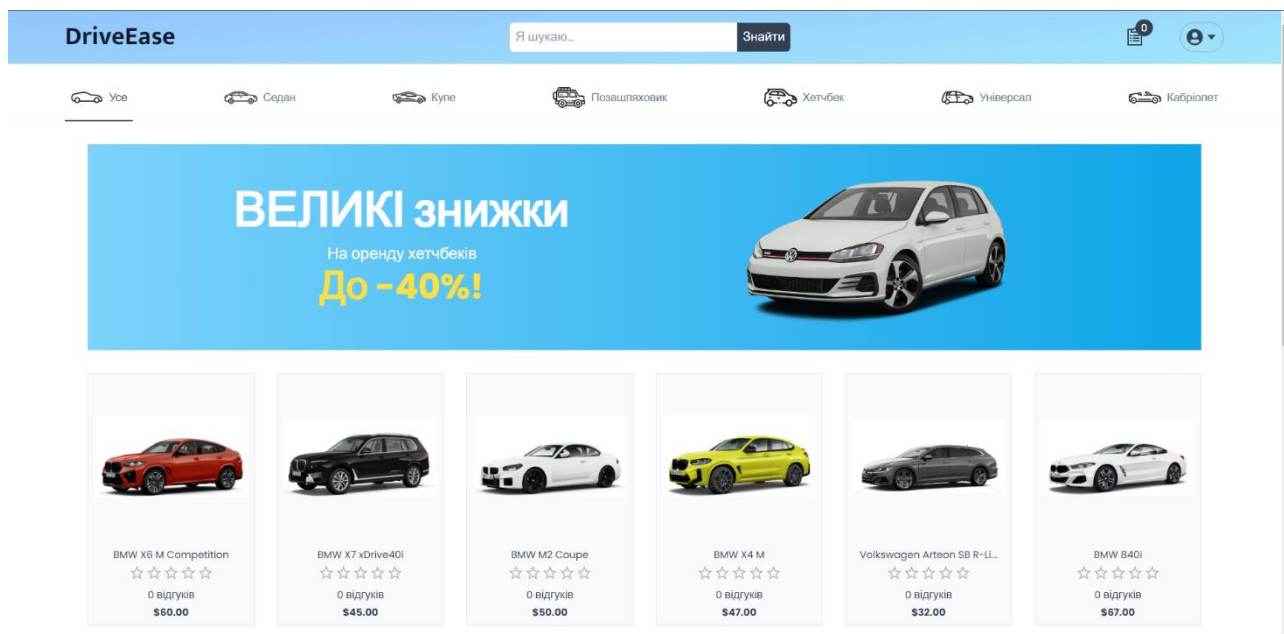


Рисунок 3.16 – Головна сторінка сервісу каршерінгу

На головній сторінці представлено всі автомобілі, доступні в сервісі каршерінгу (рис. 3.17). Користувачі можуть переглядати фотографії кожного автомобіля, що дозволяє отримати візуальне уявлення про транспортний засіб перед орендою. Також відображається середній рейтинг автомобіля, який базується на оцінках користувачів, а також кількість написаних відгуків, що допомагає зробити більш обґрунтований вибір.

У сервісі реалізована функція пошуку автомобіля за назвою (рис. 3.18), що значно полегшує процес підбору транспортного засобу. Для цього користувачам потрібно ввести назву автомобіля у відповідне поле пошуку та натиснути кнопку «Знайти». Дана функція забезпечує швидкий і зручний доступ до необхідної інформації, допомагаючи користувачам легко знаходити автомобілі, які їх цікавлять.

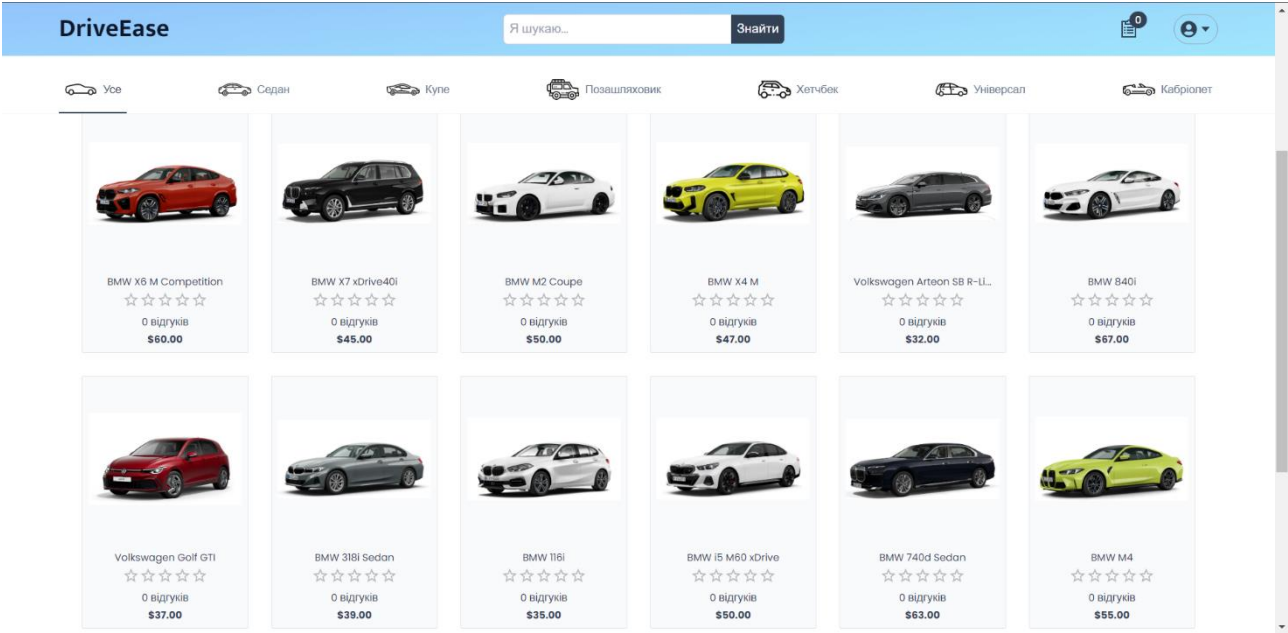


Рисунок 3.17 – Перелік всіх автомобілів

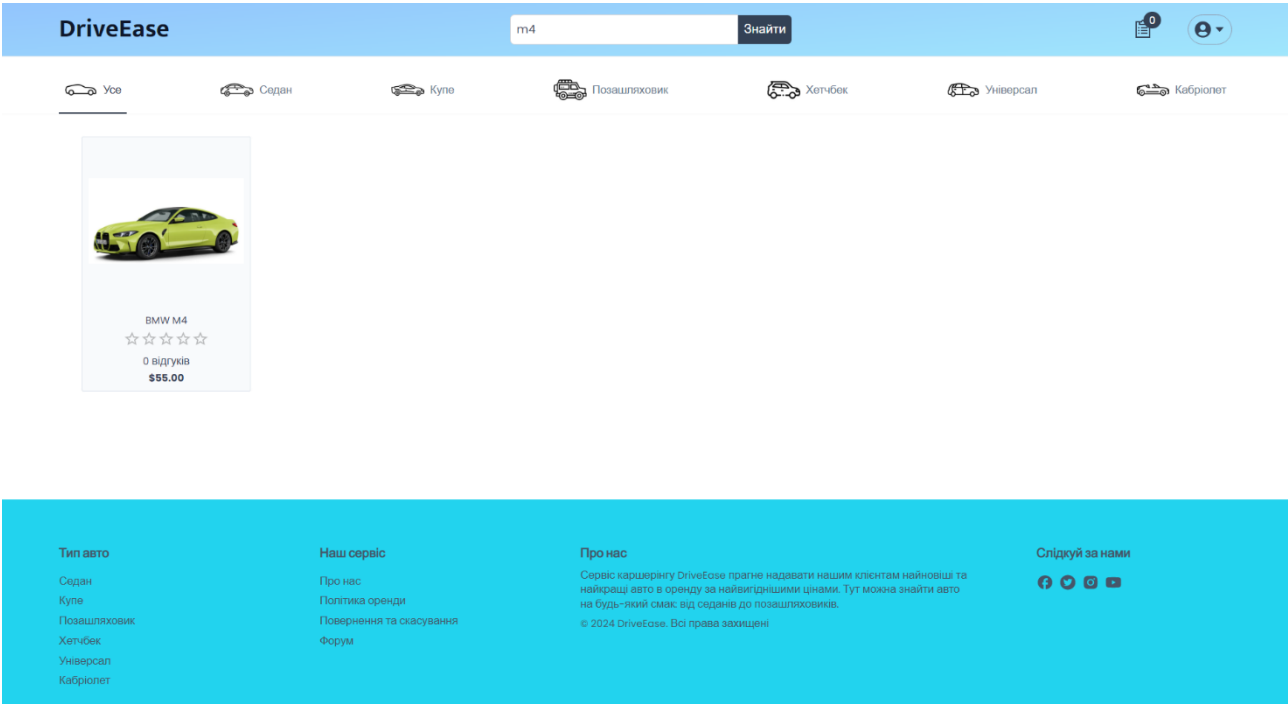


Рисунок 3.18 – Пошук автомобіля за назвою: результати знайдені

Якщо пошук автомобіля не дав результатів, на екрані з'являється відповідне повідомлення: «Авто не знайдено. Натисніть «Усе», щоб очистити фільтри» (рис. 3.19). Це сповіщення інформує користувача про відсутність результатів за введеними критеріями та пропонує простий спосіб скинути всі фільтри для нової спроби пошуку.

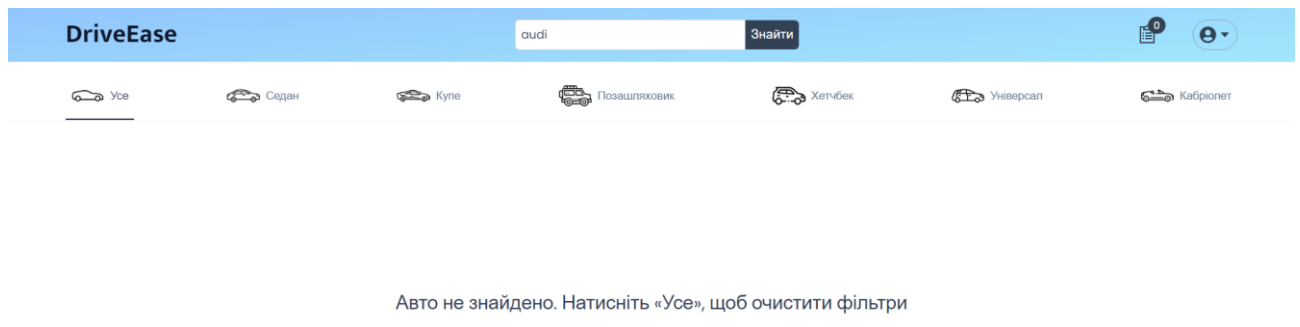


Рисунок 3.19 – Пошук автомобіля за назвою: результати не знайдені

Після того як користувач обрав автомобіль та натиснув на нього, він автоматично перенаправляється на сторінку детального опису цього автомобіля (рис. 3.20). На цій сторінці відображаються фотографії автомобіля, його назва, детальні характеристики, відгуки інших користувачів та можливість додати автомобіль до списку бажаних. Такий підхід забезпечує користувачам зручний та швидкий доступ до всієї необхідної інформації про обраний автомобіль, сприяючи більш інформованому та комфортному вибору.

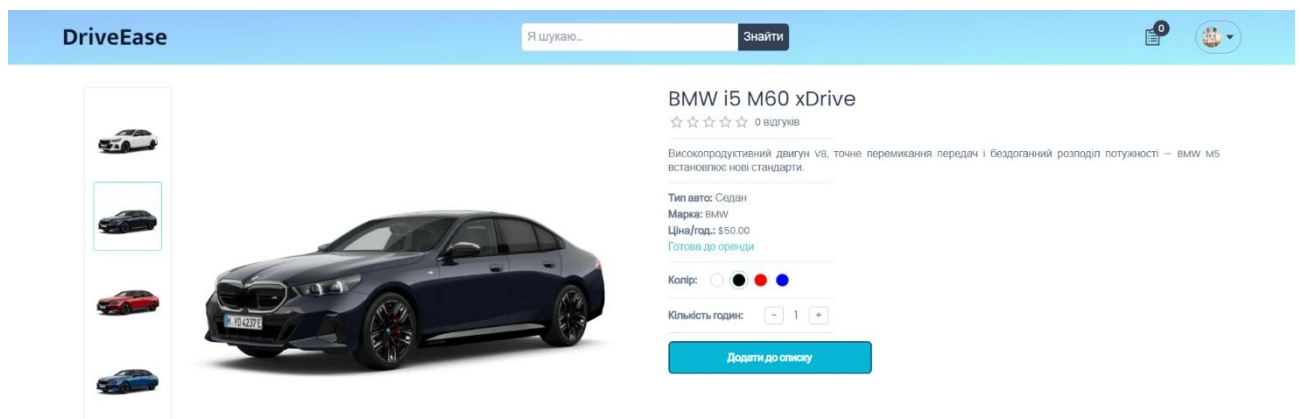


Рисунок 3.20 – Відображення вибраного автомобіля

На сторінці обраного автомобіля користувач має можливість вибрати бажаний колір із доступного переліку варіантів (рис. 3.21). Після вибору колір миттєво оновлюється, і фотографія автомобіля автоматично змінюється, відображаючи вибраний колір. Це дозволяє користувачам візуально оцінити модель автомобіля у різних кольорах, забезпечуючи більш інтерактивний та персоналізований досвід.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

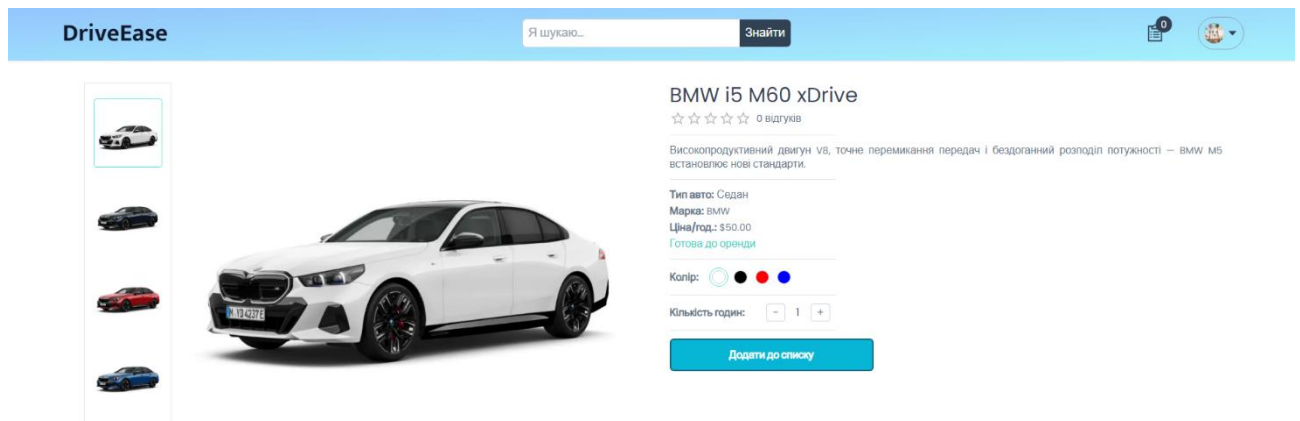


Рисунок 3.21 – Зміна кольору вибраного автомобіля

Якщо автомобіль недоступний для оренди, на сайті відображається повідомлення «Недоступна» (рис. 3.22).

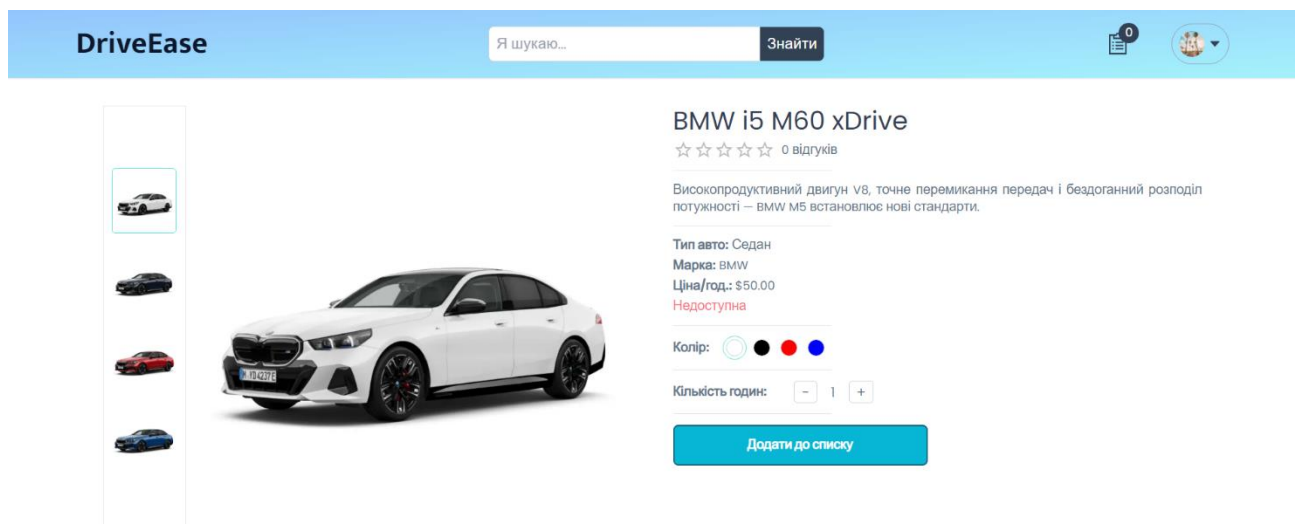


Рисунок 3.22 – Повідомлення «Недоступна»

На сайті користувач має можливість увійти та зареєструватись, використовуючи відповідні посилання, доступні у випадаючому списку меню (рис. 3.23). Це забезпечує зручний доступ до авторизаційних опцій без необхідності виходити з поточної сторінки.

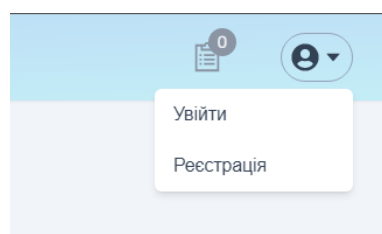


Рисунок 3.23 – Випадаюче меню користувача

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

Користувач має можливість авторизуватися у сервісі двома способами: через обліковий запис Google або введенням електронної пошти та пароля, за умови якщо він вже зареєстрований (рис. 3.24). Це надає користувачам гнучкість у виборі методу входу, що відповідає їхнім вподобанням та забезпечує зручність при використанні сайту.

Рисунок 3.24 – Сторінка входу

Якщо користувач ще не має облікового запису, він має можливість зареєструватися двома способами: через акаунт Google або шляхом введення свого імені, електронної пошти і обраного пароля (рис. 3.25).

Рисунок 3.25 – Сторінка реєстрації

Після успішного введення необхідних даних та вдалого входу на сайт, користувач бачить повідомлення з текстом «Успішний вхід. Перенаправлення...» (рис. 3.26), яке підтверджує його авторизацію.

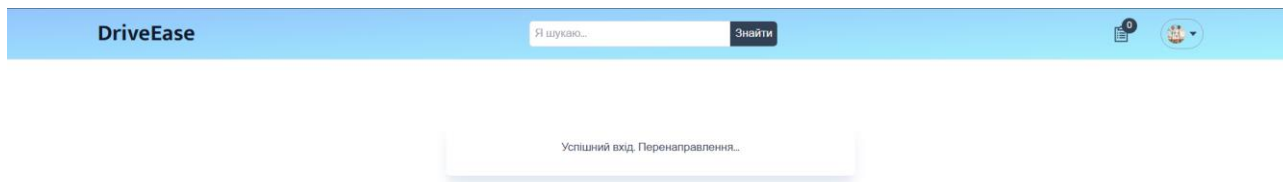


Рисунок 3.26 – Успішне виконання входу

Після перенаправлення зі сторінки з підтвердженням успішного входу на сайт, користувачу відображається список обраних автомобілів (рис. 3.27).

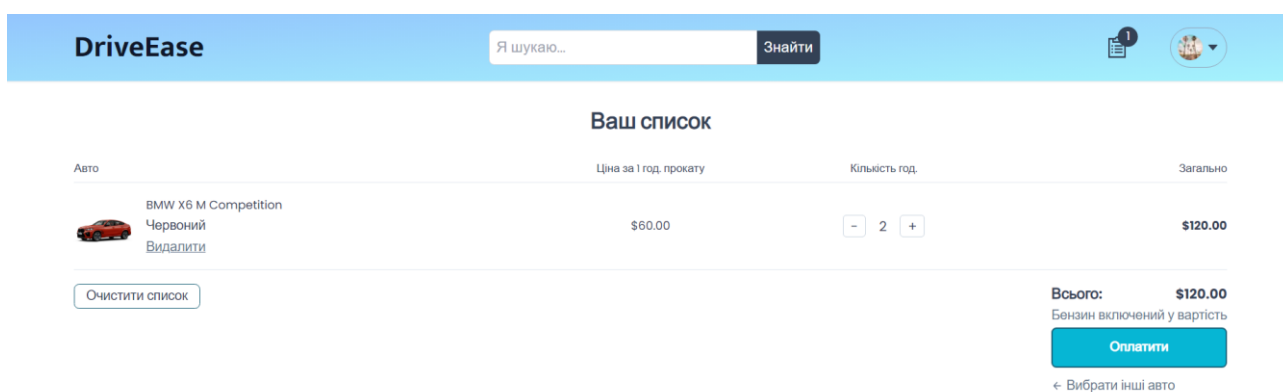


Рисунок 3.27 – Наповнена сторінка списку вибраних автомобілів

Якщо користувач ще не додав жодного автомобіля до свого списку, він побачить повідомлення «Ваш список пустий», до якого додається посилання на перегляд доступних автомобілів (рис. 3.28).

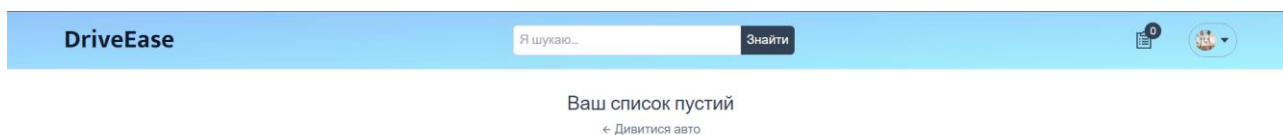


Рисунок 3.28 – Сторінка списку вибраних автомобілів

Після ознайомлення зі списком обраних автомобілів, їх кількістю, цінами та загальною вартістю, користувач може забронювати обраний автомобіль, натиснувши кнопку «Оплатити». Це спричинить відображення форми для введення даних, необхідних для оформлення оренди (рис. 3.29). У цій формі

потрібно вказати прізвище, ім'я, адресу, поштовий індекс та дані банківської картки орендатора.

Рисунок 3.29 – Форма оформлення оренди

Після успішного виконання платежу на екрані з'явиться повідомлення «Платіж пройшов успішно». Крім того, користувачу буде доступна кнопка «Переглянути ваші замовлення», за допомогою якої він зможе переглянути історію своїх замовлень оренди автомобілів (рис. 3.30).

Платіж пройшов успішно!

Переглянути ваші
бронювання

Рисунок 3.30 – Результат успішного платежу

Користувач має можливість переглядати свій список бронювань, де відображається загальна сума замовлення, статус платежу і стан оренди, дату замовлення (рис. 3.31).

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

DriveEase

Я шукаю...

Знайти

0

Бронювання

☐	ID	Ім'я клієнта	Сума (USD)	Статус платежу	Статус оренди	Дата	Переглянути
☐	667b354e4706ee75c440163b	TEMBId	\$55.00	успішно ✓	в оренді ⚙	29 minutes ago	
☐	667b354e4706ee75c440163a	TEMBId	\$120.00	успішно ✓	виконано ✓	29 minutes ago	
☐	667b34844706ee75c4401639	TEMBId	\$67.00	успішно ✓	виконано ✓	32 minutes ago	
☐	667b34844706ee75c4401638	TEMBId	\$67.00	успішно ✓	виконано ✓	32 minutes ago	

Rows per page: 9 1-4 of 4

Рисунок 3.31 – Перегляд списку бронювань

Крім того, є можливість детального ознайомлення з кожним конкретним бронюванням для перегляду всіх його деталей і параметрів (рис. 3.32).

DriveEase

Я шукаю...

Знайти

0

Деталі бронювання

Номер броні: 667b354e4706ee75c440163b

Статус оплати: оплачено ✓

Статус оренди: в оренді ⚙

Дата: 37 minutes ago

Орендовані машини

Машина	Ціна/год.	Кількість год.	Загальна сума
BMW M4 Золотий	\$55.00	1	\$55.00

Тип авто

Седан

Купе

Позашляховик

Хетчбек

Універсал

Кабріолет

Наш сервіс

Про нас

Політика оренди

Повернення та скасування

Форум

Про нас

Сервіс каршерінгу DriveEase прагне надавати нашим клієнтам найновіші та найкращі авто в оренду за найвигіднішими цінами. Тут можна знайти авто на будь-який смак: від седанів до позашляховиків.

© 2024 DriveEase. Всі права захищені

Слідкуй за нами

Рисунок 3.32 – Перегляд деталей вибраного бронювання

Через меню доступний перехід на панель адміністратора, де адміністратор може керувати різними аспектами функціонування сайту (рис. 3.33).

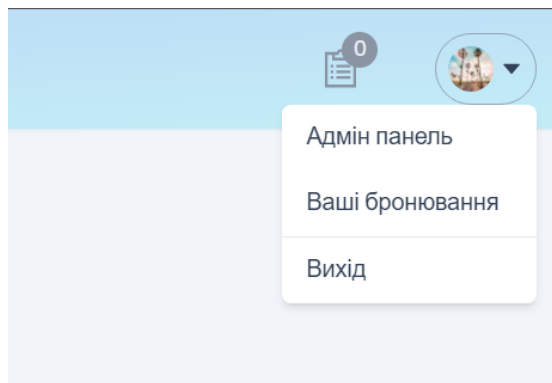


Рисунок 3.33 – Додаткове випадаюче меню користувача

На панелі адміністратора доступна можливість переглядати загальну статистику сервісу каршерінгу (рис. 3.34), включаючи такі показники, як загальний дохід, кількість автомобілів, оплачені оренди, загальна кількість оренд, неоплачені оренди та загальна кількість користувачів.

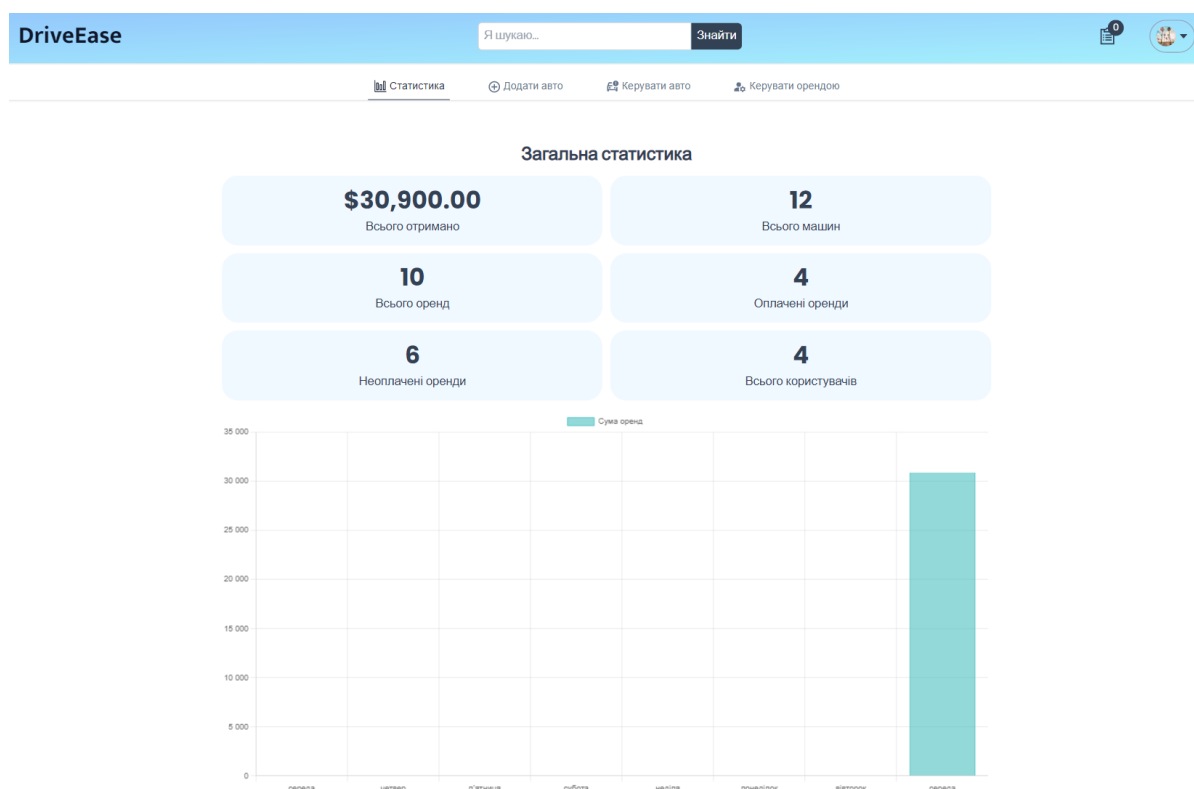


Рисунок 3.34 – Загальна статистика в панелі адміністратора

Адміністратор може керувати автомобілями, а саме переглядати всі автомобілі, їхню детальну інформацію, змінити наявність та видалити відповідний запис автомобіля (рис. 3.35).

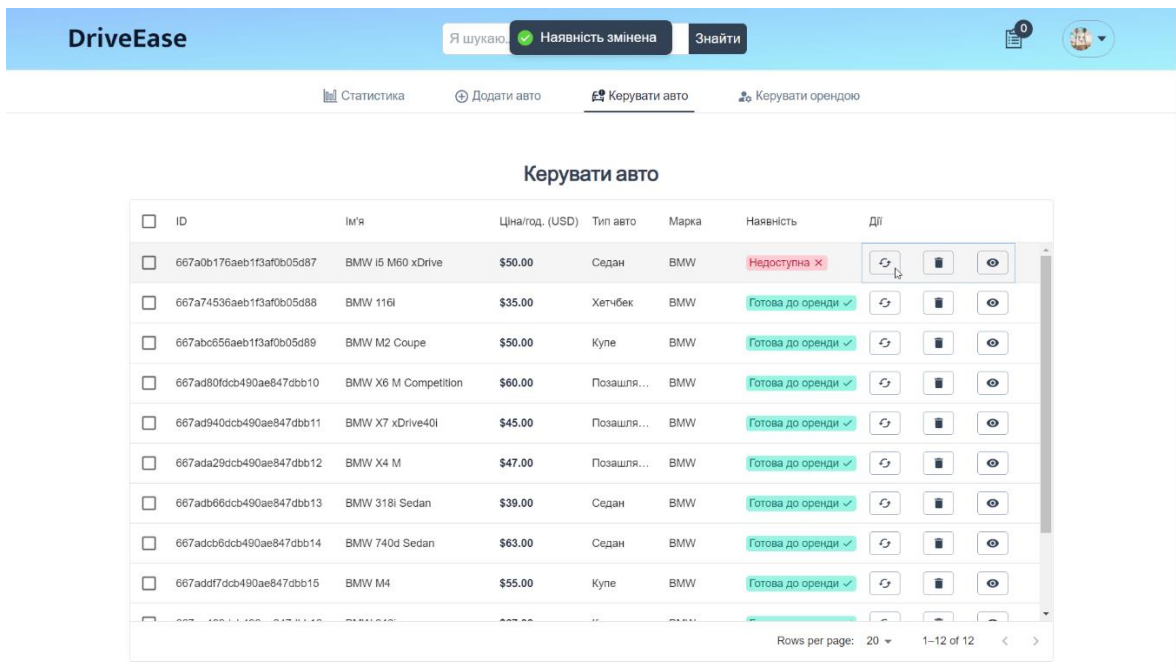


Рисунок 3.35 – Керування авто в панелі адміністратора

На панелі адміністратора є функція додавання нового автомобіля (рис. 3.36). Щоб додати новий автомобіль, адміністратор повинен ввести назву, ціну за годину, марку, опис, вказати доступність для оренди, тип авто, його колір та додати фотографії.

Додавання авто

Назва

Ціна/год.

Марка

Опис

☐ Доступне для оренди

Виберіть тип авто

Седан

Купе

Позашляховик

Хетчбек

Універсал

Кабріолет

Виберіть доступні кольори автомобіля та завантажте зображення.
 Необхідно завантажити зображення для кожного вибраного кольору, інакше копіїр буде проігноровано.

☐ Білий
 ☐ Чорний

☐ Срібний
 ☐ Сірий

☐ Червоний
 ☐ Золотий

☐ Блакитний
 ☐ Графіт

Додати авто

Рисунок 3.36 – Додавання автомобіля адміністратором

Також адміністратор може керувати орендою, а саме переглядати всі бронювання, їхню детальну інформацію, змінити статус платежу і оренди (рис. 3.37).

ID	Ім'я клієнта	Сума (USD)	Статус платежу	Статус оренди	Дата	Дії
667b354e4706ee75c440163b	TEMb14	\$55.00	успішно ✓	виконано ✓	an hour ago	[Car Icon] [Checkmark] [Eye Icon]
667b354e4706ee75c440163a	TEMb14	\$120.00	успішно ✓	в оренді 🚗	an hour ago	[Car Icon] [Checkmark] [Eye Icon]
667b34844706ee75c4401639	TEMb14	\$67.00	успішно ✓	виконано ✓	an hour ago	[Car Icon] [Checkmark] [Eye Icon]
667b34844706ee75c4401638	TEMb14	\$67.00	успішно ✓	виконано ✓	an hour ago	[Car Icon] [Checkmark] [Eye Icon]
6679b1f16aeb1f3af0b05d80	artem	\$3,000.00	очікує ⌚	виконано ✓	a day ago	[Car Icon] [Checkmark] [Eye Icon]
6679b1f16aeb1f3af0b05d7f	artem	\$1,000.00	очікує ⌚	виконано ✓	a day ago	[Car Icon] [Checkmark] [Eye Icon]
667890de59dbe35912b14969	artem	\$1,000.00	очікує ⌚	виконано ✓	2 days ago	[Car Icon] [Checkmark] [Eye Icon]
667890de59dbe35912b14968	artem	\$1,000.00	очікує ⌚	виконано ✓	2 days ago	[Car Icon] [Checkmark] [Eye Icon]
66785ab859dbe35912b14966	artem	\$1,000.00	очікує ⌚	виконано ✓	2 days ago	[Car Icon] [Checkmark] [Eye Icon]

Рисунок 3.37 – Керування орендою в панелі адміністратора

3.4. Тестування роботи сервісу

Функціональне тестування — це різновид тестування програмного забезпечення, спрямований на перевірку роботи сервісу або програмного забезпечення. Воно зосереджується на забезпеченні того, щоб сервіс виконував свої функції відповідно до встановлених вимог і задовольняла очікування бізнесу.

Основна мета функціонального тестування полягає у перевірці функціональних можливостей сервісу та її взаємодії з різними компонентами. Це включає тестування вхідних і вихідних даних програмного забезпечення, обробку даних, взаємодію з користувачем і реакцію сервісу на різноманітні сценарії та умови [20].

Спершу потрібно провести тестування користувацького інтерфейсу сервісу каршерінгу. Це включає перевірку функціональності входу та реєстрації, пошук автомобілів за назвою, відображення інформації про автомобілі, можливість написання коментарів, перевірку процесу оплати, а також коректність виходу з сервісу та інші важливі операції.

Таблиця 3.1

Тестування користувацької сторони сервісу каршерінгу

№	Опис тесту	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Перевірка входу	Заповнення пошти та паролю або вхід за допомогою google акаунта	Виконання входу, перенаправлення у список бронювань	Виконання входу, перенаправлення у список бронювань	Пройдено
2	Перевірка реєстрації	Заповнення імені, пошти, паролю, або реєстрація за допомогою google акаунта	Виконання реєстрації, перенаправлення у список бронювань	Виконання реєстрації, перенаправлення у список бронювань	Пройдено
3	Перевірка пошуку за назвою	Введення назви автомобіля в поле пошуку	Виведення списку авто із заданим ключовим словом у назві	Виведення списку авто із заданим ключовим словом у назві	Пройдено
4	Перевірка відображення інформації про автомобіль	Перехід на сторінку вибраного автомобіля	Виведення інформації про вибраний автомобіль	Виведення інформації про вибраний автомобіль	Пройдено
5	Перевірка написання коментаря	Ввести текстове поле, поставити оцінку, натиснути опублікувати	Відображення створеного коментаря	Відображення створеного коментаря	Пройдено
6	Перевірка бронювання	Додавання автомобіля до списку	Автомобіль доданий у список бронювань користувача	Автомобіль доданий у список бронювань користувача	Пройдено

7	Перевірка оплати	Ввести дані платіжної картки	Платіж пройшов успішно	Платіж пройшов успішно	Пройдено
8	Перевірка виходу	Натиснути кнопку виходу	Виконання виходу	Виконання виходу	Пройдено

Далі слід провести тестування адміністративної панелі. Це включає перевірку доступу до панелі адміністратора, оцінку коректності відображення статистичних даних, а також можливість перегляду та редагування інформації про бронювання та автомобілі.

Таблиця 3.2

Тестування адміністративної сторони сервісу каршерінгу

№	Опис тесту	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Перевірка переходу на панель адміністратора	Перейти на адмінпанель користувачем із відповідними правами	Відбувається перенаправлення на панель адміністратора	Відбувається перенаправлення на панель адміністратора	Пройдено
2	Перевірка відображення статистики	Перейти на адмінпанель користувачем із відповідними правами	Відображається статистика за орендованими автомобілями	Відображається статистика за орендованими автомобілями	Пройдено
3	Перевірка перегляду списку автомобілів	Перейти у адмінпанель адміністратору, натиснути керуванням автомобілями	Відображення списку автомобілів	Відображення списку автомобілів	Пройдено
4	Перевірка зміни даних про автомобіль	Перейти на адмінпанель користувачем із відповідними правами, задати новий	Зміна статусу автомобіля	Зміна статусу автомобіля	Пройдено

		статус відповідному автомобілю			
5	Перевірка видалення даних про автомобіль	Перейти на адмінпанель користувачем із відповідними правами, видалити дані про відповідний автомобіль	Видалення відповідного автомобіля	Видалення відповідного автомобіля	Пройдено
6	Перевірка перегляду списку бронювань	Перейти на адмінпанель користувачем із відповідними правами, натиснути керування орендою	Відображення списку бронювань	Відображення списку бронювань	Пройдено
7	Перевірка зміни статусів бронювань	Перейти на адмінпанель користувачем із відповідними правами, задати новий статус відповідному бронюванню	Зміна статусу бронювання	Зміна статусу бронювання	Пройдено

Висновки до третього розділу

У третьому розділі кваліфікаційної роботи було проведено детальний аналіз і опис інтерфейсу створеного сервісу каршерінгу. Було визначено ключові елементи інтерфейсу та їх функціональні можливості. Проаналізовано послідовність використання сервісу, охоплюючи всі етапи взаємодії користувача з сервісом, від головної сторінки до адміністративної панелі.

Також проведено функціональне тестування розробленого сервісу для перевірки його працездатності та відповідності вимогам. У ході тестування детально перевірено роботу як адміністративної панелі, так і користувацької частини сервісу каршерінгу.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було проведено дослідження технічного завдання для створення сервісу каршерінгу. Перший розділ зосереджений на чіткому формулюванні основних цілей та вимог до сервісу, враховуючи потреби майбутніх користувачів. Аналіз існуючих технологій дозволив визначити оптимальний стек технологій, який включає HTML, CSS, React.js, TypeScript, Tailwind CSS, Prisma, Firebase, MongoDB і Stripe. Цей вибір базується на їхніх перевагах і можливостях для забезпечення ефективного та стабільного функціонування сервісу каршерінгу.

Другий розділ роботи включав детальне вивчення функціональних та нефункціональних вимог до сервісу. Аналіз цих вимог дозволив сформулювати чітке уявлення про те, як сервіс повинен працювати та які вимоги вона повинна задовольняти. Була розроблена об'єктно-орієнтована модель сервісу, в якій детально описано всі основні об'єкти та їх взаємодію. Використання UML діаграм для ілюстрації взаємодії між компонентами сервісу дозволило зрозуміти і оптимізувати основні процеси роботи сервісу.

Третій розділ кваліфікаційної роботи був присвячений детальному аналізу та опису інтерфейсу створеного сервісу каршерінгу для користувачів. Були визначені ключові елементи інтерфейсу та їх функціональні можливості з метою забезпечення зручного та інтуїтивно зрозумілого використання. Функціональне тестування сервісу каршерінгу було проведене для перевірки його працездатності та відповідності вимогам. Результати тестування підтвердили коректну роботу адміністративної панелі та клієнтської частини сервісу.

У результаті виконання кваліфікаційної роботи був створений сучасний та функціональний сервіс каршерінгу, що відповідає найвищим стандартам якості та забезпечує зручний та безпечний досвід користувачів. Розроблена платформа використовує нові технології для забезпечення надійності та ефективності обслуговування.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		56

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Класичний прокат та каршерінг: у чому різниця? [Електронний ресурс] – Режим доступу до ресурсу: <https://autoprokat.net.ua/ua/articles/klasichnij-prokat-i-karshering.html>
2. Learning React: Modern Patterns for Developing React Apps, 2nd edition / Alex Banks, Eve Porcello, 2020 – 307 p.
3. React Explained: Your Step-by-Step Guide to React, 1st edition / Zac Gordon, 2019 – 366 p.
4. React Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://legacy.reactjs.org/docs/getting-started.html>
5. Firebase [Електронний ресурс] – Режим доступу до ресурсу: <https://firebase.google.com/>
6. W3Schools JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
7. Responsive Web Design with HTML5 and CSS, 4th edition / Ben Frain, 2022 – 498p.
8. Електронний HTML і CSS довідник [Електронний ресурс] – Режим доступу до ресурсу: <https://html-css.co.ua/>
9. Use Case Diagrams [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/use-case-diagram/>
10. Діаграма діяльності в UML [Електронний ресурс] – Режим доступу до ресурсу: <https://www.guru99.com/uk/uml-activity-diagram.html>
11. Class Diagram [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/>
12. Collaboration Diagrams [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/collaboration-diagrams-unified-modeling-languageuml/>
13. Learning TypeScript: Enhance Your Web Development Skills Using Type-Safe JavaScript, 1st edition / John Goldberg, 2022 – 317 p.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

14. TypeScript Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.typescriptlang.org/docs/>
15. Next.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs>
16. Tailwind CSS Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://tailwindcss.com/docs>
17. Stripe [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.stripe.com/>
18. Mongodb [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/mongodb>
19. Prisma Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.prisma.io/docs>
20. Functional Testing: A Detailed Guide [Електронний ресурс] – Режим доступу до ресурсу: <https://www.browserstack.com/guide/functional-testing>

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

Файл головної сторінки.

Лістинг Page.tsx

```

export const revalidate = 0;

import Container from "../components/Container";
import HomeBanner from "../components/HomeBanner";
import { truncateText } from "@/utils/truncateText";
import CarCard from "../components/cars/CarCard";
import getCars, { ICarParams } from "@/actions/getCars";
import NullData from "../components/NullData";
import ClientOnly from "../components/ClientOnly";

interface HomeProps {
  searchParams: ICarParams;
}

const HomePageContent: React.FC<{ cars: any[] }> = ({ cars }) => {
  return (
    <div className="p-8">
      <Container>
        <ClientOnly>
          <HomeBanner />
        </ClientOnly>
        <div className="grid grid-cols-2 sm:grid-cols-3 lg:grid-cols-4 xl:grid-
cols-5 2xl:grid-cols-6 gap-8">
          {cars.map((car: any) => (
            <CarCard data={car} key={car.id} />
          ))}
        </div>
      </Container>
    </div>
  );
};

export default async function Home({ searchParams }: HomeProps) {
  const cars = await getCars(searchParams);

  if (cars.length === 0) {
    return (
      <NullData title="Авто не знайдено. Натисніть «Усе», щоб очистити фільтри" />
    );
  }

  // Алгоритм Фішера-Єйтса
  function shuffleArray(array: any) {
    for (let i = array.length - 1; i > 0; i--) {
      const j = Math.floor(Math.random() * (i + 1));
      [array[i], array[j]] = [array[j], array[i]];
    }
  }

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    return array;
  }

  const shuffledCars = shuffleArray(cars);

  return <HomePageContent cars={shuffledCars} />;
}

```

Додаток Б

Файл форми входу.

Лістинг LoginForm.tsx

```

"use client";

import { useEffect, useState } from "react";
import Heading from "../components/Heading";
import Input from "../components/inputs/Input";
import { FieldValues, SubmitHandler, useForm } from "react-hook-form";
import Button from "../components/Button";
import Link from "next/link";
import { AiOutlineGoogle } from "react-icons/ai";
import { signIn } from "next-auth/react";
import { useRouter } from "next/navigation";
import { toast } from "react-hot-toast";
import { SafeUser } from "@types";

interface LoginFormProps {
  currentUser: SafeUser | null;
}

const LoginForm: React.FC<LoginFormProps> = ({ currentUser }) => {
  const [isLoading, setIsLoading] = useState(false);
  const {
    register,
    handleSubmit,
    formState: { errors },
  } = useForm<FieldValues>({
    defaultValues: {
      email: "",
      password: "",
    },
  });

  const router = useRouter();

  useEffect(() => {
    if (currentUser) {
      router.push("/cart");
    }
  });

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        router.refresh();
    }
}, []);

const onSubmit: SubmitHandler<FieldValues> = (data) => {
    setIsLoading(true);
    signIn("credentials", {
        ...data,
        redirect: false,
    }).then((callback) => {
        setIsLoading(false);

        if (callback?.ok) {
            router.push("/cart");
            router.refresh();
            toast.success("Успішний вхід");
        }

        if (callback?.error) {
            toast.error(callback.error);
        }
    });
};

if (currentUser) {
    return <p className="text-center">Успішний вхід. Перенаправлення...</p>;
}

return (
    <>
        <Heading title="Увійти в DriveEase" />
        <Button
            outline
            label="Продовжити з Google"
            icon={AiOutlineGoogle}
            onClick={() => {
                signIn("google");
            }}
        />
        <hr className="bg-slate-300 w-full h-px" />
        <Input
            id="email"
            label="Email"
            disabled={isLoading}
            register={register}
            errors={errors}
            required
        />
        <Input
            id="password"
            label="Пароль"

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        disabled={isLoading}
        register={register}
        errors={errors}
        required
        type="password"
      />
      <Button
        label={isLoading ? "Завантаження" : "Увійти"}
        onClick={handleSubmit(onSubmit)}
      />
      <p className="text-sm">
        Немає облікового запису?{" "}
        <Link className="underline" href="/register">
          Зареєструватися
        </Link>
      </p>
    </>
  );
};

export default LoginForm;

```

Додаток В

Файл сторінки списку.

Лістинг CartClient.tsx

```

"use client";
import { useCart } from "@/hooks/useCart";
import Link from "next/link";
import { MdArrowBack } from "react-icons/md";
import Heading from "../components/Heading";
import Button from "../components/Button";
import ItemContent from "../ItemContent";
import { formatPrice } from "@/utils/formatPrice";
import { SafeUser } from "@/types";
import { useRouter } from "next/navigation";

interface CartClientProps {
  currentUser: SafeUser | null;
}

const CartClient: React.FC<CartClientProps> = ({ currentUser }) => {
  const { cartCars, handleClearCart, cartTotalAmount } = useCart();

  const router = useRouter();

  if (!cartCars || cartCars.length === 0) {
    return (
      <div className="flex flex-col items-center">
        <div className="text-2xl">Ваш список пустий</div>

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    <div>
      <Link
        href={"/"}
        className="
          text-slate-500 flex items-center gap-1
          mt-2
          "
        >
        <MdArrowBack />
        <span>Дивитися авто</span>
      </Link>
    </div>
  </div>
);
}

return (
  <div>
    <Heading title="Ваш список" center />
    <div
      className="grid
        grid-cols-5
        text-xs
        gap-4
        pb-2
        items-center
        mt-8
        "
      >
      <div className="col-span-2 justify-self-start">Авто</div>
      <div className="justify-self-center">Ціна за 1 год. прокату</div>
      <div className="justify-self-center">Кількість год.</div>
      <div className="justify-self-end">Загально</div>
    </div>
    <div>
      {cartCars &&
        cartCars.map((item) => {
          return <ItemContent key={item.id} item={item} />;
        })
      }
    </div>
    <div className="border-t-[1.5px] border-slate-200 py-4 flex justify-between gap-4">
      <div className="w-[150px]">
        <Button
          label="Очистити список"
          onClick={() => {
            handleClearCart();
          }}
          small
          outline
        />
      </div>
    </div>
  </div>
);

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    </div>
    <div className="text-sm flex flex-col gap-1 items-start">
      <div className="flex justify-between w-full text-base font-semibold">
        <span>Всього:</span>
        <span>{formatPrice(cartTotalAmount)}</span>
      </div>
      <p className="text-slate-500">Бензин включений у вартість</p>
      <Button
        label={currentUser ? "Оплатити" : "Увійдіть для оплати"}
        outline={currentUser ? false : true}
        onClick={() => {
          currentUser ? router.push("/checkout") : router.push("/login");
        }}
      />
      <Link
        href={"/"}
        className="
text-slate-500 flex items-center gap-1
mt-2
"
      >
        <MdArrowBack />
        <span>Вибрати інші авто</span>
      </Link>
    </div>
  </div>
</div>
);
};

export default CartClient;

```

Додаток Г

Файл функції додавання машин для адміністратора.

Лістинг AddCarForm.tsx

```

"use client";

import Button from "@app/components/Button";
import Heading from "@app/components/Heading";
import CategoryInput from "@app/components/inputs/CategoryInput";
import CustomCheckBox from "@app/components/inputs/CustomCheckBox";
import Input from "@app/components/inputs/Input";
import SelectColor from "@app/components/inputs/SelectColor";
import TextArea from "@app/components/inputs/TextArea";
import firebaseApp from "@libs/firebase";
import { categories } from "@utils/Categories";
import { colors } from "@utils/Colors";
import { useCallback, useEffect, useState } from "react";
import { FieldValues, SubmitHandler, useForm } from "react-hook-form";

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import toast from "react-hot-toast";
import {
  getDownloadURL,
  getStorage,
  ref,
  uploadBytesResumable,
} from "firebase/storage";
import axios from "axios";
import { useRouter } from "next/navigation";

export type ImageType = {
  color: string;
  colorCode: string;
  image: File | null;
};

export type UploadedImageType = {
  color: string;
  colorCode: string;
  image: string;
};

const AddCarForm = () => {
  const router = useRouter();
  const [isLoading, setIsLoading] = useState(false);
  const [images, setImages] = useState<ImageType[] | null>();
  const [isCarCreated, setIsCarCreated] = useState(false);

  const {
    register,
    handleSubmit,
    setValue,
    watch,
    reset,
    formState: { errors },
  } = useForm<FieldValues>({
    defaultValues: {
      name: "",
      description: "",
      brand: "",
      category: "",
      inStock: false,
      images: [],
      price: "",
    },
  });

  useEffect(() => {
    setCustomValue("images", images);
  }, [images]);

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

useEffect(() => {
  if (isCarCreated) {
    reset();
    setImages(null);
    setIsCarCreated(false);
  }
}, [isCarCreated]);

const onSubmit: SubmitHandler<FieldValues> = async (data) => {
  console.log("Car Data", data);
  // upload images to fb
  // save car to mongodb
  setIsLoading(true);
  let uploadedImages: UploadedImageType[] = [];

  if (!data.category) {
    setIsLoading(false);
    return toast.error("Category is not selected!");
  }

  if (!data.images || data.images.length === 0) {
    setIsLoading(false);
    return toast.error("No selected image!");
  }

  const handleImageUploads = async () => {
    toast("Додавання авто, зачекайте...");
    try {
      for (const item of data.images) {
        if (item.image) {
          const fileName = new Date().getTime() + "-" + item.image.name;
          const storage = getStorage(firebaseApp);
          const storageRef = ref(storage, `cars/${fileName}`);
          const uploadTask = uploadBytesResumable(storageRef, item.image);

          await new Promise<void>((resolve, reject) => {
            uploadTask.on(
              "state_changed",
              (snapshot) => {
                const progress =
                  (snapshot.bytesTransferred / snapshot.totalBytes) * 100;
                console.log("Upload is " + progress + "% done");
                switch (snapshot.state) {
                  case "paused":
                    console.log("Upload is paused");
                    break;
                  case "running":
                    console.log("Upload is running");
                    break;
                }
              }
            ),
              resolve,
              reject
            );
          });
        }
      }
    },
  },

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        (error) => {
            console.log("Error uploading image", error);
            reject(error);
        },
        () => {
            getDownloadURL(uploadTask.snapshot.ref)
                .then((downloadURL) => {
                    uploadedImages.push({
                        ...item,
                        image: downloadURL,
                    });
                    console.log("File available at", downloadURL);
                    resolve();
                })
                .catch((error) => {
                    console.log("Error getting the download URL", error);
                    reject(error);
                });
        }
    );
});
}
} catch (error) {
    setIsLoading(false);
    console.log("Error handling image uploads", error);
    return toast.error("Error handling image uploads");
}
};

await handleImageUploads();
const carData = { ...data, images: uploadedImages };

axios
    .post("/api/car", carData)
    .then(() => {
        toast.success("Авто додане");
        setIsCarCreated(true);
        router.refresh();
    })
    .catch((error) => {
        toast.error("Помилка запису авто у БД");
    })
    .finally(() => {
        setIsLoading(false);
    });
};

const category = watch("category");

const setCustomValue = (id: string, value: any) => {

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    setValue(id, value, {
      shouldValidate: true,
      shouldDirty: true,
      shouldTouch: true,
    });
  });
};

const addImageToState = useCallback((value: ImageType) => {
  setImages((prev) => {
    if (!prev) {
      return [value];
    }

    return [...prev, value];
  });
}, []);

const removeImageFromState = useCallback((value: ImageType) => {
  setImages((prev) => {
    if (prev) {
      const filteredImages = prev.filter(
        (item) => item.color !== value.color
      );
      return filteredImages;
    }

    return prev;
  });
}, []);

return (
  <>
    <Heading title="Додавання авто" center />
    <Input
      id="name"
      label="Назва"
      disabled={isLoading}
      register={register}
      errors={errors}
      required
    />
    <Input
      id="price"
      label="Ціна/год."
      disabled={isLoading}
      register={register}
      errors={errors}
      type="number"
      required
    />
    <Input

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        id="brand"
        label="Марка"
        disabled={isLoading}
        register={register}
        errors={errors}
        required
    />
    <TextArea
        id="description"
        label="Опис"
        disabled={isLoading}
        register={register}
        errors={errors}
        required
    />
    <CustomCheckBox
        id="inStock"
        register={register}
        label="Доступне для оренди"
    />
    <div className="w-full font-medium">
        <div className="mb-2 font-semibold">Виберіть тип авто</div>
        <div className="grid grid-cols-2 md:grid-cols-3 gap-3 max-h[50vh] overflow-
y-auto">
            {categories.map((item) => {
                if (item.label === "Усе") {
                    return null;
                }

                return (
                    <div key={item.label} className="col-span">
                        <CategoryInput
                            onClick={({category}) => setCustomValue("category", category)}
                            selected={category === item.label}
                            label={item.label}
                            icon={item.icon}
                        />
                    </div>
                );
            })}
        </div>
    </div>
    <div className="w-full flex flex-col flex-wrap gap-4">
        <div>
            <div className="font-bold">
                Виберіть доступні кольори автомобіля та завантажте зображення.
            </div>
            <div className="text-sm">
                Необхідно завантажити зображення для кожного вибраного кольору,
                інакше колір буде проігноровано.
            </div>
        </div>
    </div>

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    </div>
    <div className="grid grid-cols-2 gap-3">
      {colors.map((item, index) => {
        return (
          <SelectColor
            key={index}
            item={item}
            addImageToState={addImageToState}
            removeImageFromState={removeImageFromState}
            isCarCreated={isCarCreated}
          />
        );
      })}
    </div>
  </div>
  <Button
    label={isLoading ? "Завантаження..." : "Додати авто"}
    onClick={handleSubmit(onSubmit)}
  />
</>
);
};

export default AddCarForm;

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		