

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота
на тему: «Розробка вебсистеми бронювання автомобілів для мандрівників»

Виконав: студент 4 курсу групи 1ПІ-216
спеціальності 121 – Інженерія програмного
забезпечення

(шифр і назва напрямку підготовки, спеціальності)

Мельничук М. В.

(прізвище та ініціали)

Керівник: к.т.н., ст. викладач Стахов О. Я.

(прізвище та ініціали)

Рецензент: к.т.н., проф. Захарченко С. М.

(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ

«09» 06 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
« 21 » березня 2025 р.

ЗАВДАННЯ НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Мельничуку Максиму Васильовичу

1. Тема роботи – Розробка вебсистеми бронювання автомобілів для мандрівників.

Керівник роботи: Стахов Олексій Ярославович, доктор Філософії PhD, сташ. викл., затверджені наказом вищого навчального закладу від 20 березня 2025 р. № 97.

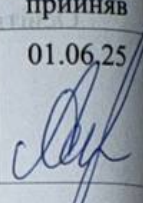
2. Строк подання студентом роботи 2 червня 2025.

3. Вихідні дані до роботи: середовище розробки Visual Studio Code, мова розробки TypeScript, Windows 11 23H2, алгоритми бронювання автомобілів та обробки даних про транспортні засоби.

4. Зміст текстової частини: вступ; аналіз розвитку вебсистем бронювання автомобілів для мандрівників та постановка задач розробки, розробка моделей та алгоритмів програмного продукту, розробка програмних модулів реалізації мобільного застосунку, тестування застосунку; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу: приклади аналогів; модель роботи застосунку; структурна схема застосунку; блок-схеми алгоритмів роботи застосунку; графічний інтерфейс застосунку; тестування застосунку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Стахов О.Я., к.т.н, ст. викладач	24.03.25 	01.06.25 

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

Ч.ч.	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітк а
1	Аналіз стану питання та постановка задачі	25.03.25 – 31.03.25	Вик
2	Розробка моделей, структури та алгоритмів	1.04.25 – 10.04.25	Вик
3	Розробка програмного забезпечення	11.04.25 – 28.04.25	Вик
4	Тестування застосунку	29.04.25 – 15.05.25	Вик
5	Оформлення матеріалів до захисту БКР	16.05.25 – 30.05.25	Вик

Студент

Керівник бакалаврської кваліфікаційної роботи


(підпис)

Мельничук М. В.
(ініціали та прізвище)


(підпис)

Стахов О. Я.
(ініціали та прізвище)

У,
М

мандрі
програ
забезп
рис. : 8

У

для бр

реаліза

У

для ма

та фі

корист

процес

1

програ

Mongo

Для за

сучасн

створе

TypeS

АНОТАЦІЯ

УДК 004.4

Мельничук М. В. Розробка вебсистеми бронювання автомобілів для мандрівників: бакалаврська кваліфікаційна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – Інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 114 с. На укр. мові. Бібліогр. : 23 назв; рис. : 81; табл. 9.

У бакалаврській кваліфікаційній роботі проведено аналіз стану вебсистем для бронювання автомобілів, доведено доцільність розробки власної програмної реалізації, зважаючи на порівняльний аналіз аналогів.

У ході виконання роботи розроблено вебсистему бронювання автомобілів для мандрівників, що включає в себе модулі для створення, перегляду, пошуку та фільтрації доступних автомобілів, а також графічного інтерфейсу користувача. Система призначена для спрощення та підвищення ефективності процесу бронювання автомобілів для подорожей.

Вебсистема розроблена з використанням фреймворку Next.js, мови програмування TypeScript, серверного середовища Node.js, бази даних MongoDB, бібліотеки React, а також середовища розробки Visual Studio Code. Для забезпечення зручного та інтуїтивно зрозумілого інтерфейсу використано сучасні бібліотеки та інструменти.

Розроблені програмні модулі можуть бути використані розробниками для створення різних вебсистем бронювання.

Ключові слова: вебсистема бронювання, автомобілі, мандрівники, Next.js, TypeScript, Node.js, MongoDB, React.

ANNOTATION

UDC 004.4

Melnychuk M.V. Development of the web system of booking cars for travelers: bachelor's thesis in specialty 121 Software Engineering, educational program - Software Engineering. Vinnitsa: VNTU, 2024. 114 s. in Ukrainian. Bibliogram.: 22 titles; Figure: 27; Table 2.

In the bachelor's thesis, an analysis of the state of websystems for booking cars was carried out, the feasibility of developing their own software implementation was proved, due to the comparative analysis of analogues.

In the course of the work, a web-based car booking system for travelers has been developed, which includes modules for creating, viewing, searching and filtering available cars, as well as a graphical user interface. The system is designed to simplify and increase the efficiency of the process of booking cars for travel.

The web system was developed using the Next.js framework, the TypeScript programming language, the Node.js server environment, the MongoDB database, the React library, and the Visual Studio Code development environment. Modern libraries and tools are used to provide a convenient and intuitive interface.

The developed software modules can be used by developers to create various booking web systems.

Keywords: web booking system, cars, travelers, Next.js, TypeScript, Node.js, MongoDB, React.

ЗМІСТ

ВСТУП	3
1 АНАЛІЗ РОЗВИТКУ ВЕБСИСТЕМ БРОНЮВАННЯ АВТОМОБІЛІВ ДЛЯ МАНДРІВНИКІВ ТА ПОСТАНОВКА ЗАДАЧ РОЗРОБКИ.....	6
1.1 Аналіз стану вебсистем бронювання автомобілів для мандрівників	6
1.2 Порівняльний аналіз аналогів	7
1.3 Аналіз методів розв’язання задачі	13
1.4 Постановка задачі.....	14
1.5 Висновки	15
2 РОЗРОБКА МОДЕЛЕЙ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ....	16
2.1 Аналіз вхідних даних системи	16
2.2 Розробка моделі системи	18
2.4 Розробка алгоритму захисту даних платіжної картки у вебсистемі.....	21
2.5 Висновки	24
3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ ВЕБСИСТЕМИ	25
3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення.....	25
3.2 Розробка модуля фільтрації автомобілів у каталозі	27
3.3 Розробка модуля оформлення бронювання автомобіля.....	31
3.4 Розробка інтерфейсу	38
3.5 Висновки	46
4 ТЕСТУВАННЯ ЗАСТОСУНКУ	48
4.1 Тестування системи	48
4.2 Розробка інструкції користувача.....	54
4.3 Висновки	56
ВИСНОВКИ.....	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	59
ДОДАТОК А (обов’язковий). Технічне завдання	61
ДОДАТОК Б (обов’язковий). Протокол перевірки на плагіат	64
ДОДАТОК В (довідниковий). Лістинг програмного коду	67
ДОДАТОК Г (обов’язковий). Ілюстративна частина	102

ВСТУП

Обґрунтування вибору теми дослідження. Сучасний туристичний світ та мандри вимагають ефективних інструментів для планування та бронювання транспорту. Збільшення автомобільних подорожей підкреслює важливість зручних вебсистем, що надають користувачам швидкий та безпечний спосіб бронювання автомобілів. Проте, наявні рішення часто не відповідають потребам мандрівників, адже не враховують особливі вимоги до зручності користування, опцій фільтрації та захисту інформації [1].

У відповідь на ці виклики, стає актуальною розробка програмного забезпечення вебсистеми, яка б спеціально відповідала потребам мандрівників. Ця система значно полегшить процес бронювання автомобілів, надаючи автоматизований пошук доступних автомобілів, інтуїтивну зрозумілу навігацію, фільтрацію за критеріями та збереження інформації про бронювання.

Розробка подібної вебсистеми становить собою ключовий етап на шляху покращення до полегшення доступу до транспортних засобів для мандрівників [2]. Розробка надасть у користування засіб для бронювання автомобілів, що сприятиме зростанню ефективності планування подорожей та комфорту для користувачів. Така система сформує сприятливе середовище для оперативного вибору автобіля та оптимального маршруту, що є надзвичайно важливим аспектом для туристів та мандрівників.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є збільшення ефективності та зручності процесу бронювання автомобіля для мандрівників, шляхом розробки вебсистеми бронювання транспортних засобів. Система полегшить оптимальний вибір автомобіля, надаючи швидкий доступ до актуальної інформації та зручне оформлення бронювання

Основними задачами дослідження є:

- розробка бази даних для зберігання автомобілів та бронювання;
- розробка інтерфейсу користувача, що забезпечить зручну взаємодію з застосунком та легкий доступ до необхідної інформації;
- розробка модуля пошуку та фільтрації автомобілів за параметрами (обладнання, ціна, місто тощо);
- розробка модуля оформлення бронювання.

Об’єкт дослідження – процес автоматизованого бронювання автомобілів через вебсайт.

Предмет дослідження – методи та засоби розробки програмних модулів для веб-систем бронювання автомобілів.

Методи дослідження. У процесі розробки вебсистеми були застосовані підходи створення вебзастосунків з використанням фреймворку Next.js, що працює з мовою TypeScript. Такий підхід гарантує високу швидкість роботи та зручність в розробці сучасних вебсистем з динамічним контентом. Серверна частина реалізована на Node.js з використанням бази даних MongoDB, що забезпечує ефективне зберігання та опрацювання інформації про транспортні засоби та їх бронювання. Архітектура REST API полегшує доступ до даних з клієнтської частини застосунку. Для розробки візуальної частини було використано бібліотеку React, яка дала змогу створити адаптивний дизайн та інтуїтивно зрозумілий інтерфейс для користувачів. Тестування вебсистеми було проведено для контролю коректності функціонування основних модулів та зручності взаємодії, що дозволило виявити та виправити можливі помилки ще на етапі розробки.

Новизна отриманих результатів.

1. Подальшого розвитку зазнав метод бронювання автомобілів через вебсистему, що відрізняється від наявних реалізацією інтуїтивного інтерфейсу для вибору машини та завершення бронювання. Запропоноване рішення дає змогу мандрівникам без зусиль знаходити та замовляти транспортні засоби, враховуючи різноманітні критерії, що покращує зручність взаємодії з системою. Розроблена вебсистема автоматизує процес

бронювання, пропонуючи швидкий пошук автомобіля за допомогою фільтрів та зрозуміле оформлення замовлення.

2. Подальшого розвитку набув метод створення вебзастосунків для бронювання автомобілів, у якому застосовано технології: Next.js, TypeScript, Node.js, MongoDB та React. Це дозволило досягти високої продуктивності системи, адаптивності інтерфейсу та ефективної обробки великої кількості інформації про автомобіль.

Практична цінність отриманих результатів. Практична цінність здобутих результатів полягає у можливості застосування теоретичних положень, отриманих у бакалаврській кваліфікаційній роботі. Запропоновані алгоритми та розроблені програмні засоби вебсистеми для бронювання автомобілів, можуть бути використані туристичними агентствами та сервісами прокату для забезпечення швидкого й зручного процесу бронювання автомобілів.

Особистий внесок здобувача. Усі наукові результати, що містяться у бакалаврській кваліфікаційній роботі, отримані автором особисто. Здобувач розробив задум вебсистеми для бронювання автомобілів, здійснив аналіз існуючих аналогів, спроектував архітектуру системи та написав програмні модулі для втілення ключових функцій, таких як пошук, фільтрація та бронювання транспортних засобів. Додатково, автор самостійно протестував систему та покращив її ефективність.

Апробація матеріалів бакалаврської кваліфікаційної роботи. Основні положення бакалаврської кваліфікаційної роботи доповідалися та обговорювалися на внутрішньому семінарі кафедри програмного забезпечення Вінницького національного технічного університету. Результати роботи отримали позитивну оцінку та рекомендації щодо впровадження у сфері туристичних послуг та сервісів оренди автомобілів.

Публікації. Основні результати досліджень не опубліковані у наукових працях.

1 АНАЛІЗ РОЗВИТКУ ВЕБСИСТЕМ БРОНЮВАННЯ АВТОМОБІЛІВ ДЛЯ МАНДРІВНИКІВ ТА ПОСТАНОВКА ЗАДАЧ РОЗРОБКИ

1.1 Аналіз стану вебсистем бронювання автомобілів для мандрівників

Сучасний етап розвитку інформаційних технологій вимагає постійного вдосконалення методів обробки даних та взаємодії між користувачами, зокрема у веб-сфері онлайн-сервісів. Однією з найбільш динамічних та популярних галузей є веб-системи бронювання автомобілів для мандрівників. Розвиток цієї сфери зумовлений потребою в автоматизації процесів оренди, зручності для користувачів і можливості швидкого доступу до актуальної інформації про автомобілів.

Використання вебтехнологій у системах бронювання автомобілів для мандрівників дозволяє оптимізувати процес пошуку, вибору та бронюванню транспорту. Сучасні веб-системи забезпечують користувачам можливість переглядати доступні автомобілі, порівнювати їх характеристики, ознайомлюватися з умовами оренди та оформлювати бронювання в режимі реального часу, що створює зручний та швидкий спосіб взаємодії між користувачами і компанією-орендодавцем.

Одним із ключових аспектів розвитку таких веб-систем є швидкість і надійність обробки даних. Завдяки використанню потужних серверів і сучасних технологій, таких як Node.js, Express і MongoDB, забезпечується стабільна робота сервісу навіть за великої кількості запитів. Важливу роль відіграє також інтеграція з різноманітними API, яка дозволяє розширювати функціональність системи, наприклад, через підключення платіжних систем або сервіси геолокації.

Сучасні вебдодатки для бронювання автомобілів мають високий рівень автоматизації процесів, що дозволяє мінімізувати ручну роботу завдяки повністю цифровізованому процесу оренди. Вони забезпечують швидкий доступ до актуальних даних про наявність автомобілів і вартість оренди, що дає змогу користувачам оперативно приймати рішення. Зручність користування

забезпечується адаптивним інтерфейсом, доступним із будь-якого пристрою, включаючи мобільні телефони та планшети.

Використання вебтехнологій також забезпечує точність і актуальність даних. Завдяки інтеграції з базами даних у реальному часі будь-які зміни в наявності або умовах оренди автомобіля моментально відображаються на сайті. Це підвищує рівень довіри користувачів до системи та створює позитивний користувацький досвід [3].

Отже, аналіз стану вебтехнологій у сфері бронювання автомобілів підтверджує важливість впровадження сучасних рішень, що забезпечують швидку, зручну та надійну взаємодію користувача із системою. Розробка веб-застосунку для бронювання автомобілів з використанням передових технологій стає необхідністю для покращення якості послуг і підвищення конкурентоспроможності компаній. У наступних розділах будуть детально розглянуті вимоги до системи, методи її реалізації та особливості програмної архітектури.

1.2 Порівняльний аналіз аналогів

Використання веб-систем для бронювання автомобілів стає все більш популярним завдяки зручності, швидкості та доступності таких сервісів. Сучасні рішення забезпечують ефективність взаємодії між клієнтом і сервісом оренди, автоматизують процес бронювання та обробки замовлень, а також мінімізують ризик помилок під час введення і збереження даних.

Розробка власного програмного продукту вимагає ретельного аналізу вже існуючих рішень, щоб виявити їхні сильні та слабкі сторони, визначити ключові функції та знайти можливості для вдосконалення. До найбільш відомих платформ, що надають послуги оренди автомобілів онлайн, належать: Rentalcars, Hertz, Avis, Localrent та EconomyBookings.

Rentalcars – один із найбільших міжнародних сервісів оренди автомобілів. Платформа пропонує широкий вибір автомобілів у різних країнах, інтегруючись із багатьма локальними та глобальними компаніями [4]. Веб-застосунків має

зручний і простий інтерфейс, систему фільтрації за параметрами (тип автомобіля, вартість, місцезнаходження), а також можливість миттєвого бронювання. Водночас серед недоліків можна відзначити часті затримки в оновленні інформації про наявність транспортних засобів, що може призводити до помилок у бронюванні (див. рисунок 1).

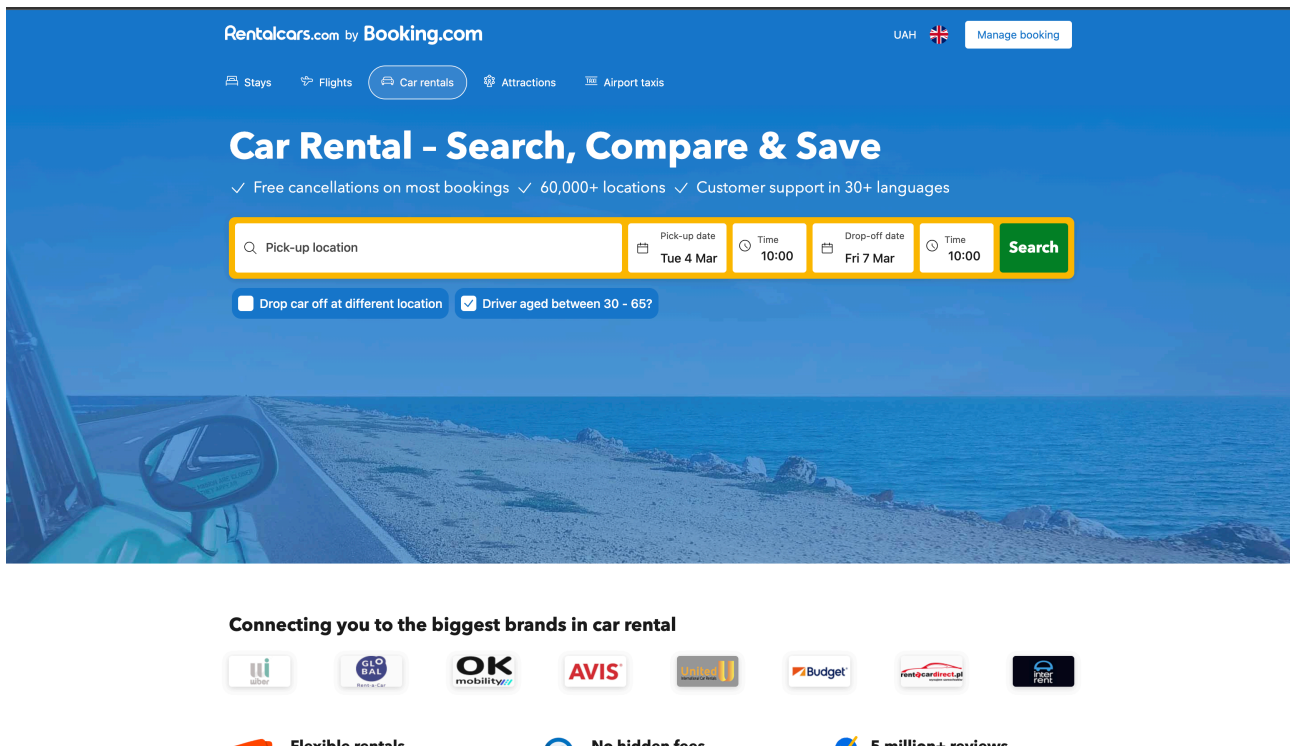


Рисунок 1.1 — Приклад інтерфейсу застосунку Rentalcars

TheRentalRadar — міжнародна компанія з багаторічним досвідом, що пропонує оренду автомобілів через веб-платформу [5]. Відмінною рисою є широкий спектр додаткових послуг, таких як страхування, оренда з водієм, дитячі крісла та навігаційні пристрої. Платформа забезпечує точність інформації та швидкість обробки замовлень. Водночас інтерфейс веб-сайту потребує оновлення, оскільки не завжди відповідає сучасним вимогам до зручності та адаптивності (див. рисунок 2).

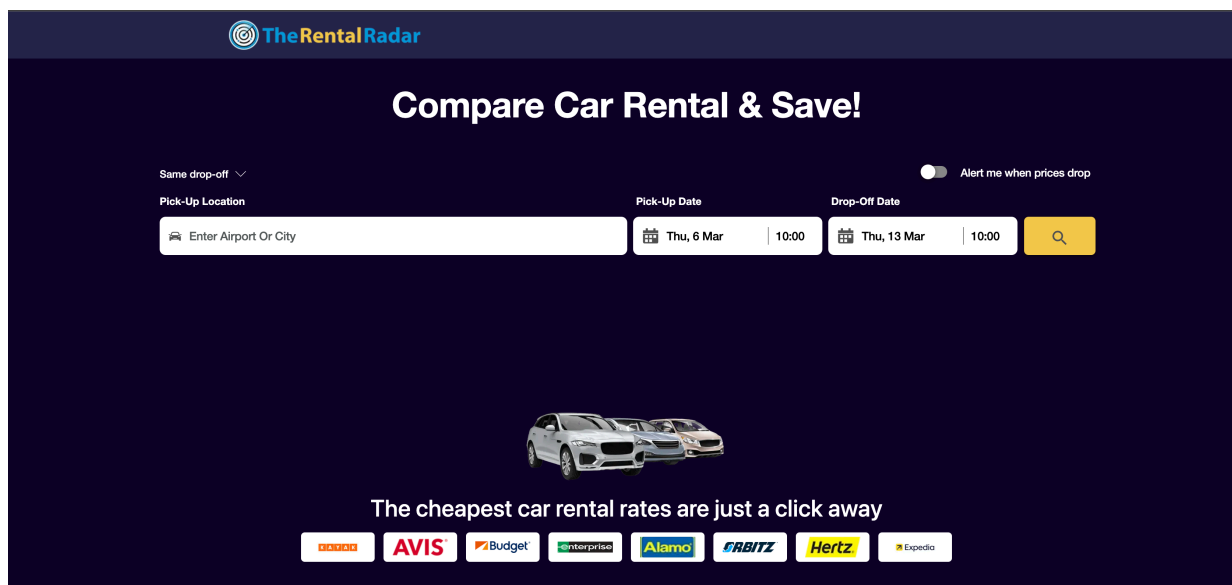


Рисунок 1.2 — Приклад інтерфейсу застосунка TheRentalRadar

Avis – ще один популярний міжнародний сервіс, що надає послуги оренди автомобілів (див. рисунок 1.3). Платформа пропонує гнучку систему знижок і програм лояльності, що робить її привабливою для постійних клієнтів. Зручний мобільний додаток дозволяє користувачам легко здійснювати бронювання, відстежувати статус замовлення та вносити зміни. Однак швидкість обробки запитів може бути нижчою порівняно з конкурентами [6].

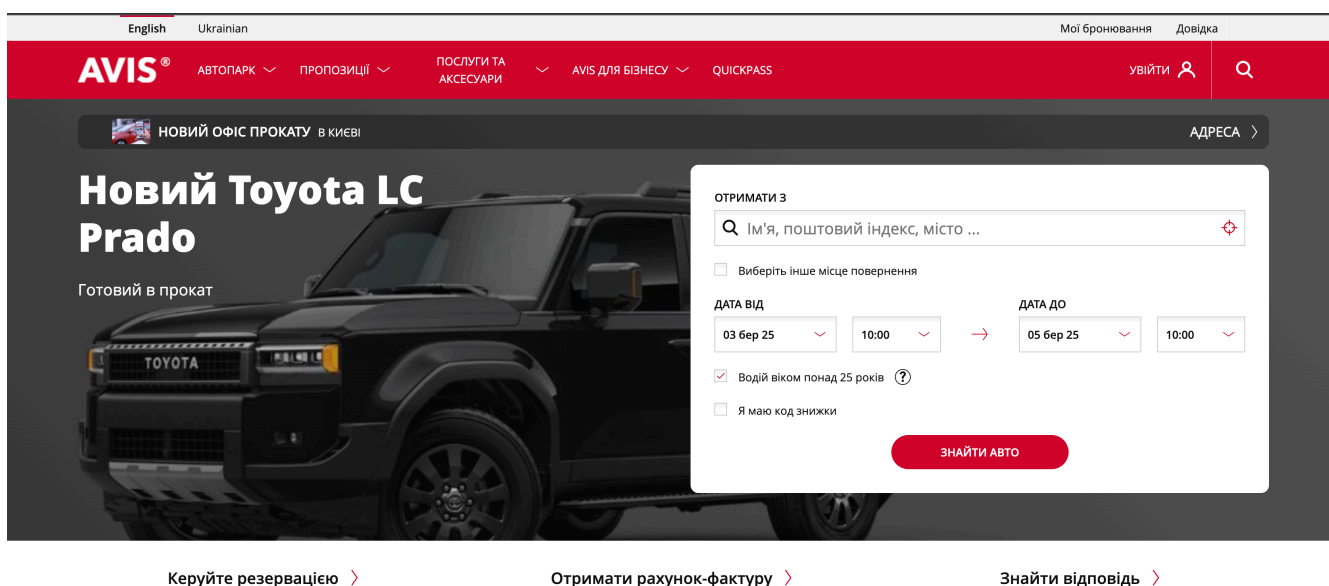


Рисунок 1.3 – Приклад інтерфейсу застосунку Avis

EconomyBookings – глобальна платформа, що спеціалізується на наданні недорогих варіантів оренди автомобілів (див. Рисунок 1.4). Веб-сайт пропонує різноманітні варіанти транспорту за конкурентоспроможними цінами. Платформа інтегрує різні платіжні системи, забезпечуючи зручність оплати. Проте деякі користувачі відзначають проблеми з поверненням коштів у разі скасування бронювання [7].

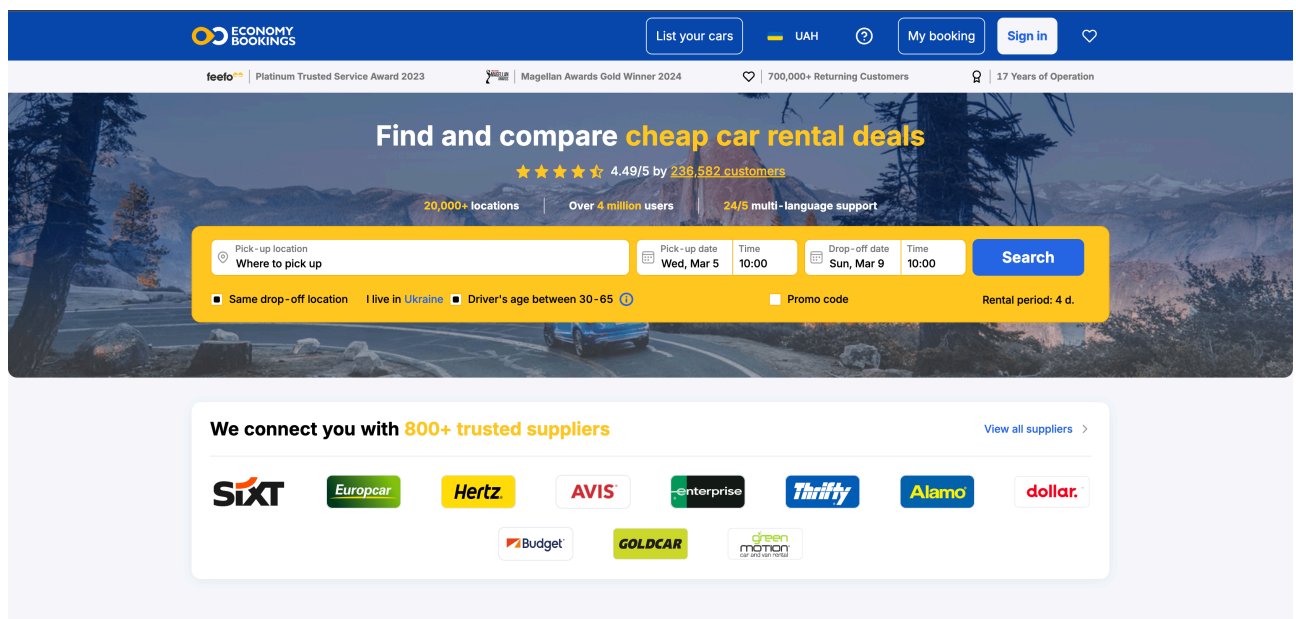


Рисунок 1.4 – Приклад інтерфейсу застосунку EconomyBookings

Localrent – локальний сервіс, орієнтований на надання автомобілів у конкретних регіонах (див. рисунок 1.5). Основною перевагою є доступна ціна та персоналізований підхід до клієнтів. Водночас платформа часто поступається міжнародним аналогам у кількості доступних автомобілів, стабільності роботи системи та рівні підтримки користувачів [8].

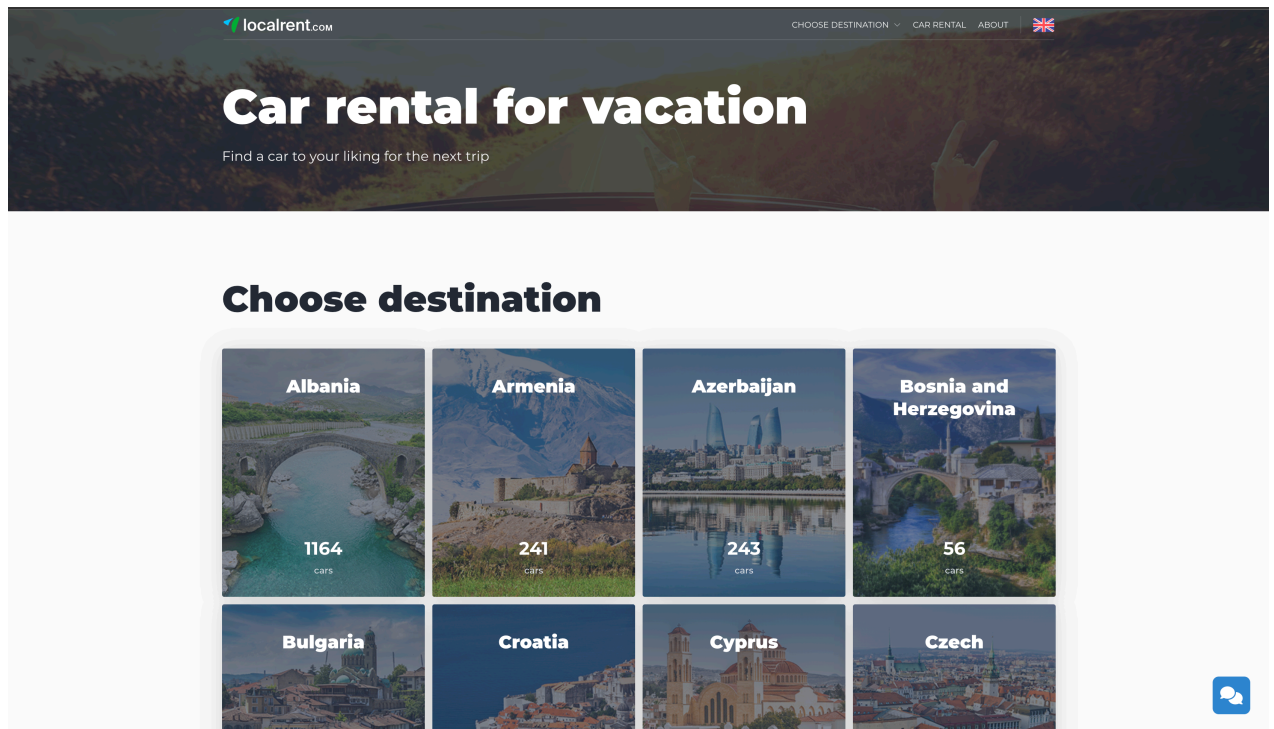


Рисунок 1.5 – Приклад інтерфейсу застосунку Localrent

Розробка веб-системи для оренди автомобілів потребуватиме глибоких знань у галузі веб-програмування, проектування баз даних і створення зручного користувацького інтерфейсу. Це означає створення внутрішньої системи для ефективного керування автомобільною базою даних, обробки бронювань запитів та збереження інформацій про користувачів та їх замовлення. Важлива частина застосунка – розробка користувацького інтерфейсу з легким функціоналом та можливостями при цьому бути адаптивним та легко доступно до всіх опцій системи.

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 — Порівняльна характеристика аналогів

Критерії	Rentalcars	TheRental Radar	Avis	Economy Bookings	Localrent	Власна розробка
Архітектура REST API	-	+	+	-	-	+

Продовження таблиці 1.1

Критерії	Rentalcars	TheRental Radar	Avis	Economy Bookings	Localrent	Власна розробка
Швидкість обробки замовлень	+	+	+	-	-	+
Автоматизація процесу бронювання	+	+	+	-	-	+
Використання бази даних (MongoDB, SQL)	+	-	+	-	-	+
Шифрування даних (SSL, HTTPS)	+	+	-	+	-	+
Загальна	4	4	4	1	0	5

Відповідно до таблиці порівняльних характеристик розробка власних програмних засобів для бронювання автомобілів для мандрівників є доцільною. Отриманий продукт зможе покрити недоліки існуючих веб-систем та забезпечити розширені можливості для бронювання автомобілів мандрівникам.

Отже, розробка програмних засобів бронювання автомобілів для мандрівників відкриває широкі можливості для оптимізації процесу оренди транспортних засобів. Запропонований підхід забезпечує не лише зручність та швидкість обробки запитів, а підвищує актуальність інформації про доступні автомобілі та умови їх оренди. Розроблений продукт дозволяє суттєво знизити

ризик помилок під час бронювання, скоротити час обробки запитів та покращити взаємодію користувачів із сервісом, що є важливим чинником підвищення конкурентності компаній, що надають послуги оренди авто.

1.3 Аналіз методів розв’язання задачі

Створення веб-системи для бронювання автомобілів для мандрівників потребує складного підходу до розв’язання завдань щодо забезпечення зручного та надійного сервісу шляхом автоматизацію процесів оренди авто та надання доступу до актуальної інформації про транспортні засоби; де важлива простота користувацького інтерфейсу та безпеки обробки персональних даних.

Одним із найпоширеніших підходів у розробці програмного застосунка є класична клієнт-серверна архітектура. Цей підхід передбачають чітку відокремлення між фронтендом (користувацькою частиною) та бекендом (серверною частиною). Фронтенд відповідає за інтерфейс користувача та його взаємодію з програмою - вибір автомобіля, перегляд характеристик та оформлення бронювання. Бекенд же забезпечує обробку запитів, роботу з базою даних та виконання бізнес-логіки. Основною перевагою такого підходу є висока продуктивність та стабільність системи. Також цей метод потребує уважного проектування API та оптимізацій запитів до сервера.

Іншим сучасним підходом є використання SPA (Single Page Application) — веб-застосунків, які працюють як одна веб-сторінка, динамічно змінюючи контент без повного перезавантаження. SPA забезпечує значно вищу швидкість роботи програми, зменшує навантаження на сервер і покращує користувацький досвід завдяки плавності взаємодії. Для створення SPA зазвичай використовуються сучасні фреймворки, такі як React, Angular або Vue.js. Проте цей підхід має певні недоліки, зокрема складність SEO-оптимізації та початкове завантаження великого обсягу даних.

У контексті обробки даних ефективним методом є використання REST API разом з базою даних MongoDB. REST API дозволяють швидку взаємодію між користувачем та сервером за допомогою стандартизованих HTTP-запитів

(GET, POST, PUT, DELETE), тоді як MongoDB як NoSQL база даних забезпечуватиме гнучке збереження інформацій про користувачів авто та бронювання у форматі JSON для вашо зручності. Основною перевагою такого підходу щд простота інтеграцій та можливість масштабування системи. Однак має недолік - валідація даних та захисту приватності [].

Ще одним перспективним методом є використання GraphQL, який забезпечує оптимізацію запитів до API, дозволяючи клієнту отримувати лише необхідні дані одним запитом. У системі бронювання автомобілів це дозволяє зменшити кількість запитів і мінімізувати трафік, підвищуючи швидкість роботи програми. Водночас реалізація GraphQL вимагає більше часу на налаштування і є складнішою в обробці запитів [].

Розглянувши унікальність кожного з методів, ефективним є використання комбінованого підходу, який опирається на швидкість, гнучкість і масштабованість в собі дозволить створити веб-платформу для бронювання автомобілей з особливим наголосом на високу продуктивність та інтуїтивний інтерфейс для користувача разом з обробкою даних. Це сприятиме покращенню якості обслуговування та полегшить мандрівникам користуватися сервісом.

1.4 Постановка задачі

Проаналізувавши переваги та недоліки існуючих веб-систем для бронювання автомобілів, було визначено основні завдання, які необхідно виконати для успішної розробки програмного забезпечення:

- розробити алгоритм створення та управління базою даних автомобілів, що міститиме інформацію про доступні транспортні засоби, їх характеристики, статус і вартість оренди;
- розробити алгоритм пошуку та фільтрації автомобілів за різними параметрами (марка, модель, ціна, клас, місцезнаходження), що забезпечить швидкий та зручний доступ до необхідної інформації.
- реалізувати функціонал реєстрації та авторизації користувачів із можливістю збереження особистих даних і перегляду історії бронювань.

- розробити модуль бронювання автомобілів із вибором дати, часу та місця отримання й повернення транспортного засобу.
- створити зручний та інтуїтивно зрозумілий графічний інтерфейс, орієнтований на потреби мандрівників, що забезпечить комфортну взаємодію з веб-системою.
- автоматизувати процес обробки замовлень, включаючи підтвердження бронювання, зміну або скасування заявки.
- забезпечити високий рівень безпеки даних користувачів і фінансових операцій через інтеграцію з надійними платіжними системами.
- провести тестування веб-системи на відповідність функціональним вимогам, стабільність роботи та зручність використання.

В результаті, виконання цих завдань дозволить створити високоякісний вебсистему, яка відповідає потребам сервісам бронювання автомобіля.

1.5 Висновки

У першому розділі було проаналізовано стан існуючих веб-систем для бронювання автомобілів, таких як Rentalcars, DiscoverCars, EconomyBookings, Hertz та Avis. Кожна з цих систем має свої переваги та недоліки, що були детально розглянуті.

Проаналізувавши наявні рішення, було сформульовано завдання для подальшої розробки власної веб-системи бронювання автомобілів, орієнтованої на потреби мандрівників. Основні завдання включають створення бази даних транспортних засобів, реалізацію функціоналу пошуку, фільтрації та бронювання автомобілів, розробку зручного графічного інтерфейсу, забезпечення безпеки даних і фінансових операцій.

Таким чином, було підтверджено доцільність розробки власної веб-системи з унікальними можливостями та зручним функціоналом. На основі отриманих даних було сформовано перелік завдань, необхідних для реалізації програмного продукту, що забезпечить високу якість послуг з бронювання автомобілів.

2 РОЗРОБКА МОДЕЛЕЙ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних даних системи

Аналіз вхідних даних для системи – це ключовий етап у розробці вебсистеми бронювання автомобілів для мандрівників. Цей процес передбачає уважне дослідження та розуміння різноманітних видів інформації, що можуть надходити до системи в процесі її використання.

Основні вхідні дані, які надходять до вебсистеми, складаються з параметрів пошуку та сортування автомобілів, введені користувачем. Ці налаштування можуть містити: населений пункт оренди, вимоги до оснащення (кондиціонування, автоматична трансмісія, GPS, автокрісло) і відбір за вартістю. Ці відомості застосовуються для формування грамотного запиту до сховища даних та візуалізації автомобілів, що задовольняють заданим вимогам. Отже, коректність та повнота вказаних параметрів мають безпосередній вплив на релевантність пошуку та відображення відповідних пропозицій.

Ще одним важливим різновидом даних, що надходять, є дані бронювання. Вони містять особисту інформацію клієнта: ім'я, номер телефону, електронну пошту. Також вказуються дати початку та завершення оренди. До того ж, клієнт може додати свої побажання, як-от додаткове обладнання чи супутні послуги. Окрім цього, форма бронювання передбачає вибір способу оплати. Користувач має змогу обрати з кількох запропонованих варіантів, наприклад: оплата готівкою при отриманні машини, онлайн-платіж банківською карткою або через електронні платіжні системи. Збереження цих даних у базі даних критично важливе для коректної обробки замовлення та уникнення можливих проблем при передачі автомобіля. Більше того, надійне зберігання даних є ключовим елементом безпеки системи, особливо якщо йдеться про фінансову інформацію.

Для гарантування коректного введення інформації використовується валідація з боку клієнта та сервера. Скажімо, коли дата закінчення оренди передуює даті старту, система пропонує виправити дані. Подібним чином, під час введення контактних даних (номер телефону, електронна пошта) здійснюється

перевірка їхнього формату. Якщо виникає помилка, користувач отримує повідомлення з порадою щодо виправлення, що зменшує можливість помилкового бронювання та позитивно впливає на взаємодію з системою.

Окрему увагу варто зосередити на обробці непередбачуваних обставин та неправильних запитів. Скажімо, якщо в ході бронювання виникає інформація про відсутність обраної машини на зазначену дату, система має надати альтернативні варіанти або надати можливість змінити параметри пошуку. Подібні механізми скорочують кількість скасування бронювань і покращують рівень задоволення клієнтів.

Специфікою системи є також опрацювання даних користувачів після успішного бронювання. Успішно оформлена заявка передбачає збереження всіх внесених даних до бази та надсилання електронного повідомлення з підтвердженням на вказану електронну адресу. У випадку виявлення помилки під час збереження інформації, система автоматично інформує про це та пропонує спробувати знову або звернутися до служби підтримки.

Аналіз вхідних даних включає також обробку інформації про відгуки та оцінки автомобілів. Користувачі мають можливість залишити свої коментарі та оцінити авто після завершення оренди. Ці дані використовуються для формування рейтингу автомобілів та поліпшення якості послуг. Відгуки, що містять ненормативну лексику або невірну інформацію, піддаються фільтрації за допомогою автоматизованої модерації.

У підсумку, розбір вхідної інформації сприяє усвідомленню потреб користувачів, оптимізації роботи з формами та гарантії стабільності системи. Це ключовий етап у процесі створення веб-застосунку, адже від коректної обробки вхідних відомостей залежить точність відображення результатів пошуку та ефективність процедури бронювання. Рациональне структурування даних дозволяє збільшити продуктивність системи та забезпечити її надійне функціонування в різних умовах взаємодії з користувачами.

2.2 Розробка моделі системи

Для глибшого розуміння логіки роботи веб-системи бронювання автомобілів для мандрівників була створено UML-діаграму діяльності, яка демонструє ключові етапи взаємодії користувача із програмним застосунком (див. рисунок 2.1). Такий підхід дає можливість візуалізувати процеси, що відбуваються під час користування веб-системою, та визначити ключові етапи, на яких можливі помилки або потрібна додаткова обробка даних.

Робота з системою стартує із завантаження головної сторінки, де користувачеві відкривається огляд доступного функціоналу та кнопка для переходу до каталогу авто. Інтерфейс створено з акцентом на комфорт, аби базові операції, такі як перегляд автомобілів, сортування та вибір, були інтуїтивно зрозумілими та швидкими.

Перейшовши до каталогу, користувач має можливість використовувати форму пошуку, вказуючи місто, тип автомобіля чи інші опції, наприклад, наявність кондиціонера або дитячого автокрісла. Система аналізує введені дані, фільтруючи пропозиції. У разі відсутності відповідних варіантів відображається повідомлення про відсутність результатів.

У разі, якщо користувач знайшов потрібний автомобіль, він може переглянути деталі на окремій сторінці: фотографії, опис, ціну, технічні характеристики та відгуки інших клієнтів. Це дає можливість прийняти обґрунтоване рішення перед бронюванням.

Обравши автомобіль, користувач потрапляє до форми бронювання, де заповнює особисті дані, обирає дати оренди та метод оплати. Перед відправкою заявка проходить перевірку на коректність, а у випадку помилок система інформує про необхідність виправлення. Якщо всі дані вказані правильно, користувач підтверджує бронювання. Система здійснює фінальну перевірку наявності авто на вибрані дати і, якщо все гаразд, відображає підтвердження бронювання.

Підсумовуючи, розроблена модель дозволяє оптимізувати взаємодію з користувачем, мінімізуючи ризик помилок та підвищуючи зручність використання веб-сервісу.

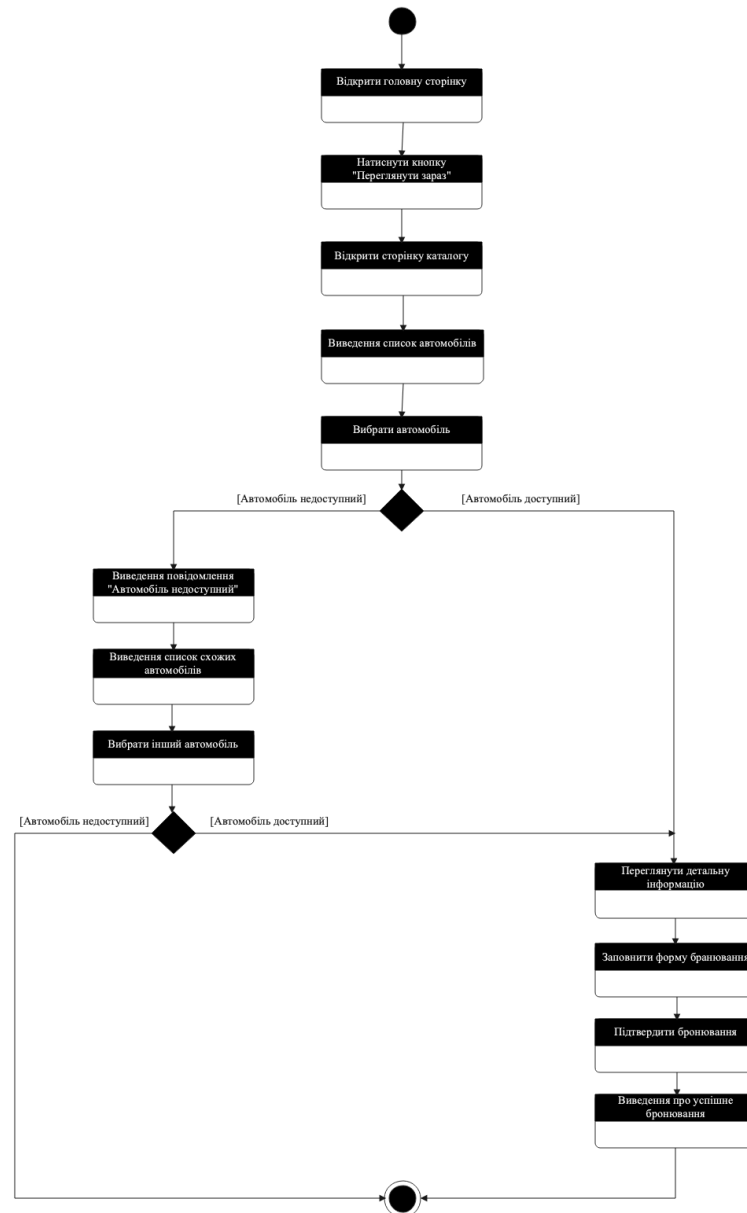


Рисунок 2.1 – Модель у вигляді діаграми діяльності веб-системи

Розробка діаграми діяльності була виконана у програмному забезпеченні EdrawMax [9].

2.3 Розробка структури системи

Для розробки вебсистеми бронювання авто для мандрівників необхідно окреслити її структуру, яка поділена на основні складові та відтворює послідовність взаємодії користувача з системою. Такий підхід забезпечить інтуїтивно зрозумілу навігацію та комфортність використання вебзастосунку.

Відповідно до розробленої структури, взаємодія користувача з системою розпочинається з головної сторінки. На ній розташоване навігаційне меню, яке містить основні пункти: «Головна», «Каталог» та «Улюблені». Також на головній сторінці розміщено кнопки для переходу до каталогу та перегляду улюблених.

На сторінці «Каталог» користувач має можливість переглянути список доступних авто, використати фільтри за місцем прокату, обладнанням або ціною, а також сортувати авто за цими параметрами. Вибравши конкретний автомобіль, користувач переходить на детальну сторінку, де може ознайомитися з описом транспортного засобу, характеристиками, відгуками інших клієнтів та скористатися формою бронювання.

На сторінці «Улюблені» користувач має змогу переглядати перелік доданих до обраного автомобілів та переходити до їхніх детальних сторінок. Це дає змогу швидко знайти авто, яке зацікавило раніше.

Детальна сторінка про автомобіль охоплює ключові розділи: зображення, назва моделі, місто, ціна, рейтинг, опис транспортного засобу, його характеристики, відгуки інших користувачів, а також форму для бронювання. Форма бронювання включає поля для вводу персональної інформації, вибору термінів оренди та способу оплати. Після успішної фіксації бронювання система виводить сповіщення про це на екран та дублює його на електронну пошту користувача.

У випадку невдалого бронювання, система видає повідомлення з підказками, як виправити помилки. Це дає користувачу змогу швидко відкоригувати введені дані та завершити бронювання.

Окрім того, було розроблено схематичне зображення, що візуально демонструє структуру веб-системи бронювання авто для туристів, включаючи всі ключові модулі та взаємозв'язки між ними.

Структуру вебсистеми відображено на рисунку 2.2 .

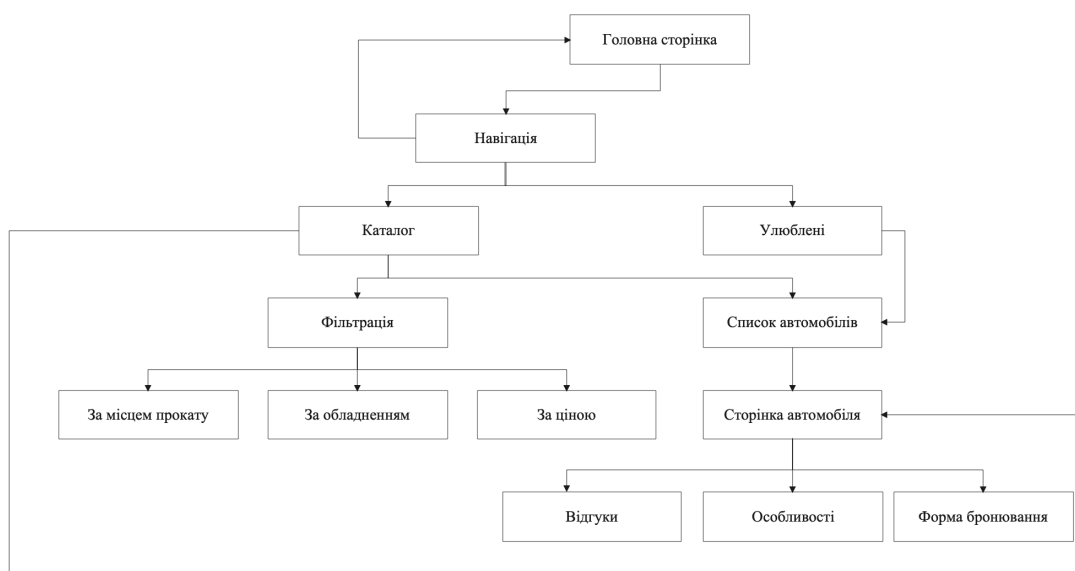


Рисунок 2.2 – Структура мобільного застосунку

Отже, побудова структури системи – ключовий момент у розробці, адже дає змогу забезпечити послідовність дій користувача та логічну організацію інтерфейсу. Це робить користування вебзастосунком комфортним і зменшує ймовірність помилок при взаємодії з системою.

2.4 Розробка алгоритму захисту даних платіжної картки у вебсистемі

Для забезпечення таємності та захищеності даних платіжних карток у веб-системі, що обслуговує бронювання автомобілів для мандрівників, було впроваджено захищений механізм опрацювання платіжної інформації. Цей механізм спирається на передовий платіжний сервіс Stripe SDK. Його застосування дозволяє не тільки уникнути потреби зберігати конфіденційні дані

на стороні клієнта або сервера, а й значно знижує ймовірність витоку персональних даних.

Процес захисту даних розпочинається з того, що дані картки передаються в Stripe SDK. Тут відбувається попереднє шифрування та підготовка інформації для токенизації. На цьому етапі ключово важливо переконатися в правильності введених реквізитів: номера картки, терміну її дії, CVV-коду та імені власника. Далі система визначає значення токена. Токен – це цифрове відображення платіжної інформації, котре не містить реальних реквізитів, тому його можна безпечно передавати між різними системами.

Далі відбувається валідація успішного отримання токена, що є аналізом цілісності інформації. Якщо дані успішно зчитані, система переходить до їх обробки та вилучення потрібної інформації. Після цього відбувається трансфер ключів та відповідних значень для формування єдиного масиву даних, який планується для подальшого запису. В останньому етапі застосовується Stripe SDK для генерації токена, котрий забезпечує захищене взаємоз'єднання між користувачем і платіжною платформою.

Після створення токена система переходить до фази оцінки можливості збереження даних. В цьому моменті здійснюється перевірка за певним сценарієм: у випадку, коли запис дозволено і жодних помилок не виявлено, система заносить токен до бази даних. Ця база даних знаходиться в ізоляції від публічної інформації, з використанням захищеного з'єднання та шифрування. Якщо виникає будь-яка помилка під час запису, користувач отримує відповідне сповіщення, а система призупиняє процес збереження. Це забезпечує відсутність пошкоджених або неповних даних.

Блок-схема алгоритму захисту платіжної інформації користувача представлена на рисунку 2.3.

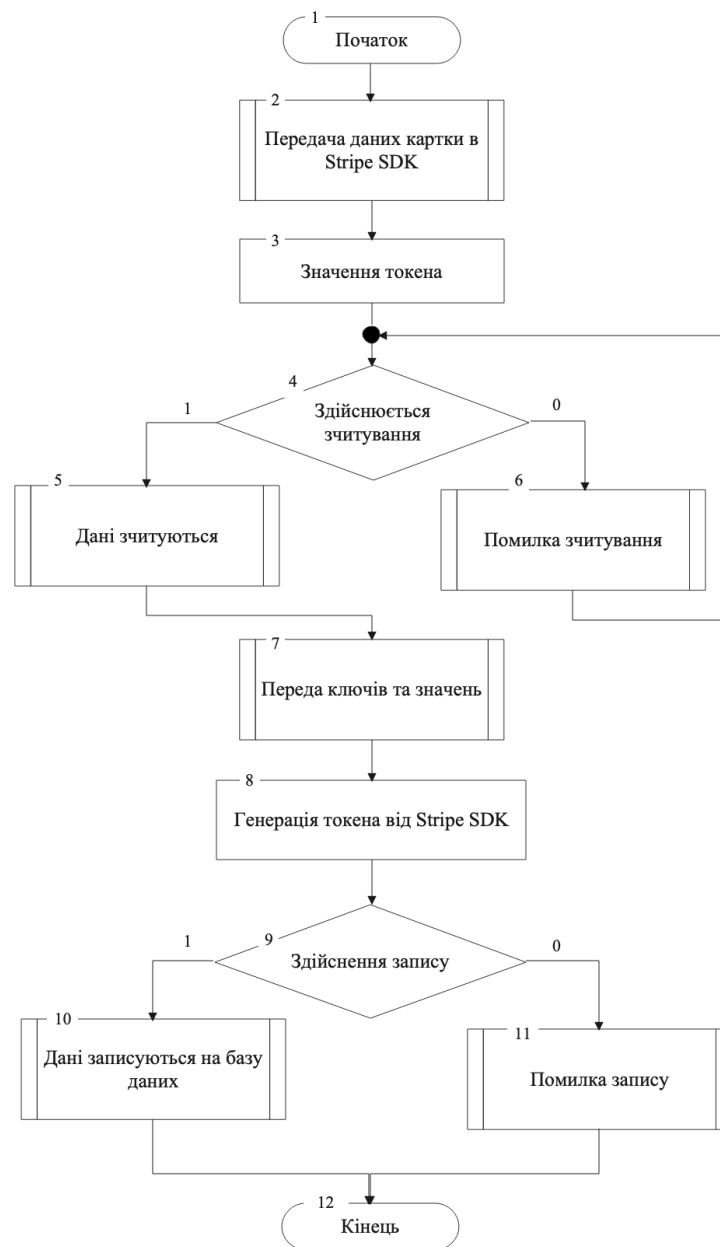


Рисунок 2.3 — Блок-схема алгоритму захисту даних платіжної картки користувача у вебсистемі

Отже, цей алгоритм гарантує максимальний захист даних вашої платіжної картки. Він не лише забороняє зберігання важливої інформації у загальному доступі, але й застосовує токенізацію та шифрування, після чого надійно зберігає дані у базі. Цей метод є ідеальним для онлайн-сервісів, які оперують електронними платежами, зокрема для систем бронювання, де першочерговими є безпека та швидкість проведення транзакцій.

2.5 Висновки

У другому розділі було проведено аналіз вхідних та вихідних даних програмних засобів. Було розроблено модель роботи програмного застосунку за допомогою UML діаграм та визначено структурну схему мобільного застосунку. Також, було розроблено алгоритм захисту даних користувачів мобільного застосунку для підтримки діяльності благодійних організацій.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ ВЕБСИСТЕМИ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Під час створення вебсистеми, призначеної для бронювання автомобілів, важливо визначитися з оптимальними технологіями для втілення кожного з модулів системи. Грамотний підбір технологічного стеку забезпечує високу продуктивність, інтуїтивної зрозумілості, стабільності веб-застосунку та його можливості адаптуватися до змін і збільшення налаштування.

Наступний етап розвитку отримала методики розробки вебсистем для бронювання транспортних засобів. У цій системі, на відміну від існуючих аналогів, були застосовані сучасні технології: Next.js, TypeScript, Node.js, MongoDB та React. Таке рішення дало змогу забезпечити високу швидкість роботи системи, гнучкості інтерфейсу, стабільність зберігання даних та можливість обробки значного обсягу інформації про автомобілі.

React – це популярна бібліотека для розробки користувацьких інтерфейсів, яка використовує компонентний принцип і дозволяє створювати вебзастосунки, що адаптуються та змінюються динамічно. Основна перевага React полягає в використанні компонентів, які повторно використовуються, що значно пришвидшує розробку та збільшує надійність коду. Застосовуючи віртуальний DOM, гарантує швидке оновлення інтерфейсу без необхідності перезавантажувати сторінку [10].

Компонентний підхід у React сприяє поділу інтерфейсу на логічні блоки, що спрощує підтримку та тестування. Крім того, застосування стану (State) та властивостей (Props) дозволяє ефективно керувати змінним контентом, що має важливе значення для вебсистем з інтенсивними взаємодіями.

TypeScript гарантує суворий контроль типів у проєкті, що дає змогу уникнути значної кількості помилок ще на етапі компіляції. Застосування TypeScript у вебсистемі збільшує стабільність коду та полегшує його підтримку завдяки інтуїтивно зрозумілому синтаксису. TypeScript дає змогу інтегрувати

існуючі JavaScript-бібліотеки без потреб повного переписування коду. Це робить його зручним у процесі поетапної міграції проекту на нові технології. Окрім цього, підтримка об'єктно-орієнтованого програмування дозволяє реалізувати складніші моделі даних та логіку бізнес-процесів [11].

Next.js – це популярний фреймворк для розробки вебзастосунків із взаємодією клієнт та сервера, який базується на бібліотеці React та використовує мову TypeScript для поліпшення якості коду та зниження кількості помилок на етапі розробки. Next.js реалізує статичну генерацію сторінок (SSG), так і серверний рендеринг (SSR), що сприяє високій продуктивності та зменшує час відповіді вебсистеми [12].

Завдяки вбудованій системі та підтримці API [13], Next.js спрощує організацію навігації та доступ до ключових функцій. Інтелектуальне попереднє завантаження сторінок сприяє створенню швидких та інтерактивних інтерфейсів, що позитивно впливає на зручність користування. Крім того, Next.js надає можливість для втілення динамічних маршрутів та створення сторінок з параметрами URL, що знадобиться під час відображення деталей бронювання певного автомобіля.

Node.js реалізує асинхронне виконання операцій, що дає змогу обробляти одночасно багато запитів одночасно, не блокуючи головний потік. Це важливо для вебсистеми бронювання, де швидкість обробки запитів є ключовим аспектом. Node.js підтримує модульну структуру, використовуючи npm, що дозволяє оперативно інтегрувати необхідні бібліотеки для обробки запитів, збереження даних та управління сесіями. Завдяки активному співтовариству та великій бібліотеці модулів, Node.js представляє оптимальним вибором для розробки продуктивних серверних застосунків [14].

MongoDB – це NoSQL база даних, що зберігає інформацію у вигляді JSON-документів. Цей спосіб чудово підходить для зберігання даних про автомобілі, бронювання та інші сутності. Завдяки гнучкості схеми даних дозволяє оперативно додавати нові поля, не переносючи всю базу. MongoDB демонструє високу швидкість при операціях читання та запису, що робить її

відміним вибором для вебзастосунків, які постійно змінюються. Крім того, реалізовано шардінг для горизонтального масштабування, що дає змогу працювати з величезним обсягами даних, без зниження продуктивності [15].

Для стилізації інтерфейсу було віддано перевагу вбудованим засобам CSS відмовившись від сторонніх бібліотек. Це сприяло зменшенню розміру коду та спрощенню процесу налаштування стилів під власні потреби. Застосування сучасних можливостей CSS [16], зокрема Flexbox і Grid, дає змогу розробляти адаптивний інтерфейс без потреб підключати додаткові модулі.

Завдяки застосування CSS-модулів, стилі кожного компонента є відокремленими, що зменшує ймовірність конфліктів та полегшує підтримку проекту. Цей підхід полегшує зміну тем оформлення та забезпечує стабільну роботу інтерфейсу на різноманітних пристроях.

Поєднання React, Next.js, TypeScript, Node.js, MongoDB та впроваджених властивостей CSS дало можливості створити швидкодіючу, адаптивну та надійну вебсистему. Застосування цих технологій дає змогу гарантувати високу продуктивність, швидкий час відповіді та зручність користування.

Отже, вибір технологічного стеку для реалізації вебсистеми бронювання автомобілів забезпечив створення продуктивного, надійного та зручного продукту, який відповідає сучасним вимогам до вебдодатків.

3.2 Розробка модуля фільтрації автомобілів у каталозі

Під час створення модуля фільтрації автомобілів у веб-системі бронювання для мандрівників взято до уваги чимало функціональних та технічних вимог, що мають забезпечити швидкий та зручний пошук потрібного транспорту. Головне завдання модуля – надати користувачеві інструменти для фільтрації автомобілів за ключовими параметрами: місце прокату, порядок сортування ціни (зростання або спадання) та доступне обладнання.

Модуль фільтрування втілено у вигляді окремого компоненту Filter, що розроблено за допомогою бібліотеки React. Основні параметри фільтрації включають місто прокату, набір обладнання, а також порядок сортування за

ціною. Всі ці налаштування зберігаються у станах компонента, що забезпечує динамічне оновлення результатів без потреб перезавантаження сторінки.

Фільтрування за локацією прокату втілено за допомогою текстового поля, де користувач може вказати назву міста або пункту оренди. Механізм запам'ятовування введеної інформації функціонує через змінну стану `location`, що динамічно змінюється при кожному вводі тексту. Після активації кнопки пошуку, система використовує введене значення місця для формування пошукового запиту. Такий спосіб дає змогу легко знаходити автомобілі саме у потрібному місці. Алгоритм фільтрації за локацією прокату реалізовано у функції `handleSearch` та зображено на рисунку 3.1.

```
<label className={s.locationLabel}>
  <span className={s.fieldTitle}>Місце</span>
  <div className={s.inputWrapper}>
    <Icon name="map" className={s.icon} size={20} />
    <Input
      type="search"
      name="location"
      placeholder="Місто"
      value={location}
      onChange={e => setLocation(e.target.value)}
      className={s.inputWithIcon}
    />
  </div>
</label>
```

Рисунок 3.1 — Логіка вибору місця оренди

Фільтрація за обладнанням здійснена через функцію `toggleEquipment`, що або додає, або видаляє об'єкт зі списку, відповідно до кліку на відповідну кнопку. Користувач може вибрати декілька опцій, серед яких: автоматична трансмісія, кондиціонер, кухня, ванна кімната та інше. Активні опції виділяються у інтерфейсі, що дозволяє миттєво визначити поточні

налаштування. Алгоритм вибору обладнання та його втілення з використанням функції `toggleEquipment` демонструється на рисунку 3.2.

```
const toggleEquipment = (item: string) => {
  setSelectedEquipment(prev =>
    prev.includes(item) ? prev.filter(i => i !== item) : [...prev, item]
  );
};
```

Рисунок 3.2 — Логіка вибору обладнання у модулі фільтрації автомобілів

Користувачу надається вибір способу сортування автомобілів за ціною: від найменшої до найбільшої або у зворотному порядку. Перемикання способу сортування відбувається завдяки функції `handleSort`, яка змінює значення змінної `sortPrice` у відповідності з вибраним варіантом. Цей механізм забезпечує зручний інструмент для користувача, щоб налаштовувати порядок відображення результатів відповідно до його вподобань. Алгоритм сортування за ціною, реалізований у функції `handleSort`, наочно показано на ілюстрації 3.3.

```
// Зміна сортування за ціною
const handleSort = (order: string) => {
  setSortPrice(order);
};

<div className={s.filterGroup}>
  <h4 className={s.groupTitle}>Сортування за ціною</h4>
  <div className={s.optionsGrid}>
    <button
      type="button"
      onClick={() => handleSort("asc")}
      className={` ${s.optionBtn} ${sortPrice === "asc" ? s.selected : ""} `>
      </button>
      Зростання
    </button>
    <button
      type="button"
      onClick={() => handleSort("desc")}
      className={` ${s.optionBtn} ${sortPrice === "desc" ? s.selected : ""} `>
      </button>
      Спадання
    </button>
  </div>
</div>
```

Рисунок 3.3 — Логіка сортування за ціною

Після того, як користувач обрав потрібні налаштування, він може активувати пошук, натиснувши кнопку «Пошук». Функція `handleSearch` відповідає за обробку всіх введених даних (місто, вибране обладнання та критерії сортування) та надсилання їх у вигляді запиту для показу відфільтрованого переліку автомобілів. Якщо необхідно скасувати фільтри, застосовується функція `handleReset`, яка видаляє всю інформацію з полів форми та повертає параметри до початкових. Принцип роботи пошуку та скидання фільтрів, реалізований у функціях `handleSearch` та `handleReset`, зображено на рисунку 3.4.

```
const handleSearch = () => {  
  onFilter(location, selectedEquipment, sortPrice);  
};  
  
// Скидання всіх фільтрів  
const handleReset = () => {  
  setLocation('');  
  setSelectedEquipment([]);  
  setSortPrice('');  
  onFilter('', [], '');  
};
```

Рисунок 3.4 — Логіка пошуку та скидання фільтрів у модулі фільтрації автомобілів

Зовнішній вигляд модуля фільтрації реалізований як панель, що містить поля для вказування міста, кнопки вибору техніки та сортування за вартістю. Інтерфейс організований за функціональним принципом: у верхній частині знаходиться поле для вводу місця прокату, далі – блок вибору обладнання, а внизу – кнопки сортування та управління пошуком. Завдяки такій структурі забезпечується зручне використання та швидке налаштування параметрів. Графічне представлення модуля фільтрації автомобілів показано на рисунку 3.5.

The image shows a web interface for filtering car rental options. It is titled "Фільтри" (Filters). Under the "Місце" (Location) section, there is a button labeled "Місто" (City). The "Обладнання транспортного засобу" (Vehicle equipment) section contains six buttons arranged in a 3x2 grid: "Автоматичний" (Automatic), "Ручна" (Manual), "AC", "Кухня" (Kitchen), "TV", and "Ванна кімната" (Bathroom). The "Сортування за ціною" (Sort by price) section has two buttons: "Зростання" (Ascending) and "Спадання" (Descending). At the bottom, there is a prominent red "Пошук" (Search) button and a "Скинути фільтри" (Reset filters) button.

Рисунок 3.5 — Графічний інтерфейс модуля фільтрації

Отже, модуль фільтрації автомобілів у вебсистемі бронювання є зручним інструментом для оперативного пошуку транспортних засобів відповідно до потреб користувача. Впровадження базової логіки компонента, динамічний вибір обладнання, фільтрація за місцем прокату та сортування за ціною дають змогу оптимізувати взаємодію користувача з каталогом автомобілів.

3.3 Розробка модуля оформлення бронювання автомобіля

Головним завданням модуля бронювання автівки є гарантування комфортного та безперебійного процесу подачі запиту на оренду обраного транспортного засобу. Модуль дозволяє користувачеві внести потрібні відомості для оформлення бронювання, обрати метод оплати та фіналізувати замовлення.

Для реалізації поставленої задачі було розроблено компонент `BookingForm` в `React`, що вміщує форму з базовими полями для введення інформації. Компонент включає текстові поля для введення імені, e-mail адреси,

контактного номеру телефону, вибору дати старту оренди, а також можливість додати коментар або побажання. Специфікою реалізації є опція вибору способу оплати, що дає змогу користувачу обрати між оплатою готівкою або використовуючи банківську картку.

У разі обрання готівкового розрахунку, все залишається без змін. Натомість, якщо клієнт планує сплачувати банківською карткою, система передбачає появу додаткових полів, куди необхідно ввести номер картки, дату закінчення терміну дії та код CVV. Це дозволяє клієнтам обирати найзручніший для них варіант оплати та адаптує процес бронювання відповідно до їхніх особистих вподобань. Механіка вибору способу оплати базується на змінній стану `paymentMethod`, яка зберігає дані про обраний спосіб. Користувач робить вибір між готівкою та карткою, використовуючи радіокнопки, які змінюють значення стану за допомогою функції `setPaymentMethod`. Відповідно до вибраного значення, форма динамічно оновлюється, відображаючи або приховуючи секцію з інформацією про картку. Логіка вибору способу оплати проілюстрована фрагментом коду на рисунку 3.6.

```
<div className={s.paymentMethods}>
  <label>
    <input
      type="radio"
      name="payment"
      value="cash"
      checked={paymentMethod === 'cash'}
      onChange={() => setPaymentMethod('cash')}
    />
    Готівка
  </label>
  <label>
    <input
      type="radio"
      name="payment"
      value="card"
      checked={paymentMethod === 'card'}
      onChange={() => setPaymentMethod('card')}
    />
    Картка
  </label>
</div>
```

Рисунок 3.6 — Логіка вибору способу оплати у формі бронювання

При введенні номера платіжної картки система автоматично застосовує форматування. Після кожної групи з чотирьох цифр додається пробіл. Це робить введення зручнішим і візуально зрозумілішим. Функція `formatCardNumber` відповідає за форматування номера картки. Вона прибирає з тексту всі символи, окрім цифр, і вставляє пробіли після кожних чотирьох цифр (див. рисунок 3.7).

```
// Форматування номера картки (додає пробіл кожні 4 цифри)
const formatCardNumber = (value: string) => {
  return value
    .replace(/\D/g, '')
    .replace(/(.{4})/g, '$1 ')
    .trim();
};
```

Рисунок 3.7 — Логіка форматування номеру картки у формі бронювання

Схожим чином організовано форматування дати закінчення дії картки. Після введення двох цифр місяця автоматично з'являється роздільник – слеш. Цей підхід забезпечує правильний формат введення дати. Функція `formatCardExpiry` реалізує логіку, що відповідає за коректне відображення терміну дії картки (див. рисунок 3.8). Такий принцип гарантує, що платіжні реквізити будуть введені правильно, що позитивно впливає на зручність використання форми.

```
// Форматування терміну дії (додає слеш після двох цифр)
const formatCardExpiry = (value: string) => {
  return value
    .replace(/\D/g, '') // Видаляємо всі нецифрові символи
    .replace(/^(\d{2})(\d)/, '$1/$2') // Додаємо слеш після двох цифр
    .slice(0, 5); // Обмежуємо довжину до 5 символів
};
```

Рисунок 3.8 — Логіка форматування терміну дії картки у формі бронювання

Для коректного введення даних платіжних засобів в системі бронювання передбачено валідацію ключових реквізитів картки. Перевіряються номер картки, період її чинності та CVV-код. Це сприяє зменшенню помилок при введенні та забезпечує захищене зберігання платіжної інформації. Валідація номера картки базується на регулярному виразі, який допускає 16 цифр, допускаючи пробіли між кожними чотирма цифрами. Це дає можливість вводити номер картки у звичному форматі, не переймаючись пробілами. Функція `isValidCardNumber` переконується у відповідності загальної кількості цифр та відсутності небажаних символів. У випадку невідповідності стандарту номера картки, система інформує про помилку (дивіться рисунок 3.9).

```
// Валідація номера картки (16 цифр з можливими пробілами)
const isValidCardNumber = (cardNumber: string) => {
  const cardRegex = /^(\d{4}[\s-]?){3}\d{4}$/;
  return cardRegex.test(cardNumber);
};
```

Рисунок 3.9 — Логіка валідації номера картки у формі бронювання

Для зазначення терміну придатності картки застосовується перевірка у форматі ММ/YY. Функція `isValidCardExpiry` виконує контроль як відповідності формату дати, так і актуальності періоду. У випадку, коли користувач вказує місяць, що перевищує 12, або рік з минулого, на формі буде відображено сповіщення про некоректний термін дії (див. Рисунок 3.10). Окрім цього, після введення двох цифр для місяця автоматично з'являється слеш, забезпечуючи коректне форматування. Це сприяє зменшенню типових помилок при введенні дати та спрощує процес. Таким чином клієнт має впевненість у відповідності даних платіжної картки вимогам системи, що знижує ймовірність відмови у проведенні платежу.

```
// Валідація терміну дії картки (формат ММ/YY)
const isValidCardExpiry = (expiry: string) => {
  // Перевіряємо формат ММ/YY
  const expiryRegex = /^(0[1-9]|1[0-2])\/\d{2}$/;
  if (!expiryRegex.test(expiry)) return false;

  // Розділяємо місяць і рік
  const [month, year] = expiry.split('/').map(Number);

  // Отримуємо поточний місяць та рік
  const now = new Date();
  const currentMonth = now.getMonth() + 1; // місяць від 0 до 11
  const currentYear = Number(String(now.getFullYear()).slice(2)); // останні 2 цифри року

  // Перевіряємо чи рік дійсний та не в минулому
  if (year < currentYear) return false;

  // Якщо рік той самий, перевіряємо місяць
  if (year === currentYear && month < currentMonth) return false;

  return true;
};
```

Рисунок 3.10 — Логіка валідації терміну дії картки у формі бронювання

Валідація CVV полягає у перевірці на тризначне числове значення. Якщо замість цифр вводяться букви або символи, система реагує інформуванням про помилку, відображаючи відповідний текст під полем (дивіться рисунок 3.11). Функція `isValidCVV` дозволяє швидко визначити коректність введеного коду. При виникненні помилок усі повідомлення про некоректне введення відображаються безпосередньо під відповідними полями через стан `errors`. Це дає можливість оперативно виправити помилки, не повертаючись до заповнення з початку.

```
// Валідація CVV (3 цифри)
const isValidCVV = (cvv: string) => {
  const cvvRegex = /^\d{3}$/;
  return cvvRegex.test(cvv);
};
```

Рисунок 3.11 — Логіка валідації терміну дії картки у формі бронювання

Логіка вибору дат втілена через компонент DatePicker [17], що забезпечує зручний ввід та валідність дат. У формі бронювання користувачеві потрібно вказати дві дати: дату початку оренди та дату завершення оренди. Найменшою можливою датою початку є сьогоднішня, що унеможливлює вибір дат у минулому. Це досягається параметром `minDate`, що дорівнює сьогоднішній даті.

Після вибору дати початку оренди, система автоматично змінює мінімальну дату завершення. Найменша дата завершення встановлюється на обрану дату початку або пізнішу. Це потрібно для запобігання ситуаціям, коли користувач випадково обере дату завершення раніше дати початку. Приклад логіки вибору дат показано на рисунку 3.12.

```
<div className={s.datePickers}>
  <DatePicker
    selected={startDate}
    onChange={(date) => setStartDate(date)}
    minDate={today}
    placeholderText="Початок бронювання*"
    className={s.input}
    dateFormat="yyyy-MM-dd"
    required
  />
  <DatePicker
    selected={endDate}
    onChange={(date) => setEndDate(date)}
    minDate={startDate || today}
    placeholderText="Кінець бронювання*"
    className={s.input}
    dateFormat="yyyy-MM-dd"
    required
  />
</div>
```

Рисунок 3.12 — Логіка вибору дат у формі бронювання

Для коректного внесення контактних відомостей у форму резервування впроваджено перевірку номеру телефону та електронної пошти. Валідація телефонного номеру відбувається за допомогою функції `isValidPhone`, що застосовує регулярний вираз для контролю формату `+380XXXXXXXXXX`. Вираз враховує пробіли, тире та круглі дужки, дозволяючи вносити номер у звичній формі. При виявленні неточностей система миттєво сповіщає користувача.

Перевірка електронної адреси реалізується функцією `isValidEmail`, яка контролює присутність символу `"@"` та коректність доменного імені. Якщо формат адреси виявиться невірним, система видає сповіщення про помилку. Такий метод мінімізує можливість помилок при введенні контактних даних. Логіку перевірки реалізовано у функціях `isValidPhone` та `isValidEmail`, що зображено на рисунку 3.13.

```
// Валідація телефону
const isValidPhone = (phone: string) => {
  const phoneRegex = /^\\+?\\d\\s-()]{10,12}$\\//;
  return phoneRegex.test(phone);
};

// Валідація номера картки (16 цифр з можливими пробілами)
const isValidCardNumber = (cardNumber: string) => {
  const cardRegex = /^(\\d{4}[\\s-]?){3}\\d{4}$\\//;
  return cardRegex.test(cardNumber);
};
```

Рисунок 3.13 — Логіка валідації номера телефону та електронної пошти

Після натискання кнопки «Надіслати», система створює об'єкт, збираючи всі введені користувачем дані, і відправляє його на сервер за допомогою асинхронного запиту. Перед відправкою дані форми конвертуються у формат JSON. Якщо користувач обрав оплату карткою, то номер, термін дії та CVV картки попередньо перетворюються на токен, щоб гарантувати безпечну

передачу. Механізм збереження цих даних втілено у функції `handleSubmit`, яка формує структуру об'єкта `bookingData`.

Після успішної відправки запиту сервер перевіряє отримані дані та коректність платіжної інформації. Якщо оплата карткою, система взаємодіє з платіжним шлюзом для створення токена, який захищає передачу платіжної інформації. Токен зберігається у базі даних разом з іншими деталями бронювання, що дозволяє проводити подальші операції без зберігання реальних даних картки.

Після успішного збереження на сервері, база даних оновлюється новим записом про бронювання. У випадку помилки або отримання негативного результату від сервера, система сповіщає користувача про збій через повідомлення, згенероване функцією `toast.error()`. У разі успіху, користувач отримує повідомлення про успішне бронювання, а всі поля форми автоматично очищаються, щоб забезпечити зручність подачі нової заявки.

Отже, модуль бронювання не лише полегшує процес подачі заявки, але й забезпечує безпечне зберігання даних у базі, роблячи його надійним та функціональним інструментом для користувачів.

3.4 Розробка інтерфейсу

Під час розробки інтерфейсу вебсистеми бронювання автомобілів було приділено особливу увагу його зрозумілості та зручності для користувача.

Головним кольором інтерфейсу обрано білий, оскільки даний колір асоціюється з чистотою, простотою та легкістю сприйняття інформації. Додатковими кольорами обрано чорний та оранжевий, оскільки вони контрастують один одному. При відкритті веб-сторінки, користувача зустрічає головна сторінка (див. рисунок 3.14).

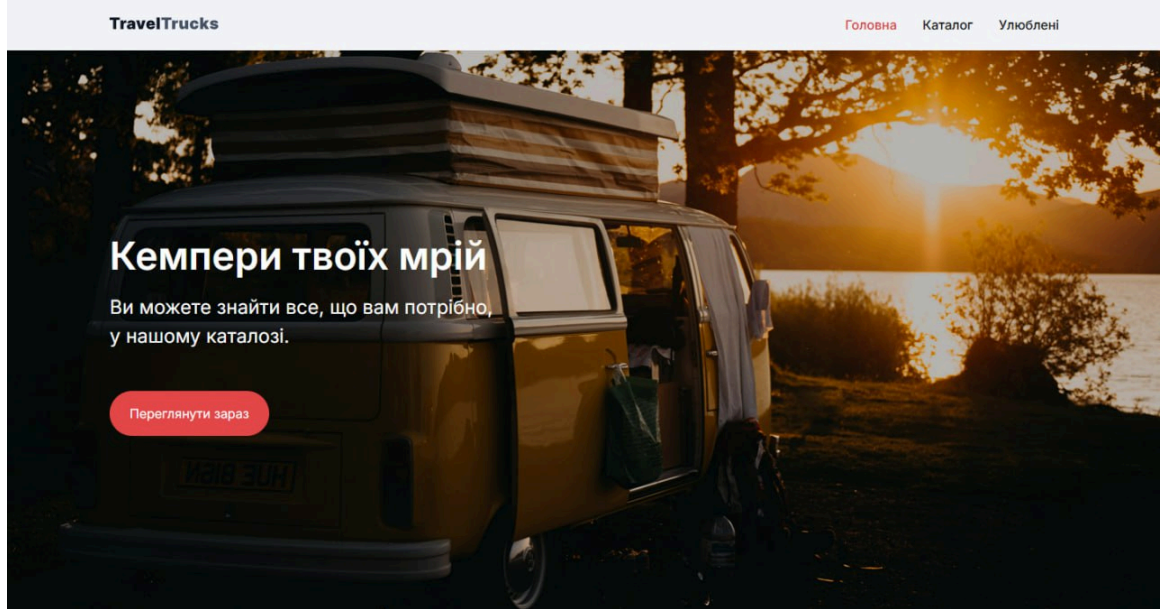


Рисунок 3.14 – Екран завантаження

Оскільки веб-система повинна забезпечувати взаємодію користувачів із процесом замовлення транспортних засобів для мандрівок, було вирішено сконструювати зручний для сприйняття інтерфейс із чітким розподілом між функціональними блоками. Зокрема, були спроектовані окремі розділи для підбору авто, перегляду деталей замовлення та управління особистими відомостями користувача. Окрім того, розроблено меню веб-застосунку (див. рисунок 3.15), що полегшує навігацію та доступ до основних функцій системи.



Рисунок 3.15 – Меню веб-сторінки

Каталог автомобілів – це фундамент веб-системи, що надає користувачам можливість легко знаходити й обирати автомобіль для бронювання. Дизайн каталогу структуровано так, щоб користувачі могли миттєво знаходити потрібний автомобіль за допомогою пошукової системи та фільтрів.

Для забезпечення комфортного пошуку реалізовано поле введення місця оренди, де планує взяти авто в оренду. Після введення даних, система покаже лише ті автомобілі, що доступні в зазначеному регіоні. Крім того, у каталозі реалізовано фільтрацію за параметрами: тип транспорту та обладнанням.

Результати пошуку подаються у форматі карток з базовою інформацією про кожну машину (див. рисунок 3.16).

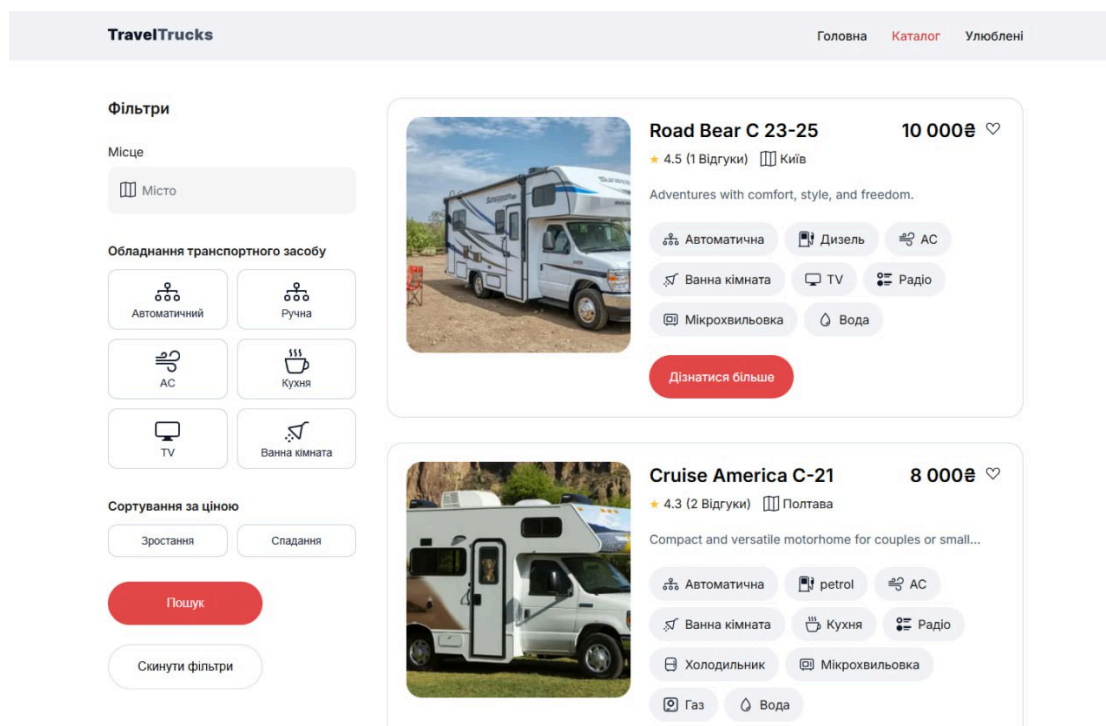


Рисунок 3.16 — Каталог автомобілів

Картка автомобіля – ключовий візуальний акцент каталогу, що забезпечує користувачеві доступ до базових відомостей про транспортний засіб. Кожна картка містить такі елементи:

- зображення автомобіля;
- назву автомобіля;
- ціну оренди;
- кнопку «Улюблене» для додавання автомобіля до списку обраних;
- місце розташування автомобіля;
- рейтинг користувачів;
- короткий опис;
- теги (наприклад, «кондиціонер», «автомат», «GPS»);
- кнопку «Дізнатися більше», що веде на сторінку з детальною інформацією.

Картка – інтерактивний об’єкт. Після кліку на кнопку «Дізнатися більше» відбувається перехід на сторінки конкретного автомобіля з розширеним описом та можливістю оформлення бронювання автомобіля (див. рисунок 3.17).

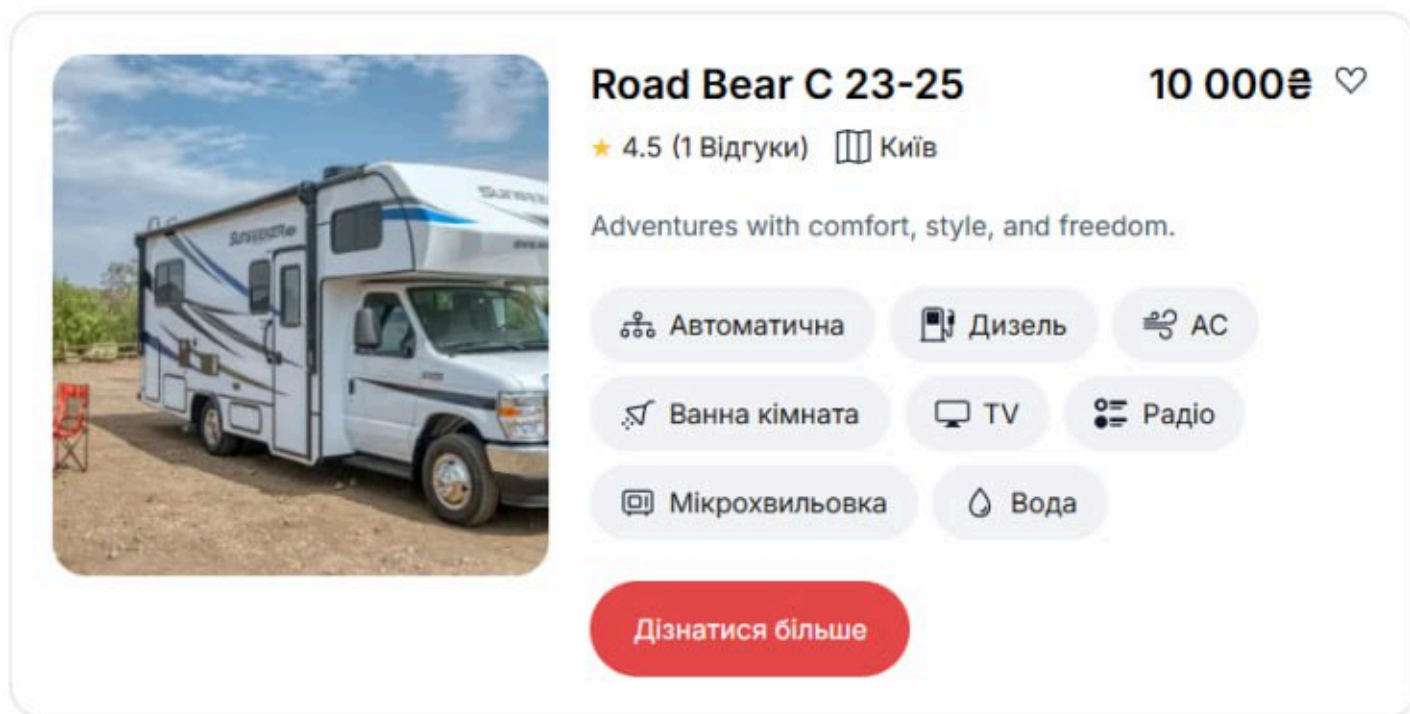


Рисунок 3.17 — Картка автомобіля

Також у кожній картці є кнопка «Дізнатися більше», натискання якої відкриває сторінку з детальною інформацією про автомобіль, відгуками та формою бронювання.

Після натискання кнопки «Дізнатися більше» користувач потрапляє на сторінку відповідного автомобіля, де наводиться детальна інформація. Представлено розширений опис автомобіля, зокрема його технічні дані та зручності. Основна інформація подана як заголовок, що містить назву автомобіля, середню оцінку користувачів із кількістю рецензій, місця розташування та ціну оренди. Основне зображення автомобіля дає змогу отримати візуальне враження про транспортний засіб, а додаткова галерея містить фото інтер'єру та екстер'єру. Такий спосіб дозволяє користувачу миттєво оцінити автомобіль ще до перегляду детальної інформації. Загальний вигляд верхньої частини сторінки автомобіля зображено на рисунку 3.18.

Road Bear C 23-25

★ 4.5 (1 Відгуків) 📍 Київ

10 000€

Пригоди з комфортом, стилем і свободою.

Рисунок 3.18 — Загальний вигляд сторінки автомобіля

Під головним контентом сторінки автомобіля розміщено блоки, що містять інформацію про автомобіль: його особливості, відгуки користувачів та форма бронювання. Над блоком особливостей розташовані дві кнопки: «Особливості» та «Відгуки». Завдяки цим кнопкам можна вільно перемикатися між блоками, забезпечуючи комфортне ознайомлення з даними. За замовчуванням відображається блок «Особливості», але при натисканні на кнопку «Відгуки» відбувається зміна на блок з коментарями користувачів. Якщо натиснути на кнопку «Особливості», поверне до переліку технічних характеристик автомобіля. Така структура дозволяє раціонально використовувати простір на сторінці та акцентувати увагу на найбільш важливу інформацію.

Блок особливостей представляє ключові характеристики автомобіля у вигляді стислих піктограм, що демонструють наявність тих чи інших опцій обладнання. Серед ключових особливостей можна виділити автоматичну або ручну трансмісію, кондиціонер, кухню, ванну кімнату та радіо. Використання

графічних елементів сприяє легкому застосуванню інформації навіть при швидкому перегляді. Особливості автомобіля зображено на рисунку 3.19.

Особливості Відгуки

Автоматична

Дизель

Кондиціонер

Ванна кімната

Телевізор

Радіо

Мікрохвильовка

Вода

Деталі автомобіля

Форма	Алков
Довжина	7.3m
Ширина	2.65m
Висота	3.65m
Бак	208l
Споживання	30l/100km

Рисунок 3.19 — Блок особливостей автомобіля

При переході на вкладку «Відгуки» користувачеві стає доступним блок з коментарями попередніх бронювання, що дає змогу ознайомитися з реальним досвідом використання автомобіля. Блок відгуків розроблений для формування враження про автомобіль на основі оцінок користувачів. Він містить загальну

оцінку транспортного засобу, кількість опублікованих відгуків та можливість залишення власного коментаря після завершення оренди. Це дозволяє формувати рейтинг автомобіля на основі реального досвіду клієнтів, що підвищує рівень довіри до системи. Відгуки користувачів представлені на рисунку 3.20.

Особливості Відгуки

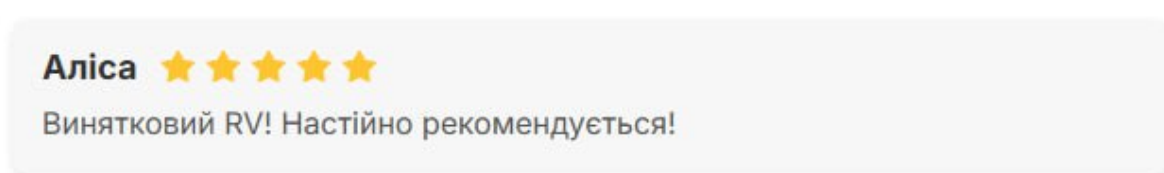


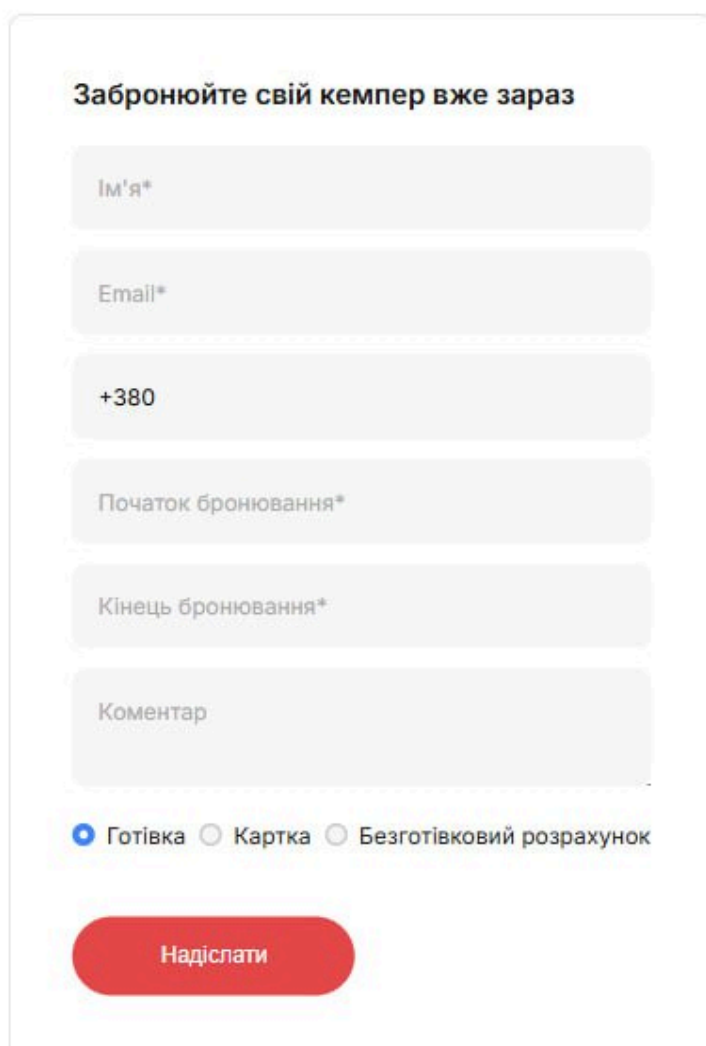
Рисунок 3.20 — Блок відгуків автомобіля

Форма бронювання розташована поруч із блоками з особливостями та відгуками. Вона залишається статичною, незалежно від того, який з цих блоків переглядається (особливості чи відгуки). Така структура забезпечує користувачеві безперервний доступ до форми, значно полегшуючи процес бронювання.

Для оформлення бронювання потрібно ввести особисту інформацію, а саме: ім'я, електронної адреси та номер телефону. Також необхідно повинен вказати дати початку та завершення бронювання, що дозволяє точно планувати оренду. Додатково, передбачено поле для коментаря, де користувач може вказати додаткові побажання або уточнення, наприклад, потребу в дитячому кріслі чи GPS-навігаторі.

Важливим елементом форми є блок вибору способу оплати. Користувачу пропонується обрати один із трьох варіантів: готівка, банківська картка або безготівковий розрахунок. При виборі оплати карткою, з'являються додаткові поля для введення платіжних даних: номер картки, термін дії (у форматі ММ/

YY) та CVV-код. Це дає змогу адаптувати процес бронювання під потреби користувача, надаючи можливість вибір зручного способу оплати. Інтерфейс форми з вибором готівкової або безготівкової оплати зображено на рисунку 3.21.



The image shows a web form titled "Забронюйте свій кемпер вже зараз" (Book your camper now). The form contains the following elements:

- A header text: "Забронюйте свій кемпер вже зараз".
- A text input field labeled "Ім'я*" (Name*).
- A text input field labeled "Email*".
- A text input field with the prefix "+380" for a phone number.
- A text input field labeled "Початок бронювання*" (Start of booking*).
- A text input field labeled "Кінець бронювання*" (End of booking*).
- A text input field labeled "Коментар" (Comment).
- A payment method selection section with three radio buttons: "Готівка" (Cash), "Картка" (Card), and "Безготівковий розрахунок" (Non-cash payment). The "Готівка" option is selected.
- A red button labeled "Надіслати" (Send).

Рисунок 3.21 — Форма бронювання

Після введення всіх необхідних даних та вибору способу оплати, користувач натискає кнопку «Надіслати» для підтвердження замовлення. Система автоматично надсилає на вказану електронну пошту підтвердження з деталями бронювання. Такий підхід гарантує оперативну обробку заявок та зручність і безпеку користувача під час оформлення бронювання.

Інтерфейс форми бронювання зображено на рисунку 3.22.

The form is titled "Забронуйте свій кемпер вже зараз" (Book your camper now). It contains several input fields: "Ім'я*" (Name), "Email*", a phone number field with "+380" as a prefix, "Початок бронювання*" (Start of booking), "Кінець бронювання*" (End of booking), and a "Коментар" (Comment) field. Below these is a payment method selection section with three radio buttons: "Готівка" (Cash), "Картка" (Card, which is selected), and "Безготівковий розрахунок" (Non-cash payment). Following the payment selection are three more input fields: "Номер картки" (Card number), "Термін дії (ММ/YY)" (Expiration date), and "CVV". At the bottom of the form is a red button labeled "Надіслати" (Send).

Рисунок 3.22 — Форма бронювання з вибором оплати карткою

Розробка графічного інтерфейсу програмного застосунку була проведена у середовищі розробки VS Code [18] мови програмування TypeScript.

3.5 Висновки

У третьому розділі було обґрунтовано вибір технологій та інструментів, що використовувалися для розробки програмних модулів вебсистеми бронювання автомобілів. Після аналізу вимог до системи та її основних задач

було обрано комбінацію технологій, а саме React для створення інтерфейсу користувача, Next.js для серверного рендерингу та маршрутизації, Node.js для реалізації серверної логіки та MongoDB як базу даних. У розділі також детально описано розробку основних програмних модулів, зокрема модуля фільтрації автомобілів, модуля відображення каталогу транспортних засобів та форми бронювання. Крім того, було презентовано процес створення інтерфейсу сторінки автомобіля, який передбачає логічний поділ на блоки: загальна інформація, технічні характеристики автомобіля, відгуки користувачів та форма бронювання. Значну увагу було приділено зручності використання інтерфейсу та взаємодії з даними.

4 ТЕСТУВАННЯ ЗАСТОСУНКУ

4.1 Тестування системи

Тестування програмного забезпечення вебсистеми бронювання автомобілів передбачає ретельну перевірку функціональності з метою виявлення недоліків, похибок чи невідповідної поведінки. Також воно включає оцінку відповідності заявленим вимогам до функцій [19].

Ключовою метою тестування є забезпечення якості, надійності та безперебійної роботи системи перед її впровадженням і початком реального використання.

Підготовка до тестування передбачала розгортання локальної тестової копії веб-застосунку та складання основного плану перевірки. Було виокремлено основні сценарії користувацької взаємодії з інтерфейсом: від перегляду доступних автомобілів у каталозі до завершення бронювання. Тестування здійснювалось за допомогою браузерів, включаючи Google Chrome, Mozilla Firefox, Safari та інші найпоширеніші платформи.

Насамперед було проведено функціональне тестування, що дозволило оцінити коректність роботи ключових складових: фільтрування авто, сортування, перегляд інформації про транспорт та взаємодія з формою бронювання. Особливу увагу приділили тестуванню логіки перевірки даних, введених користувачем, та їх обробці. Виявлені помилки чітко відображались поруч з відповідними полями, не зупиняючи користувача в його діях [20].

Наступним етапом було тестування на відмови. Воно включало перевірку відповіді системи на введення помилкової інформації, пропуск обов'язкових полів чи недотримання формату [21]. Система успішно обробляла подібні ситуації, надавала підказки та запобігала надсиланню неповних або некоректних даних.

Окрім того, було проведено тестування продуктивності. Протестована швидкість завантаження сторінки каталогу, оперативність фільтрації та

стійкість реакції на взаємодію з інтерфейсом. Отримані результати засвідчили задовільний час відгуку для всіх ключових операцій.

Тестування продуктивності вебсистеми за допомогою інструмента Lighthouse відображено на рисунку 4.1.

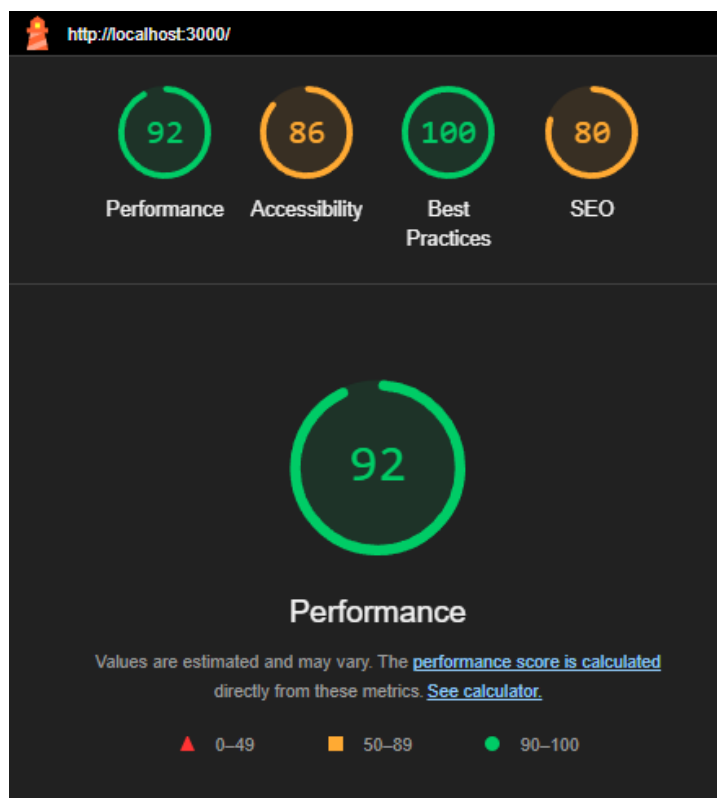


Рисунок 4.1 — Результати тестування продуктивності вебсистеми в Lighthouse

Було також здійснено тестування поведінки веб-системи, коли користувач заповнює форму бронювання з помилками або не повністю. Перевірялися ситуації, коли обов'язкові поля залишалися порожніми або дані вводилися в неправильному форматі, наприклад, неправильна адреса електронної пошти або номер телефону. У таких випадках система негайно реагує – відображає повідомлення про помилку, блокуючи надсилання форми до моменту виправлення всіх даних. Цей сценарій продемонстровано на рисунку 4.2, де наведено приклад повідомлення про некоректне заповнення полів.

The image shows a web form titled "Забронюйте свій кемпер вже зараз" (Book your camper now). The form contains several input fields, each with a validation error message in red text below it:

- Имя:** Макс. No error.
- Email:** max@gmailcom. Error: "Неправильний формат email" (Incorrect email format).
- Телефон:** +38068541. Error: "Неправильний формат телефону (10-12 цифр)" (Incorrect phone format (10-12 digits)).
- Начало бронирования:** 2025-05-21. No error.
- Конец бронирования:** 2025-05-28. No error.
- Комментарий:** Коментар. No error.
- Платеж:** Radio buttons for "Готівка" (Cash) and "Картка" (Card). "Картка" is selected.
- Номер карты:** 8787. Error: "Некоректний номер картки (16 цифр)" (Incorrect card number (16 digits)).
- Срок действия:** 01/00. Error: "Некоректний термін дії (ММ/YY або минула дата)" (Incorrect validity term (MM/YY or past date)).
- CVV:** 12. Error: "Некоректний CVV (3 цифри)" (Incorrect CVV (3 digits)).

At the bottom of the form is a red button labeled "Надіслати" (Send).

Рисунок 4.2 — Реакція вебсистеми на введення некоректних даних у форму бронювання

Окрему увагу зосереджено на валідації графіка бронювання. Підтверджено, що користувач не може вибрати дати, які вже минули – календар реалізує автоматичне блокування. Додатково, не дозволено вказувати дату закінчення оренди, яка передує даті її початку. Ця логіка працює в реальному часі та не вимагає ручного контролю з боку користувача, що гарантує коректність введених даних безпосередньо під час вибору.

Окрім цього, було перевірено ситуацію, коли користувач планує забронювати кемпер на дні, які вже зарезервовано. У такому разі система не дозволяє підтвердити замовлення та сповіщає користувача про те, що вказаний транспортний засіб недоступний у ці дати. Цей сценарій також було протестовано, і результат показано на рисунку 4.3, де видно повідомлення про зайнятість дат.

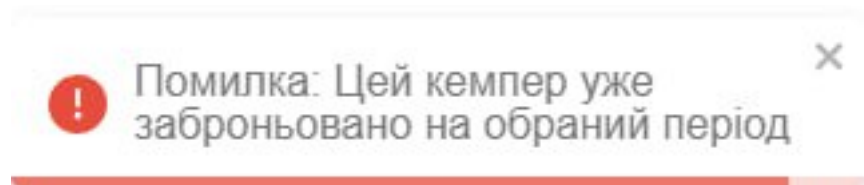


Рисунок 4.3 — Повідомлення про неможливість бронювання кемпера на зайняті дати

Окремо було протестовано функціонал фільтрування на сторінці каталогу. Результати показали правильну роботу системи при виборі фільтрів, враховуючи місце прокату, додаткове обладнання та ціновий діапазон. Після введення критеріїв система відображає виключно ті транспортні засоби, які повністю відповідають заданим умовам. Ілюстрацію цього процесу можна побачити на малюнку 4.4.

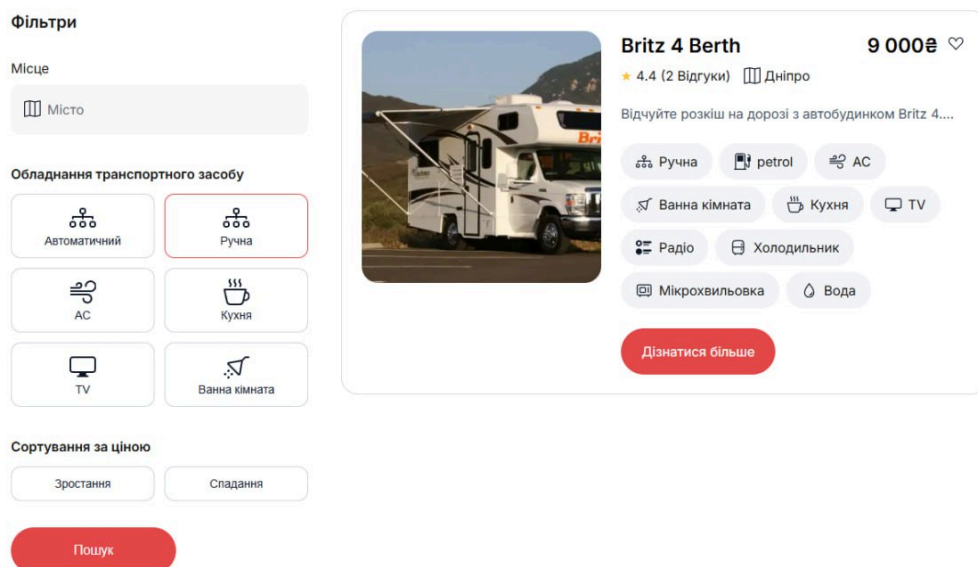


Рисунок 4.4 — Результат роботи фільтрів на сторінці каталогу

Фінальний етап перевірки стосувався цілковитого проходження користувачем процедури бронювання. Після заповнення всіх відомостей, вибору наявного періоду та підтвердження заявки система виводить повідомлення про вдале бронювання. Цей сценарій ілюструється на рисунку 4.5, що демонструє правильну обробку введених даних і завершення процесу.

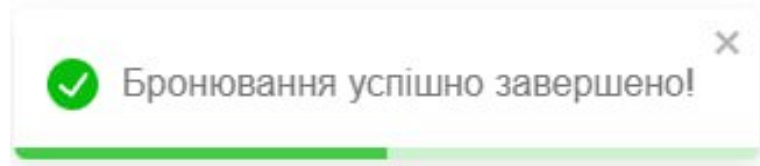


Рисунок 4.5 — Повідомлення про успішне завершення процедури бронювання

Для перевірки гнучкості інтерфейсу було випробувано відображення компонентів на різноманітних розмірах екрану, зокрема для смартфонів з діагоналлю 5.5" та планшетів 10". Усі елементи правильно масштабуються, не виходять за межі зони перегляду, забезпечуючи зручну взаємодію з юзером. Адаптивний дизайн протестовано за допомогою Google Chrome DevTools. Візуалізацію наведено на рисунку 4.6.

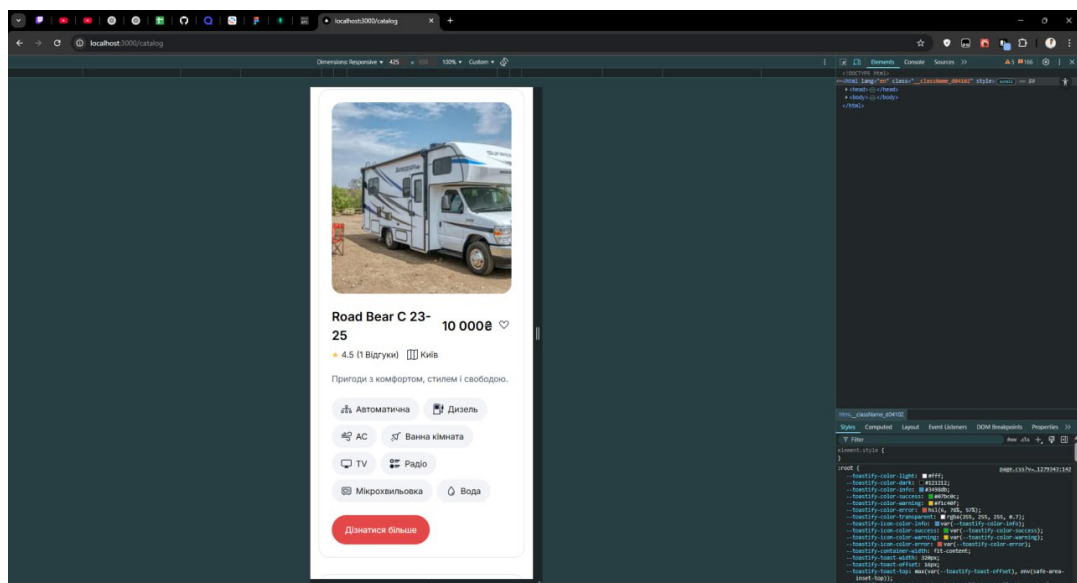


Рисунок 4.6 — Адаптивне відображення каталогу автомобілів на мобільному пристрої

За винятком перевірки каталогу, було ще випробувано адаптивність форми бронювання. Навіть на екранах малої ширини усі поля форми лишаються розбірливими, логічно впорядкованими та доступними для взаємодії без потреби горизонтального прокручування. Кнопка підтвердження розташована зручно, не перекривається іншими елементами інтерфейсу, що гарантує комфортне заповнення форми з мобільного. Результати цього тестування ілюструє малюнок 4.7.

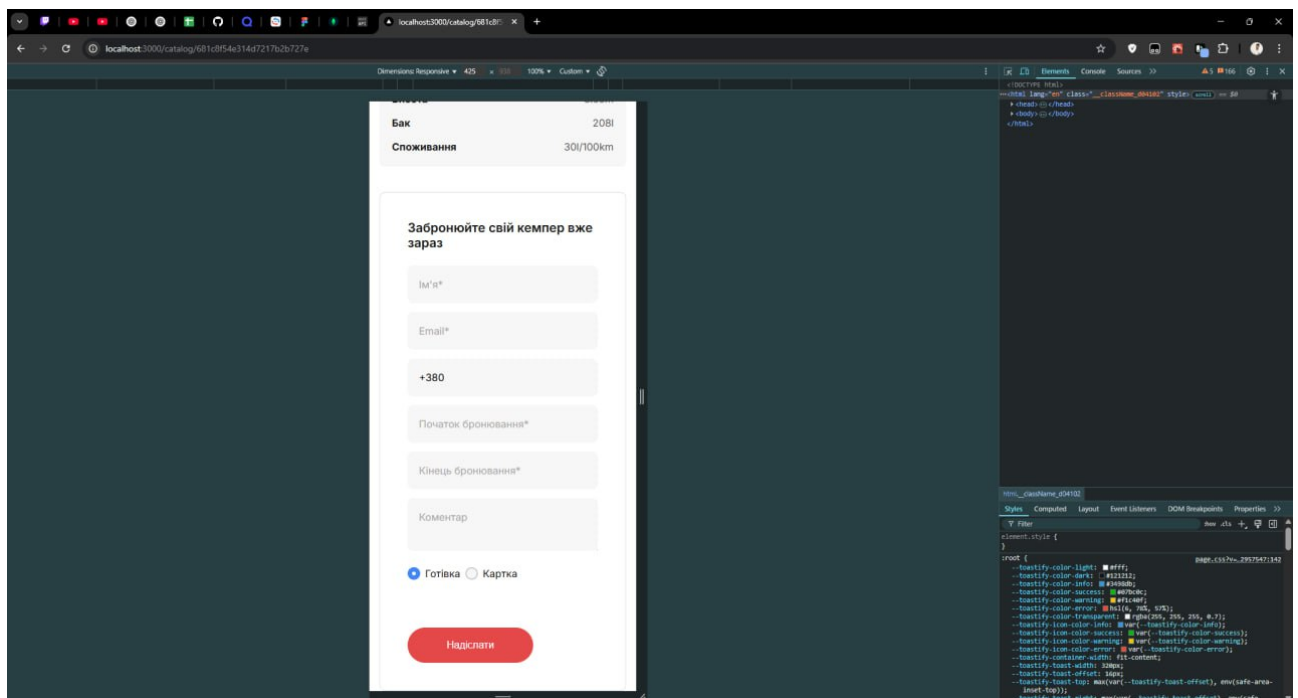


Рисунок 4.7 — Адаптивне відображення форми бронювання на мобільному пристрої

Внаслідок курсового проекту було розроблено вебсистему для резервування автомобілів, що дозволяє користувачу зручно переглядати, фільтрувати, сортувати й бронювати авто. Реалізовано адаптивний інтерфейс, перевірку введених даних, захист платіжної інформації та обробку помилок.

Проведене тестування засвідчило правильність роботи основного функціоналу, сумісність з різними пристроями та зручність використання. Система може бути використана як окрема платформа для мандрівників або як частина туристичного сервісу.

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту [22]. Деталі щодо мінімальної та рекомендованої конфігурації розміщено в таблиці 4.1.

Таблиця 4.1 – Вимоги до апаратного забезпечення

Вимоги до конфігурування	Рекомендовано	Мінімально
Процесор	Intel Core i5 або AMD Ryzen 5	Intel Core i3 або аналог
Оперативна пам'ять	8 ГБ RAM	4 ГБ RAM
Вільний простір на диску	1 ГБ (для кешу браузера, тимчасових файлів)	512 МБ
Відеокарта	Інтегрована (Intel UHD / AMD Vega)	Інтегрована (з підтримкою WebGL)
Операційна система	Windows 10 або macOS Big Sur	Windows 7 або macOS Mojave
Монітор	Роздільна здатність не менше 1920x1080 пікселів, діагональ 21" або більше	Роздільна здатність не менше 1280x720 пікселів, діагональ 15" або більше
Браузер	Google Chrome 110+, Firefox 100+, Safari 15+	Google Chrome 90+, Firefox 88+
Інтернет-з'єднання	10 Мбіт/с стабільне	1 Мбіт/с мінімум

Після того, як вебсистема бронювання автомобілів завантажується, перед користувачем постає головний екран. Цей екран відіграє основний елемент інтерфейсу, адже він створює перше враження від системи і відкриває швидкий доступ до основних функцій. Головний екран вебзастосунку виконаний в стилі мінімалізму, приділяючи увагу легкості навігації та зрозумілості відображення

інформації. Він включає основне меню, завдяки якому можна миттєво перемикатися між ключовими сторінками системи. На екрані також розташоване привітання користувача з короткою інформацією.

Загалом вебсистема розділена на три сторінки: «Головна», «Каталог» та «Улюблені». Найбільш важлива серед них – це сторінка «Каталог», де користувачі мають змогу переглядати доступні транспортні засоби, використовувати фільтри для пошуку за певними параметрами та ознайомлюватися з детальною інформацією про кожний автомобіль. Сторінка «Улюблені» надає доступ до списку обраних автомобілів, відміченими користувачем під час огляду каталогу. На цій сторінці зберігаються транспортні засоби, які користувач бажає переглянути повторно або планує орендувати пізніше. Меню навігації розміщене у верхній частині екрану та забезпечує миттєвий перехід до сторінок вебсистеми.

На екрані «Каталог» можна побачити всі транспортні засоби, що доступні для бронювання. У лівій частині каталогу розміщено панель фільтрації, яка дає змогу обрати місто оренди, необхідне обладнання та відсортувати за цінами. Поруч знаходиться перелік автомобілів, поданий у вигляді карток з основною інформацією: фото, назва, місто, ціна, короткий опис та рейтинг. Якщо натиснути на кнопку «Дізнатися більше», користувач перейде на детальну сторінку з розширеним описом, відгуками та формою бронювання.

Інтерфейс інтуїтивно зрозумілий і не потребує додаткових інструкцій для базової навігації. Усі дії користувача супроводжуються візуальними підказками, що підвищує зручність взаємодії з вебсистемою.

Після застосування фільтрів перед користувачем відображається перелік транспортних засобів, що відповідають вибраним критеріям.

Головний екран вебсистеми бронювання автомобілів представлено на рисунку 4.4.

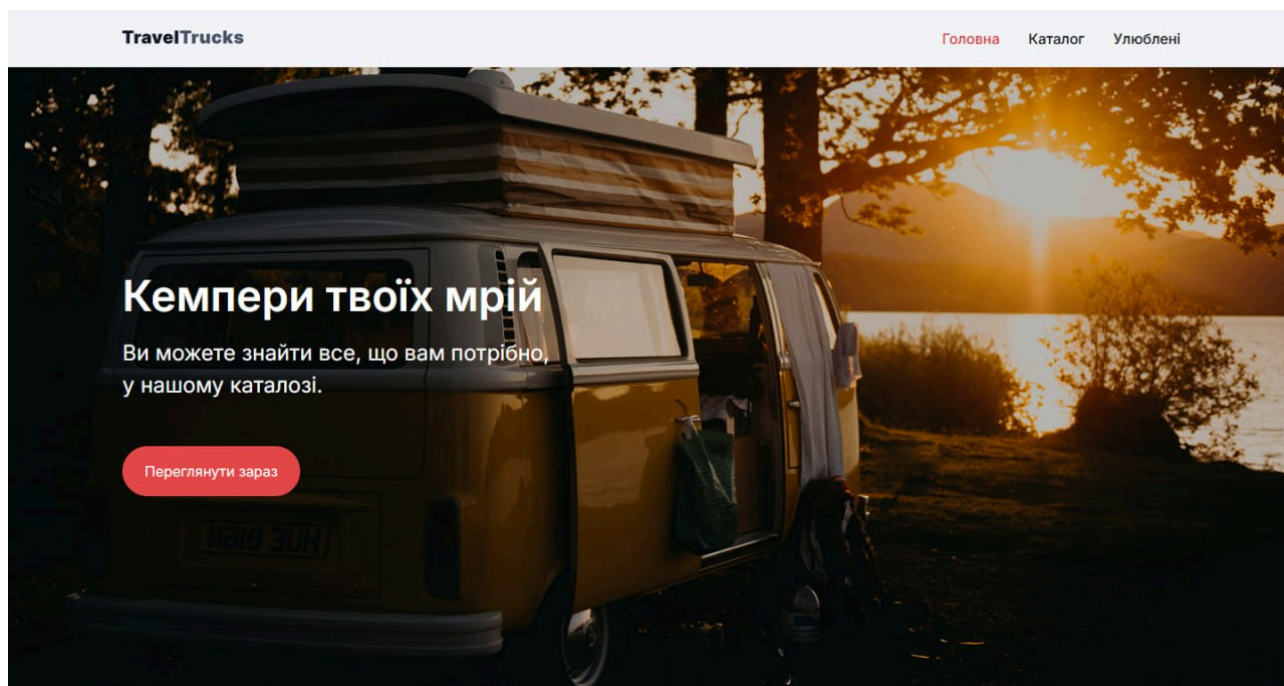


Рисунок 4.4 – Головний екран мобільного застосунку

Таким чином, користувач отримує зручний та зрозумілий інтерфейс, що дозволяє швидко орієнтуватися у функціоналі вебсистеми та ефективно здійснювати бронювання.

4.3 Висновки

У четвертому розділі проведено тестування програмних модулів мобільного застосунку для підтримки діяльності благодійних організацій. Перевірено реакцію програмних засобів на помилкові ситуації, вхідні дані різних форматів. Також, розроблено інструкцію користувача по початковій експлуатації програмного продукту.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі було розроблено вебсистему резервування автомобілів для мандрівників. Для створення програмного виробу застосовувалося сучасне середовище розробки Visual Studio Code з використанням стеку технологій JavaScript, React, Node.js та MongoDB.

У процесі виконання праці проведено аналіз поточного стану ринку вебдодатків для найму автомобілів, розглянуто аналоги, визначено їхні вади, що дозволило сформулювати перелік функціональних потреб до власної системи. На основі цього сформульовано завдання та побудовано архітектуру вебсистеми.

Під час аналізу технологій розробки було обгрунтовано вибір мов програмування JavaScript та TypeScript, бібліотеки React для клієнтської частини, Node.js з Express для серверної частини, а також бази даних MongoDB.

У бакалаврській кваліфікаційній роботі було зроблено наступне:

- розроблено модуль перегляду та фільтрації автомобілів за заданими параметрами;
- реалізовано модуль відображення детальної інформації про автомобіль, включаючи опис, характеристики, відгуки та форму бронювання;
- створено функціонал додавання автомобілів до обраного;
- реалізовано модуль оформлення бронювання з перевіркою даних, обробкою платіжної інформації та повідомленнями про статус заявки;
- розроблено алгоритм захисту даних платіжної картки з використанням генерації токена;
- створено зручний та адаптивний інтерфейс вебзастосунку;
- проведено тестування функціоналу системи відповідно до поставлених задач.

Розроблено діаграму структури інтерфейсу вебзастосунку, а також модель системи у вигляді UML-діаграми діяльності. Додатково створено блок-схему алгоритму обробки платіжних даних.

Тестування вебсистеми підтвердило її повну працездатність та відповідність технічному завданню. Також розроблено інструкцію користувача для зручної навігації та використання функцій системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гринченко В.І. Веб-технології та їх застосування в розробці інформаційних систем / В.І. Гринченко, О.С. Олейник. – Київ: Видавництво «Освіта», 2021. – 248 с.
2. Сидоренко І.М. Інформаційні системи в логістиці та транспортуванні / І.М. Сидоренко. – Львів: Видавництво ЛНУ, 2020. – 212 с.
3. Литвиненко В.С. Розробка веб-додатків: сучасні технології та інструменти / В.С. Литвиненко, А.І. Яковенко. - Київ: КНУ, 2021. - 276 с.
4. Rentalcars [Електронний ресурс] – Режим доступу до ресурсу: <https://www.rentalcars.com> (дата звернення: 5.03.2025).
5. RentalCarRadar [Електронний ресурс] – Режим доступу до ресурсу: <https://therentalradar.com/> (дата звернення: 5.03.2025).
6. Avis. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.avis.com/> (дата звернення: 5.03.2025).
7. Economic Books [Електронний ресурс] – Режим доступу до ресурсу: <https://www.economybookings.com/>.
8. Localrent [Електронний ресурс] – Режим доступу до ресурсу: <https://localrent.com/>.
9. EdrawMax User Guide [Електронний ресурс] – Режим доступу до ресурсу: <https://www.edrawsoft.com/edraw-max/>.
10. React [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/>.
11. TypeScript [Електронний ресурс] – Режим доступу до ресурсу: <https://www.typescriptlang.org/docs/>.
12. Framework Next.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs>.
13. REST API [Електронний ресурс] – Режим доступу до ресурсу: <https://goit.global/ua/articles/rest-api-shcho-tse-iak-pratsiuie-perevahy-i-prykklady/>.

14. Node.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/docs/latest/api/>.
15. MongoDB [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mongodb.com/docs/>.
16. CSS модулі [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/CSS>.
17. DatePicker [Електронний ресурс] – Режим доступу до ресурсу: <https://reactdatepicker.com/>.
18. VS Code [Електронний ресурс] – Режим доступу до ресурсу: <https://code.visualstudio.com/>.
19. Software testing [Електронний ресурс] – Режим доступу до ресурсу: <http://surl.li/uehab>.
20. Functional testing [Електронний ресурс] – Режим доступу до ресурсу: <http://surl.li/uehag>.
21. Failure testing [Електронний ресурс] – Режим доступу до ресурсу: <http://surl.li/tkely>.
22. Compatibility testing [Електронний ресурс] – Режим доступу до ресурсу: <http://surl.li/ueham>.

ДОДАТОК А
(обов'язковий)

61

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

д.т.н., проф.

О. Н. Романюк

"21" березня 2024 р.

Технічне завдання

на бакалаврську кваліфікаційну роботу «Розробка вебсистеми бронювання
автомобілів для мандрівників» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:

к.т.н., ст. викладач О.Я. Стахов

"21" 03 2025 р.

Виконав:

студент гр. ІПІ-216 М.В. Мельничук

"21" 03 2025 р.

Вінниця ВНТУ - 2025 рік

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка вебсистема бронювання автомобілів для мандрівників».

Галузь застосування – веб технології.

2. Підстава для розробки.

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від «20» березня 2025 р. ректора ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки.

Метою роботи є розробка програмного продукту для створення вебсистеми бронювання автомобілів з акцентом на швидкість роботи, точності та комфорту взаємодії користувача з системи.

Основними задачами дослідження є:

- розробка бази даних для зберігання інформації про автомобілі, замовлення та користувачів;
- створення інтерфейсу користувача, що забезпечить зручну взаємодію з вебсистемою та доступ до функцій пошуку, фільтрації і бронювання;
- реалізація модуля перегляду детальної інформації про кожен автомобіль;
- розробка форми бронювання з перевіркою введених даних та вибором способу оплати;
- розробка механізму збереження даних бронювання та підтвердження замовлення.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БКР:

1. React [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/>.
2. TypeScript [Електронний ресурс] – Режим доступу до ресурсу: <https://www.typescriptlang.org/docs/>.
3. Framework Next.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs>.
4. Node.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/docs/latest/api/>.
5. MongoDB [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mongodb.com/docs/>.

5. Технічні вимоги

- середовище розробки – Visual Studio Code;
- мова програмування – TypeScript;
- вхідні дані: інформація про доступні автомобілі та введені користувачем параметри фільтрації і бронювання;
- вихідні дані: застосунок із графічним користувацьким інтерфейсом для бронювання автомобілів.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БКР;
2. Технічне завдання;
3. Лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану питання та постановка задачі	25.03.25 – 31.03.25
2	Розробка моделей, структури та алгоритмів	01.04.25 – 10.04.25
3	Розробка програмного забезпечення	11.04.25 – 28.04.25
4	Тестування застосунку	29.04.25 – 15.05.25
5	Оформлення матеріалів до захисту БКР	16.05.25 – 30.05.25

10. Порядок контролю та прийняття

Порядок контролю і приймання роботи регламентується відповідними документами ВНТУ і державними стандартами.

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка вебсистеми бронювання автомобілів для мандрівників
 Тип роботи: бакалаврська кваліфікаційна робота
 (бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)
 Підрозділ кафедра програмного забезпечення, ФІТКІ, ІПІ-216
 (кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 8,2 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ☒ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- ☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____	_____
(прізвище, ініціали, посада)	(підпис)
_____	_____
(прізвище, ініціали, посада)	(підпис)
Особа, відповідальна за перевірку _____	Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____	Стахов О. Я., к. т. н., ст. викладач
(підпис)	(прізвище, ініціали, посада)
Здобувач _____	Мельничук М. В.
(підпис)	(прізвище, ініціали)

ДОДАТОК В

(довідниковий)

Лістинг програмного коду вебсистеми

Модуль Home.tsx

```
import Container from "@components/Container/Container";
import "../page.css"
import ButtonLink from "@components/Button/ButtonLink";

export default function Home() {
  return (
    <main>
      <section className="primary">
        <Container>
          <h1>Кемпери твоїх мрій</h1>
          <p>Ви можете знайти все, що вам потрібно, у нашому каталозі.</p>
          <ButtonLink href="/catalog">
            Переглянути зараз
          </ButtonLink>
        </Container>
      </section>
    </main>
  );
}

import Container from "@components/Container/Container";
import "../page.css"
```

```

import ButtonLink from "@components/Button/ButtonLink";

export default function Home() {
  return (
    <main>
      <section className="primary">
        <Container>
          <h1>Кемпери твоїх мрій</h1>
          <p>Ви можете знайти все, що вам потрібно, у нашому каталозі.</p>
          <ButtonLink href="/catalog">
            Переглянути зараз
          </ButtonLink>
        </Container>
      </section>
    </main>
  );
}

```

Модуль Header.tsx

```

"use client"

import Link from 'next/link'
import { usePathname } from 'next/navigation';
import s from './Header.module.css'
import Logo from '../Logo/Logo';
import Container from '../Container/Container';

```



```

export default function Header(){
  const pathname = usePathname();
  return(
    <header className={s.header}>
      <Container className={s.headerContainer}>
        <Logo />
        <nav className={s.navMenu}>
          <Link href="/" className={pathname === '/' ? s.active : "}
>Головна</Link>
          <Link href="/catalog" className={pathname === '/catalog' ?
s.active : "}>Каталог</Link>
          <Link href="/favorites" className={pathname === '/favorites' ?
s.active : "}>Улюблені</Link>
        </nav>
      </Container>
    </header>
  );
}
.header {
  position: sticky;
  top: 0;
  z-index: 3;
  min-height: 60px;
  background-color: #f2f4f7;
  border: var(--border);
  transition: background-color 500ms linear;
}

.headerContainer{
  display: flex;

```

```
justify-content: space-between;
align-items: center;
width: 100%;
margin: 0 auto;
padding: 20px;
}
```

```
.navMenu{
  display: flex;
  gap: 8px;
}
```

```
@media(min-width:769px){
  .navMenu{
    gap: 32px;
  }
}
```

```
.navMenu a{
  font-size: 14px;
  line-height: 1.5;
  font-weight: 500;
}
```

```
.navMenu a{
  font-size: 16px;
}
```

```
.navMenu a:hover, .navMenu a:focus{
  color: #D84343;
}
```

```
.active{
  color: #D84343;
}
```

Модуль Catalog.tsx

```
"use client";
```

```
import { useEffect, useState } from 'react';
import CarList from '@components/CarList/CarList';
import Container from '@components/Container/Container';
import Filter from '@components/Filter/Filter';
import s from './page.module.css';
```

```
interface Camper {
  id: string;
  name: string;
  price: number;
  rating: number;
  reviews: number;
  location: string;
  description: string;
  features: { icon: string; label: string }[];
  image: string;
  type: string;
}
```

```

export default function Catalog() {
  const [campers, setCampers] = useState<Camper[]>([]);
  const [filteredCampers, setFilteredCampers] = useState<Camper[]>([]);

  useEffect(() => {
    async function fetchCampers() {
      try {
        const response = await fetch('/api/campers');
        if (!response.ok) {
          throw new Error('Failed to fetch campers');
        }
        const data = await response.json();
        setCampers(data);
        setFilteredCampers(data); // Встановлюємо початковий список
      } catch (error) {
        console.error("Помилка отримання кемперів:", error);
      }
    }

    fetchCampers();
  }, []);

  const handleFilter = (location: string, equipment: string[], sortPrice: string) =>
  {
    let filtered = [...campers];

    if (location) {
      filtered = filtered.filter((camper) =>

```

```

        camper.location.toLowerCase().includes(location.toLowerCase())
    );
}

if (equipment.length > 0) {
    filtered = filtered.filter((camper) =>
        equipment.every((item) => camper.features.some((feature) => feature.label
=== item))
    );
}

if (sortPrice === "asc") {
    filtered.sort((a, b) => a.price - b.price);
} else if (sortPrice === "desc") {
    filtered.sort((a, b) => b.price - a.price);
}

setFilteredCampers(filtered);
};

return (
    <section>
        <Container className={s.catalogContainer}>
            <Filter onFilter={handleFilter} />
            <CarList cars={filteredCampers} />
        </Container>
    </section>

```

```
);
}
```

Модуль CarItem.tsx

```
import Image from 'next/image';
import ButtonLink from '../Button/ButtonLink';
import Features from '../Features/Features';
import Icon from '../Icon/Icon';
import s from './CarItem.module.css';
```

```
interface CarItemProps {
  id: string;
  name: string;
  price: number;
  rating: number;
  reviews: number;
  location: string;
  description: string;
  features: { icon: string; label: string }[];
  image: string;
}
```

```
export default function CarItem({
  id,
  name,
  price,
  rating,
  reviews,
  location,
```

```

description,
features,
image,
}): CarItemProps) {
  return (
    <li className={s.cardContainer}>
      <div className={s.imageWrapper}>
        <Image
          src={image}
          alt={name}
          width={290}
          height={365}
          className={s.image}
        />
      </div>

      <div className={s.infoCard}>
        <div className={s.headerCard}>
          <h3 className={s.title}>{name}</h3>
          <p className={s.price}>{price.toLocaleString()} ₪</p>
          <Icon name="heart" size={24} />
        </div>

        <div className={s.meta}>
          <span className={s.rating}>
            <span className={s.star}>★</span> {rating} <span>({reviews}
Відгуки)</span>
          </span>

```

```

    <span className={s.location}>
      <Icon name="map" size={20} /> {location}
    </span>
  </div>

  <p className={s.description}>{description}</p>

  <Features features={features} />

  <ButtonLink className={s.moreBtn} href={` /catalog/${id} `}>
    Дізнатися більше
  </ButtonLink>
</div>
</li>
);
}

```

Модуль CarList.tsx

```

import { useState } from "react";
import CarItem from "@components/CarItem/CarItem";
import Button from "@components/Button/Button";
import s from './CarList.module.css';

interface Car {
  id: string;
  name: string;
  price: number;
  rating: number;
  reviews: number;
}

```



```

location: string;
description: string;
features: { icon: string, label: string }[];
image: string;
}

```

```

interface CarListProps {
  cars: Car[];
}

```

```

export default function CarList({ cars }: CarListProps) {
  const [visibleCount, setVisibleCount] = useState(3);

```

```

  const handleLoadMore = () => {
    setVisibleCount((prevCount) => prevCount + 3);
  };

```

```

  return (
    <div>
      <ul className={s.carList}>
        {cars.slice(0, visibleCount).map((car) => (
          <CarItem key={car.id} {...car} />
        ))}
      </ul>
      {visibleCount < cars.length && (
        <div className={s.loadMoreContainer}>
          <Button onClick={handleLoadMore} className={s.loadMoreBtn}>
            Завантажити більше
          </Button>
        </div>
      )}
    </div>
  );
}

```

```

        </Button>
      </div>
    )}
  </div>
);
}

```

Модуль Filter.tsx

```

"use client";

import { useState } from "react";
import Input from "../Input/Input";
import Icon from "../Icon/Icon";
import Button from "../Button/Button";
import s from "../Filter.module.css";

const equipmentOptions = [
  { name: "Автоматичний", icon: "automatic" },
  { name: "Ручна", icon: "automatic" },
  { name: "AC", icon: "wind" },
  { name: "Кухня", icon: "kitchen" },
  { name: "TV", icon: "tv" },
  { name: "Ванна кімната", icon: "bathroom" },
];

export default function Filter({ onFilter }: { onFilter: (location: string,
equipment: string[], sortPrice: string) => void }) {
  const [location, setLocation] = useState("");
  const [selectedEquipment, setSelectedEquipment] = useState<string[]>([]);

```

```
const [sortPrice, setSortPrice] = useState<string>("");
```

```
const toggleEquipment = (item: string) => {
  setSelectedEquipment(prev =>
    prev.includes(item) ? prev.filter(i => i !== item) : [...prev, item]
  );
};
```

```
const handleSearch = () => {
  onFilter(location, selectedEquipment, sortPrice);
};
```

```
// Скидання всіх фільтрів
```

```
const handleReset = () => {
  setLocation("");
  setSelectedEquipment([]);
  setSortPrice("");
  onFilter("", [], "");
};
```

```
// Зміна сортування за ціною
```

```
const handleSort = (order: string) => {
  setSortPrice(order);
};
```

```
return (
  <form className={s.panel}>
    <h3 className={s.subheading}>Фільтри</h3>
```

```

<label className={s.locationLabel}>
  <span className={s.fieldTitle}>Місце</span>
  <div className={s.inputWrapper}>
    <Icon name="map" className={s.icon} size={20} />
    <Input
      type="search"
      name="location"
      placeholder="Місто"
      value={location}
      onChange={e => setLocation(e.target.value)}
      className={s.inputWithIcon}
    />
  </div>
</label>

<div className={s.filterGroup}>
  <h4 className={s.groupTitle}>Обладнання транспортного
засобу</h4>
  <div className={s.optionsGrid}>
    {equipmentOptions.map(opt => (
      <button
        key={opt.name}
        type="button"
        onClick={() => toggleEquipment(opt.name)}
        className={` ${s.optionBtn} ${
          selectedEquipment.includes(opt.name) ? s.selected : ""
        } `}
      >
        <Icon name={opt.icon} />

```

```

        <span>{opt.name}</span>
      </button>
    )}
  </div>
</div>

<div className={s.filterGroup}>
  <h4 className={s.groupTitle}>Сортування за ціною</h4>
  <div className={s.optionsGrid}>
    <button
      type="button"
      onClick={() => handleSort("asc")}
      className={` ${s.optionBtn} ${sortPrice === "asc" ? s.selected
: ""}` `}
    >
      Зростання
    </button>
    <button
      type="button"
      onClick={() => handleSort("desc")}
      className={` ${s.optionBtn} ${sortPrice === "desc" ?
s.selected : ""}` `}
    >
      Спадання
    </button>
  </div>
</div>

<div className={s.buttonGroup}>

```

```

        <Button onClick={handleSearch} className={s.searchBtn}>
            Пошук
        </Button>

        <Button onClick={handleReset} className={s.resetBtn}
type="button">
            Скинути фільтри
        </Button>
    </div>
</form>
);
}

```

Модуль Car.tsx

```

"use client";

import { useEffect, useState } from 'react';
import { useParams } from 'next/navigation';
import Icon from '@components/Icon/Icon';
import Features from '@components/Features/Features';
import CarSpecifications from '@components/CarSpecification/
CarSpecification';
import Reviews from '@components/Review/Review';
import BookingForm from '@components/BookingForm/BookingForm';
import Gallery from '@components/Gallery/Gallery';
import s from './page.module.css';
import Container from '@components/Container/Container';

interface Review {
    reviewer_name: string;
    reviewer_rating: number;
}

```

```

    comment: string;
}

```

```

interface Camper {
  id: string;
  name: string;
  price: number;
  rating: number;
  reviews: Review[];
  location: string;
  description: string;
  features: { icon: string; label: string }[];
  specifications: {
    form: string;
    length: string;
    width: string;
    height: string;
    tank: string;
    consumption: string;
  };
  gallery: { thumb: string, original: string }[];
}

```

```

export default function CarDetails() {
  const { id } = useParams();
  const [camper, setCamper] = useState<Camper | null>(null);
  const [activeTab, setActiveTab] = useState<'features' | 'reviews'>('features');

```

```

useEffect(() => {
  async function fetchCamper() {
    try {
      const response = await fetch(`/api/campers/${id}`);
      if (!response.ok) throw new Error('Не вдалося завантажити дані');
      const data = await response.json();
      setCamper(data);
    } catch (error) {
      console.error('Помилка завантаження автомобіля:', error);
    }
  }

  fetchCamper();
}, [id]);

```

```

if (!camper) return <p>Завантаження...</p>;

```

```

return (
  <Container>
    <div className={s.detailsContainer}>
      <h1 className={s.title}>{camper.name}</h1>
      <div className={s.meta}>
        <span className={s.rating}>
          <Icon name='star' size={16}/> {camper.rating}
          <span> ({camper.reviews.length} Відгуків)</span>
        </span>
        <span className={s.location}>
          <Icon name="map" size={16} /> {camper.location}

```



```

    </span>
  </div>

  <p className={s.price}>{camper.price.toLocaleString()}</p>

  { /* Використовуємо компонент Gallery */ }
  <Gallery images={camper.gallery} alt={camper.name} />

  <p className={s.description}>{camper.description}</p>

  <div className={s.detailsTabs}>
    <div className={s.tabs}>
      <button
        className={` ${s.tabButton} ${activeTab === 'features' ? s.active : ''}
        onClick={() => setActiveTab('features')}
      >
        Особливості
      </button>
      <button
        className={` ${s.tabButton} ${activeTab === 'reviews' ? s.active : ''}
        onClick={() => setActiveTab('reviews')}
      >
        Відгуки
      </button>
    </div>
  </div>

  <div className={s.contentWrapper}>

```

```

{activeTab === 'features' ? (
  <div className={s.featuresContainer}>
    <Features features={camper.features} />
    <CarSpecifications
      form={camper.specifications.form}
      length={camper.specifications.length}
      width={camper.specifications.width}
      height={camper.specifications.height}
      tank={camper.specifications.tank}
      consumption={camper.specifications.consumption}
    />
  </div>
) : (
  <div className={s.reviewsContainer}>
    {camper.reviews.length > 0 ? (
      <Reviews reviews={camper.reviews} />
    ) : (
      <p>Наразі відгуків немає.</p>
    )}
  </div>
)}
<div className={s.bookingForm}>
  <BookingForm camperId={camper.id} />
</div>
</div>
</Container>
);

```

```
}
```

Модуль BookingForm.tsx

```
"use client";
```

```
import { useState } from 'react';
```

```
import DatePicker from 'react-datepicker';
```

```
import { toast, ToastContainer } from 'react-toastify';
```

```
import 'react-toastify/dist/ReactToastify.css';
```

```
import Button from '../Button/Button';
```

```
import s from './BookingForm.module.css';
```

```
interface BookingFormProps {
```

```
  camperId: string;
```

```
}
```

```
export default function BookingForm({ camperId }: BookingFormProps) {
```

```
  const [name, setName] = useState("");
```

```
  const [email, setEmail] = useState("");
```

```
  const [phone, setPhone] = useState('+380');
```

```
  const [startDate, setStartDate] = useState<Date | null>(null);
```

```
  const [endDate, setEndDate] = useState<Date | null>(null);
```

```
  const [comment, setComment] = useState("");
```

```
  const [paymentMethod, setPaymentMethod] = useState('cash');
```

```
  const [cardNumber, setCardNumber] = useState("");
```

```
  const [cardExpiry, setCardExpiry] = useState("");
```

```
  const [cardCVV, setCardCVV] = useState("");
```

```
  const [errors, setErrors] = useState<{ [key: string]: string }>({});
```

```
const today = new Date();
```

```
// Валідація email
```

```
// Валідація email
```

```
const isValidEmail = (email: string) => {
  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  return emailRegex.test(email);
};
```

```
// Валідація телефону
```

```
const isValidPhone = (phone: string) => {
  const phoneRegex = /^\\+?[\\d\\s-()]{10,12}$/;
  return phoneRegex.test(phone);
};
```

```
// Валідація номера картки (16 цифр з можливими пробілами)
```

```
const isValidCardNumber = (cardNumber: string) => {
  const cardRegex = /^(\\d{4}[\\s-]?){3}\\d{4}$/;
  return cardRegex.test(cardNumber);
};
```

```
// Валідація терміну дії картки (формат ММ/YY)
```

```
const isValidCardExpiry = (expiry: string) => {
  // Перевіряємо формат ММ/YY
  const expiryRegex = /^(0[1-9]|1[0-2])\\d{2}$/;
  if (!expiryRegex.test(expiry)) return false;
```

```

// Розділяємо місяць і рік
const [month, year] = expiry.split('/').map(Number);

// Отримуємо поточний місяць та рік
const now = new Date();
const currentMonth = now.getMonth() + 1; // місяць від 0 до 11
const currentYear = Number(String(now.getFullYear()).slice(2)); // останні 2
цифри року

// Перевіряємо чи рік дійсний та не в минулому
if (year < currentYear) return false;

// Якщо рік той самий, перевіряємо місяць
if (year === currentYear && month < currentMonth) return false;

return true;
};

// Валідація CVV (3 цифри)
const isValidCVV = (cvv: string) => {
  const cvvRegex = /^\d{3}$/;
  return cvvRegex.test(cvv);
};

// Валідація дат (перевірка на наявність та коректність формату)
const isValidDateRange = (startDate: Date | null, endDate: Date | null) => {
  if (!startDate || !endDate) return false;
  return startDate <= endDate;
};

```

```

// Оновлення стану помилок
const updateErrors = () => {
  const newErrors: { [key: string]: string } = {};

  if (!name) newErrors.name = 'Ім'я є обов'язковим';
  if (!isValidEmail(email)) newErrors.email = 'Неправильний формат email';
  if (!isValidPhone(phone)) newErrors.phone = 'Неправильний формат телефону (10-12 цифр)';
  if (!isValidDateRange(startDate, endDate)) newErrors.date = 'Дата початку має бути раніше дати завершення';

  // Додаткові перевірки для картки, якщо вибраний спосіб оплати "card"
  if (paymentMethod === 'card') {
    if (!isValidCardNumber(cardNumber)) newErrors.cardNumber = 'Некоректний номер картки (16 цифр)';
    if (!isValidCardExpiry(cardExpiry)) newErrors.cardExpiry = 'Некоректний термін дії (ММ/YY або минула дата)';
    if (!isValidCVV(cardCVV)) newErrors.cardCVV = 'Некоректний CVV (3 цифри)';
  }

  // Валідація діапазону дат
  if (!isValidDateRange(startDate, endDate)) {
    newErrors.date = 'Дата початку має бути раніше дати завершення';
  }

  setErrors(newErrors);

```

```
return newErrors;
};
```

```
// Форматування номера картки (додає пробіл кожні 4 цифри)
```

```
const formatCardNumber = (value: string) => {
  return value
    .replace(/\D/g, "")
    .replace(/(.{4})/g, '$1 ')
    .trim();
};
```

```
// Форматування терміну дії (додає слеш після двох цифр)
```

```
const formatCardExpiry = (value: string) => {
  return value
    .replace(/\D/g, "") // Видаляємо всі нецифрові символи
    .replace(/^(\d{2})(\d)/, '$1/$2') // Додаємо слеш після двох цифр
    .slice(0, 5); // Обмежуємо довжину до 5 символів
};
```

```
const handleCardNumberChange = (e:
React.ChangeEvent<HTMLInputElement>) => {
  setCardNumber(formatCardNumber(e.target.value));
};
```

```
const handleCardExpiryChange = (e:
React.ChangeEvent<HTMLInputElement>) => {
  setCardExpiry(formatCardExpiry(e.target.value));
};
```

```
const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  const validationErrors = updateErrors();
  if (Object.keys(validationErrors).length > 0) return;
```

```
const bookingData = {
  camperId,
  name,
  email,
  phone,
  startDate: startDate?.toISOString().split('T')[0],
  endDate: endDate?.toISOString().split('T')[0],
  comment,
  paymentMethod,
};
```

```
try {
  const response = await fetch('/api/bookings', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(bookingData),
  });
```

```
if (!response.ok) {
  let errorMessage = 'Помилка при бронюванні';
  try {
    // Працюємо з відповіддю у форматі JSON
    const error = await response.json();
```



```

        errorMessage = error.message || error.error || 'Сталася помилка';
    } catch {
        // Якщо відповідь не є JSON, зчитуємо як текст
        const errorText = await response.text();
        errorMessage = errorText || 'Сталася невідома помилка';
    }
    toast.error(`Помилка: ${errorMessage}`);
    return;
}

toast.success('Бронювання успішно завершено!');
setName("");
setEmail("");
setPhone('+380');
setStartDate(null);
setEndDate(null);
setComment("");
setPaymentMethod('cash');
setCardNumber("");
setCardExpiry("");
setCardCVV("");
} catch (error) {
    console.error('Помилка при бронюванні:', error);
    toast.error('Не вдалося здійснити бронювання.');
```

```

return (
  <div className={s.bookingForm}>
    <h3>Забронюйте свій кемпер вже зараз</h3>
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        placeholder="Ім'я*"
        className={s.input}
        value={name}
        onChange={(e) => setName(e.target.value)}
        required
      />
      {errors.name && <span className={s.errorText}>{errors.name}</
span>}}

```

```

    <input
      type="email"
      placeholder="Email*"
      className={s.input}
      value={email}
      onChange={(e) => setEmail(e.target.value)}
      required
    />
    {errors.email && <span className={s.error}>{errors.email}</span>}}

```

```

    <input
      type="tel"
      placeholder="Номер телефону*"
      className={s.input}

```

```

value={phone}
onChange={(e) => setPhone(e.target.value)}
required
/>

{errors.phone && <span className={s.error}>{errors.phone}</span>}

<div className={s.datePickers}>
  <DatePicker
    selected={startDate}
    onChange={(date) => setStartDate(date)}
    minDate={today}
    placeholderText="Початок бронювання*"
    className={s.input}
    dateFormat="yyyy-MM-dd"
    required
  />
  <DatePicker
    selected={endDate}
    onChange={(date) => setEndDate(date)}
    minDate={startDate || today}
    placeholderText="Кінець бронювання*"
    className={s.input}
    dateFormat="yyyy-MM-dd"
    required
  />
  {errors.date && <span className={s.error}>{errors.date}</span>}

</div>

```

```

<textarea
  placeholder="Коментар"
  className={s.textarea}
  value={comment}
  onChange={(e) => setComment(e.target.value)}
/>

<div className={s.paymentMethods}>
  <label>
    <input
      type="radio"
      name="payment"
      value="cash"
      checked={paymentMethod === 'cash'}
      onChange={() => setPaymentMethod('cash')}
    />
    Готівка
  </label>
  <label>
    <input
      type="radio"
      name="payment"
      value="card"
      checked={paymentMethod === 'card'}
      onChange={() => setPaymentMethod('card')}
    />
    Картка
  </label>
</div>

```

```

{paymentMethod === 'card' && (
  <div className={s.cardDetails}>
    <input
      type="text"
      placeholder="Номер картки"
      maxLength={19}
      className={s.input}
      value={cardNumber}
      onChange={handleCardNumberChange}
    />
    {errors.cardNumber && <span className={s.error}>
      >{errors.cardNumber}</span>}

```

```

    <input
      type="text"
      placeholder="Термін дії (ММ/YY)"
      maxLength={5}
      className={s.input}
      value={cardExpiry}
      onChange={handleCardExpiryChange}
    />
    {errors.cardExpiry && <span className={s.error}>
      >{errors.cardExpiry}</span>}

```

```

    <input
      type="text"
      placeholder="CVV"
      maxLength={3}

```

```

        className={s.input}
        value={cardCVV}
        onChange={(e) => setCardCVV(e.target.value)}
      />
      {errors.cardCVV && <span className={s.error}>{errors.cardCVV}</
span>}

```

```

    </div>

```

```

  )}

```

```

    <Button type="submit" className={s.submitButton}>

```

```

      Надіслати

```

```

    </Button>

```

```

  </form>

```

```

  <ToastContainer position="top-right" autoClose={3000} />

```

```

</div>

```

```

);

```

```

}

```

```

.bookingForm{

```

```

  border: 1px solid #DADDE1;

```

```

  border-radius: 10px;

```

```

  padding: 44px;

```

```

  width: 100%;

```

```

}

```

```

.bookingForm h3{

```

```

  font-size: 20px;

```

```

  font-weight: 600;

```

```
    line-height: 1.2;
    margin-bottom: 5px;
}
```

```
.bookingForm p{
    font-size: 16px;
    line-height: 1.5;
    color: #6C717B;
}
```

```
.bookingForm form{
    margin-top: 24px;
    display: flex;
    flex-direction: column;
    gap: 14px;
}
```

```
.bookingForm input, .bookingForm textarea{
    width: 100%;
    color: #000;
    background-color: #F7F7F7;
    border-radius: 10px;
    padding: 18px;
    font-size: 16px;
    line-height: 1.5;
}
```

```
.datePickers{
```

```
width: 100%;  
display: flex;  
flex-direction: column;  
gap: 14px;  
}
```

```
.submitButton{  
  margin-top: 24px;  
}
```

```
.paymentMethods {  
  display: flex;  
  gap: 8px;  
  margin: 12px 0;  
  flex-wrap: wrap;  
}
```

```
.paymentMethods label {  
  display: flex;  
  align-items: center;  
  gap: 4px;  
  font-size: 16px;  
  cursor: pointer;  
}
```

```
.paymentMethods input[type="radio"] {  
  appearance: none;  
  width: 20px;
```



```
height: 20px;
border: 2px solid #ccc;
border-radius: 50%;
margin-right: 4px;
cursor: pointer;
transition: all 0.2s ease;
position: relative;
padding: 0;
}

.paymentMethods input[type="radio"]:checked {
  background-color: #4285f4;
  border-color: #4285f4;
}

.paymentMethods input[type="radio"]::before {
  content: "";
  position: absolute;
  top: 50%;
  left: 50%;
  width: 6px;
  height: 6px;
  background-color: white;
  border-radius: 50%;
  transform: translate(-50%, -50%) scale(0);
  transition: transform 0.2s ease;
}
```

```
.paymentMethods input[type="radio"]:checked::before {  
  transform: translate(-50%, -50%) scale(1);  
}
```

```
.paymentMethods input[type="radio"]:hover {  
  border-color: #4285f4;  
}
```

```
/* Desktop styles */
```

```
@media (min-width: 768px) {  
  .paymentMethods {  
    gap: 10px;  
  }  
}
```

```
.paymentMethods label {  
  font-size: 16px;  
}
```

```
.paymentMethods input[type="radio"] {  
  width: 16px;  
  height: 16px;  
}
```

```
.paymentMethods input[type="radio"]::before {  
  width: 6px;  
  height: 6px;  
}  
}
```

```
.cardDetails{  
  display: flex;  
  flex-direction: column;  
  gap: 14px;  
}
```

```
.error {  
  color: red;  
  font-size: 14px;  
  margin: 4px 0;  
}
```

ДОДАТОК Г
(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА
РОЗРОБКА ВЕБСИСТЕМИ БРОНЮВАННЯ АВТОМОБІЛІВ ДЛЯ
МАНДРІВНИКІВ

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення



Бакалаврська дипломна робота на тему:
“Розробка вебсистеми бронювання автомобілів для мандрівників”

Автор: ст. групи ІПІ-216 Мельничук М. В.
Науковий керівник: к.н.т. ст. викладач Стахов О. Я.

Вінниця ВНТУ - 2025

Рисунок Г.1 – Титульний слайд

Актуальність теми

Сучасний туристичний світ та мандри вимагають ефективних інструментів для планування та бронювання транспорту. Збільшення автомобільних подорожей підкреслює важливість зручних вебсистем, що надають користувачам швидкий та безпечний спосіб бронювання автомобілів. Проте, наявні рішення часто не відповідають потребам мандрівників, адже не враховують особливі вимоги до зручності користування, опцій фільтрації та захисту інформації.

У відповідь на ці виклики, стає актуальною розробка програмного забезпечення вебсистеми, яка б спеціально відповідала потребам мандрівників. Ця система значно полегшить процес бронювання автомобілів, надаючи автоматизований пошук доступних автомобілів, інтуїтивну зрозумілу навігацію, фільтрацію за критеріями та збереження інформації про бронювання.

Розробка подібної вебсистеми становить собою ключовий етап на шляху покращення до полегшення доступу до транспортних засобів для мандрівників. Розробка надасть у користування засіб для бронювання автомобілів, що сприятиме зростанню ефективності планування подорожей та комфорту для користувачів. Така система сформує сприятливе середовище для оперативного вибору автобiля та оптимального маршруту, що є надзвичайно важливим аспектом для туристів та мандрівників.

Рисунок Г.2 – Актуальність теми

Мета, об'єкт та предмет дослідження

Мета та завдання дослідження. Метою роботи є збільшення ефективності та зручності процесу бронювання автомобіля для мандрівників, шляхом розробки вебсистеми бронювання транспортних засобів. Система полегшить оптимальний вибір автомобіля, надаючи швидкий доступ до актуальної інформації та зручне оформлення бронювання

Основними задачами дослідження є:

- розробка бази даних для зберігання автомобілів та бронювання;
- розробка інтерфейсу користувача, що забезпечить зручну взаємодію з застосунком та легкий доступ до необхідної інформації;
- розробка модуля пошуку та фільтрації автомобілів за параметрами (обладнання, ціна, місто тощо);
- розробка модуля оформлення бронювання.

Об'єкт дослідження – процес автоматизованого бронювання автомобілів через вебсайт.

Предмет дослідження – методи та засоби розробки програмних модулів для веб-систем бронювання автомобілів.

Рисунок Г.3 – Мета, об'єкт та предмет дослідження

Аналіз сучасних систем

- Існуючі вебсервіси з бронювання авто здебільшого орієнтовані на загальний ринок і не враховують специфічних потреб мандрівників.
- Більшість із них мають перевантажений інтерфейс, обмежену фільтрацію та недостатній рівень захисту особистих і платіжних даних користувачів.
- Необхідна система, що забезпечує просту навігацію, розширені фільтри (місце, ціна, обладнання), адаптивність та надійне збереження інформації про бронювання.

Рисунок Г.4 – Аналіз сучасних систем

Новизна отриманих результатів

Подальшого розвитку зазнав метод бронювання автомобілів через вебсистему, що відрізняється від наявних реалізацією інтуїтивного інтерфейсу для вибору машини та завершення бронювання. Запропоноване рішення дає змогу мандрівникам без зусиль знаходити та замовляти транспортні засоби, враховуючи різноманітні критерії, що покращує зручність взаємодії з системою. Розроблена вебсистема автоматизує процес бронювання, пропонуючи швидкий пошук автомобіля за допомогою фільтрів та зрозуміле оформлення замовлення.

Подальшого розвитку набув метод створення вебзастосунків для бронювання автомобілів, у якому застосовано технології: Next.js, TypeScript, Node.js, MongoDB та React. Це дозволило досягти високої продуктивності системи, адаптивності інтерфейсу та ефективної обробки великої кількості інформації про автомобіль.

Рисунок Г.5 – Новизна отриманих результатів

Практична цінність отриманих результатів

Практична цінність одержаних результатів полягає в тому, що на основі теоретичних положень у бакалаврській дипломній роботі було розроблено вебсистему для бронювання автомобілів. Запропоновані рішення дозволяють ефективно автоматизувати процес пошуку, фільтрації та бронювання транспортних засобів для мандрівників, що сприяє підвищенню зручності та якості туристичного сервісу.

Рисунок Г.6 – Практична цінність отриманих результатів

Алгоритм захисту даних платіжної платіжної картки

Алгоритм захисту платіжної інформації у вебсистемі базується на використанні Stripe SDK, що забезпечує безпечне шифрування та токенизацію даних. Дані платіжної картки передаються безпосередньо до Stripe, де відбувається їхнє шифрування і створення токена – цифрового ідентифікатора, який не містить реальних реквізитів. Після цього система перевіряє успішність зчитування, передає необхідні ключі та значення і формує токен для захищеної передачі. У разі успішного проходження усіх етапів токен зберігається у базі даних через захищене з'єднання. Якщо ж фіксується помилка, користувача повідомляють і запис не відбувається. Такий підхід дозволяє уникнути витоку чутливої інформації, зберігаючи надійність транзакцій у системі бронювання.

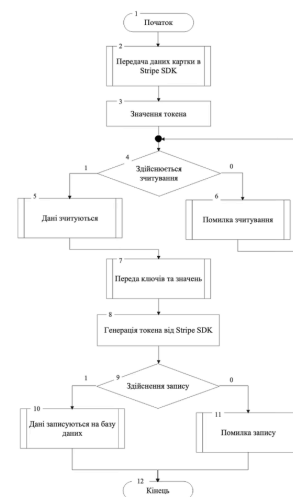


Рисунок Г.7 – Алгоритм захисту даних користувачів

Структура вебсистеми бронювання

Для розробки вебсистеми бронювання автомобілів була створена модель, що демонструє логіку взаємодії користувача з інтерфейсом. Модель описує послідовність дій: від відкриття головної сторінки до підтвердження бронювання. Це дозволяє краще зрозуміти структуру роботи вебзастосунку та передбачити етапи, на яких можливі помилки або затримки.

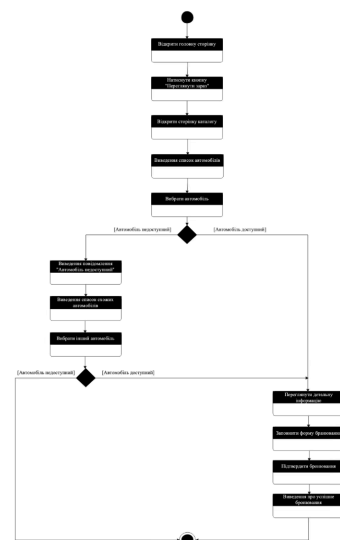


Рисунок Г.8 – Структура мобільної системи

Розробка графічного інтерфейсу

При розробці інтерфейсу вебсистеми бронювання автомобілів основну увагу було приділено його зручності та зрозумілості для користувача. Особливо важливою була задача створити візуально привабливий і функціонально зрозумілий інтерфейс, що дозволяє швидко орієнтуватися в системі та здійснювати бронювання з мінімальною кількістю кроків.

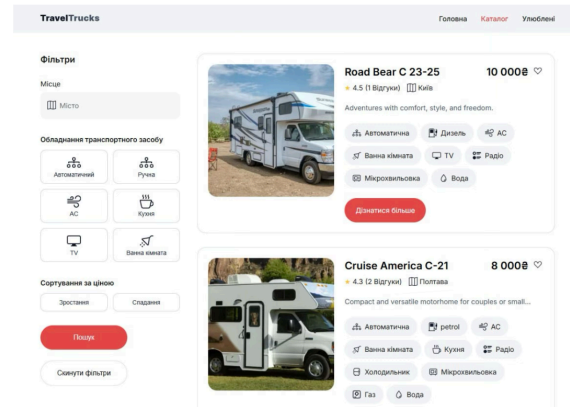


Рисунок Г.9 – Розробка графічного інтерфейсу

Тестування застосунку

Важливим аспектом тестування вебсистеми бронювання автомобілів було визначення її стійкості до типових помилок, таких як відсутність введених даних, некоректні формати або спроба бронювання без вибору авто. Перевірка проводилась з акцентом на зручність взаємодії та чіткість повідомлень про помилки. На слайді наведено приклади тестування модуля бронювання з демонстрацією валідації введених даних.

 The image shows two versions of a booking form. The left version is the 'Забронюйте свій кемпер вже зараз' (Book your motorhome now) form with fields for 'Ім'я*' (Name*), 'Електронна пошта*' (Email*), 'Номер телефону*' (Phone number*), 'Початок бронювання*' (Start booking*), 'Кінець бронювання*' (End booking*), 'Коментар' (Comment), and radio buttons for 'Готівка' (Cash), 'Карта' (Card), and 'Безготівковий розрахунок' (Non-cash payment). The right version shows the same form with validation errors: 'Некоректний формат email' (Incorrect email format) for the email field, 'Некоректний формат телефону (10-12 цифр)' (Incorrect phone format (10-12 digits)) for the phone field, 'Некоректний номер карти (16 цифр)' (Incorrect card number (16 digits)) for the card field, 'Некоректний термін дії (MM/YY або місяць/рік)' (Incorrect validity period (MM/YY or month/year)) for the expiry date field, and 'Некоректний CVV (3 цифри)' (Incorrect CVV (3 digits)) for the CVV field.

Рисунок Г.10 – Тестування застосунку

Висновки

У бакалаврській дипломній роботі було розроблено вебсистему бронювання автомобілів для мандрівників. Для реалізації використано сучасні інструменти: Visual Studio Code, JavaScript, TypeScript, React, Node.js, MongoDB.

Під час виконання роботи проаналізовано сучасний стан ринку, розглянуто аналогічні сервіси, виявлено їх недоліки та сформульовано вимоги до власної системи.

Обґрунтовано вибір технологій для frontend і backend частини, а також інструментів захисту платіжних даних.

- Реалізовано основні модулі системи: каталог авто, перегляд деталей, фільтрація, оформлення бронювання.
- Створено адаптивний інтерфейс, UML-діаграму діяльності, схему обробки платіжних даних.

Рисунок Г.11 – Висновки