

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

Веб сервіс для оренди авто

Виконав студент групи ІПЗ-21-4
КУЗЬМІН Владислав Андрійович

Керівник роботи:
ТОЛСТОЙ Ігор Анатолійович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2025

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

«ЗАТВЕРДЖУЮ»
Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк
«14» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КУЗЬМІН Владислав Андрійович**

Керівник роботи: **ТОЛСТОЙ Ігор Анатолійович**

Тема роботи: **«Веб сервіс для оренди авто»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2024 р., №61/с**

Вихідні дані для роботи: **процес надання онлайн послуг з оренди автомобілів через веб сервіс, реалізований за допомогою фреймворку Laravel та бібліотеки React, з використанням бази даних.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Толстой І.А.	27.02.2025р.	27.02.2025р.
2	Толстой І.А.	27.02.2025р.	27.02.2025р.
3	Толстой І.А.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з веб сервісу для оренди авто та пояснювальної записки. Пояснювальна записка до випускної роботи містить 70 сторінок, 19 ілюстрацій, 25 таблиць.

Метою роботи є розробка веб сервісу для оренди авто онлайн, що дозволяє користувачам здійснювати пошук, перегляд, бронювання авто. Для досягнення цієї мети було використано фреймворк Laravel для створення серверної частини застосунку, бібліотеку React для розробки інтерфейсу користувача та реляційну базу даних SQL для зберігання інформації про користувачів, транспортні засоби та замовлення.

Архітектура веб сервісу для оренди авто побудована на основі клієнт-серверної моделі, що забезпечує ефективну взаємодію між користувачами та сервером. Серверна частина відповідає за обробку запитів, збереження та обробку даних, а клієнтська частина відповідає за взаємодію користувача із сервісом через інтуїтивно зрозумілий інтерфейс.

Отриманими результатами є веб сервіс для оренди авто, який надає користувачам можливість переглядати список доступних автомобілів, реєструватися та входити в систему, здійснювати бронювання обраного автомобіля та переглядати інформацію про свої замовлення.

КЛЮЧОВІ СЛОВА: ВЕБСЕРВІС, ОRENDA, АВТОМОБІЛЬ, КОРИСТУВАЧ, АДМІН, LARAVEL, REACT, REST API.

					ІПЗ.КР.Б – 121 – 25 – ПЗ				
Зм.	Арк.	№ докум.	Підпис	Дата					
Розроб.		В.А. Кузьмін			Веб-сервіс для оренди авто		Літ.	Арк.	Аркушів
Перевір.								3	77
Керівник		І.А. Толстой					Житомирська політехніка, група ІПЗ-21-4		
Рецензент		Ю.М. Єфремов							
Зав. каф.		Т.А. Вакалюк							

ABSTRACT

The final qualifying work of a bachelor consists of a web service for car rental and an explanatory note. The explanatory note to the final thesis contains 70 pages, 19 illustrations, 25 tables.

The aim of the work is to develop a web service for booking cars online, which allows users to search, browse, and book a car. To achieve this goal, we used the Laravel framework to create the server side of the application, the React library to develop the user interface, and a SQL relational database to store information about users, vehicles, and orders.

The architecture of the web service is based on a client-server model, which ensures efficient interaction between users and the server. The server part is responsible for processing requests, storing and processing data, and the client part is responsible for user interaction with the service through an intuitive interface.

The obtained results are a web service for car rental that provides users with the ability to view a list of available cars, register and log in, make a reservation for the selected car and view information about their orders.

KEYWORDS: WEB SERVICE, RENTAL, CAR, USER, ADMIN, LARAVEL, REACT, REST API.

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1.ТЕОРЕТИЧНІ ЗАСАДИ ТА АНАЛІТИЧНЕ ОБГРУНТУВАННЯ СТВОРЕННЯ ВЕБ-СЕРВІСУ ДЛЯ ОРЕНДИ АВТОМОБІЛІВ.....	10
1.1 Аналіз стану розвитку цифрових сервісів у сфері автопрокату	10
1.2 Комплексний аналіз функціональних можливостей та недоліків існуючих веб-платформ	12
1.3 Визначення концептуальних вимог до архітектури інформаційної системи.....	20
Висновки до першого розділу	23
РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА РЕАЛІЗАЦІЯ ЇЇ ПРОГРАМНИХ КОМПОНЕНТІВ.....	24
2.1 Вибір архітектурно-технологічної платформи та інструментальних засобів	24
2.2 Формалізація функціональних сценаріїв та побудова об’єктно- орієнтованої моделі.....	28
2.3 Розроблення клієнтської та серверної частин веб-сервісу на основі фреймворків React і Laravel	40
2.4 Моделювання структури бази даних та організація взаємодії з API... ..	46
Висновки до другого розділу.....	55
РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ФУНКЦІОНУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ РОЗРОБНЕЛОГО ВЕБ-СЕРВІСУ	56
3.1 Організація тестування та опис методик перевірки працездатності системи.....	56
3.2 Оцінювання зручності використання, продуктивності та надійності сервісу	58
3.3 Напрями подальшого вдосконалення програмного продукту	62
Висновки до третього розділу	66
ВИСНОВКИ	71
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	72
ДОДАТКИ	76

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення.

БД – база даних.

API – прикладний програмний інтерфейс.

REST– архітектурний стиль взаємодії між веб-сервісами.

MVC – схема розділення даних додатків і керуючої логіки на три окремі компоненти Model-View-Controller.

SPA – динамічне оновлення контенту без повного перезавантаження сторінки.

CRUD – основні операції, що використовуються для управління даними (Create, Read, Update, Delete).

HTTP – протокол передачі гіпертексту.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

У сучасних умовах розвитку цифрових технологій попит на онлайн сервіси, що забезпечують зручність у повсякденному житті дуже зростає. Однією з таких сфер є оренда автомобілів, яка стає популярнішою завдяки мобільності, гнучкості та економічній доцільності для користувача. Створення ефективного веб-сервісу для оренди авто дозволить автоматизувати процеси пошуку, бронювання та управління транспортними засобами, що значно підвищить якість обслуговування та знизить витрати часу як для орендодавців, так і для клієнтів. Впровадження веб-сервісу дозволить створити єдину централізовану платформу, яка забезпечить прозору взаємодію між усіма учасниками процесу оренди. Завдяки використанню сучасних веб-технологій, сервіс може підтримувати різні типи користувачів, зокрема звичайних клієнтів та адміністраторів, надаючи кожному з них відповідний функціонал для зручної та безпечної роботи.

Актуальність теми обумовлена можливістю підвищити конкурентоспроможність бізнесу сфері транспортних послуг. На сьогоднішній день короткострокова оренда транспорту є дуже популярна в багатьох країнах, також галузь активно розвивається в Україні. Сучасні технології веб-програмування дозволяють створювати високопродуктивні, масштабовані та зручні системи для користувача та адміністратора, що відкриває нові можливості для бізнесу у сфері прокату транспорту.

Метою випускної кваліфікаційної роботи бакалавра є розробка сучасного веб-сервісу для оренди автомобілів, який дозволить ефективно здійснювати пошук та бронювання транспортних засобів, застосунок має забезпечити ефективну взаємодію між клієнтом та адміністратором системи через інтуїтивно зрозумілий інтерфейс, що відповідає вимогам користувачів і надійну серверну частину. Для реалізації поставленої мети потрібно виконати наступні завдання:

На початковому етапі необхідно провести огляд і аналіз сучасних програмних рішень, що вже функціонують у сфері надання послуг оренди

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

транспортних засобів. Це дозволить виявити найбільш затребувані функціональні можливості, визначити стандарти побудови архітектури таких систем та сформувані перелік функціональних і нефункціональних вимог.

Здійснити обґрунтований вибір технологічного стеку, який буде використовуватись у процесі розробки та обрати архітектурний підхід для реалізації клієнт-серверної взаємодії веб-сервісу для оренди авто. Наступним етапом є розробка клієнтської та серверної частини застосунку за допомогою сучасних веб-технологій, з використанням бази даних для зберігання інформації.

Провести експериментальне тестування, яке охоплює функціональні сценарії взаємодії, валідність обробки даних та продуктивність при підвищеному навантаженні. Підсумовуючи описати відповідність системи заявленим вимогам та скласти рекомендації для подальшого вдосконалення веб-сервісу.

Об'єктом дослідження є інформаційні системи та веб-застосунки, функціональне призначення яких полягає в автоматизації процесів надання послуг оренди транспортних засобів. У рамках дослідження вивчаються технологічні аспекти побудови таких систем, архітектурні підходи до їх реалізації, також ефективність взаємодії між клієнтом, адміністратором і базою даних з метою забезпечення стабільної та безпечної роботи сервісу.

Предметом дослідження є методологічні та технологічні засади проектування та розробки клієнт-серверного веб-додатку з використанням сучасних веб-технологій. Досліджуються принципи побудови API за допомогою фреймворку Laravel для реалізації управління маршрутизації, обробки HTTP запитів та взаємодії з базою даних. Також досліджуються підходи до побудови динамічного інтерфейсу користувача на основі JavaScript-бібліотеки React.

У процесі виконання кваліфікаційної роботи використовувались різні методології дослідження, які дозволили здійснити аналіз предметної області,

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

обґрунтувати вибір архітектурних рішень і здійснити перевірку функціональності розробленого веб-сервісу.

Метод аналізу передбачає розкладання об'єкта або процесу на складові частини з метою детального вивчення кожного елемента окремо. У дослідженні метод аналізу застосовується для вивчення вимог до веб-сервісу, структури подібних систем і функціональності наявних рішень.

Метод порівняння використовується для встановлення схожості і відмінності між об'єктами та рішеннями. У процесі дослідження він дає змогу зіставити існуючі сервіси з оренди автомобілів за різними критеріями, щоб здобути ефективні рішення та уникнути недоліків конкурентів.

Метод моделювання полягає у створенні спрощеного представлення системи, яка дозволяє досліджувати її властивості без прямого втручання в реальний об'єкт. У рамках цієї роботи застосовуються інформаційне, структурне та функціональне моделювання: побудова діаграм архітектури та взаємодії даних. Цей метод дозволяє описати логіку роботи системи до її реалізації та виявити потенційні проблеми на етапі проектування.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1.ТЕОРЕТИЧНІ ЗАСАДИ ТА АНАЛІТИЧНЕ ОБГРУНТУВАННЯ СТВОРЕННЯ ВЕБ-СЕРВІСУ ДЛЯ ОРЕНДИ АВТОМОБІЛІВ

1.1 Аналіз стану розвитку цифрових сервісів у сфері автопрокату

На сьогоднішній день стрімко розвивається ринок короткострокової оренди автомобілів, який потребує нових інструментів для оптимізації взаємодії між компаніями-орендодавцями та клієнтами. Традиційні методи оренди транспортних засобів через телефон та офлайн-офіси є застарілими, не зручними та часто пов'язані з витратами часу як для клієнтів, так і для персоналу. Провідні міжнародні компанії інвестують значні ресурси в розробку та вдосконалення цифрових платформ. Українські компанії, орієнтуючись на європейські стандарти, впроваджують автоматизовані рішення для обробки замовлень, управління автопарком та моніторингу клієнтської активності.

Веб-сервіси дозволяють зробити процес оренди значно ефективнішим, доступним та прозорим. Користувач може самостійно переглянути доступні автомобілі оцінювати їх характеристики та здійснювати бронювання в будь-який зручний час. Для адміністратора веб-застосунок відкриває нові можливості щодо оперативного управління автопарком та обліку замовлень.

Використання веб-сервісу відкриває можливості для впровадження додаткових функцій, таких як автоматизоване формування звітності, інтеграція з платіжними системами для онлайн-оплати, надсилення повідомлень клієнтам про стан замовлення, а також зберігання історії бронювань для зручності повторного користування. Це не лише спрощує управління внутрішніми процесами, а й підвищує рівень прозорості та довіри між клієнтом і компанією.

Завдяки адаптивному дизайну та оптимізації під мобільні пристрої, сервіс стає доступним з будь-якого місця та у будь-який час, що особливо важливо для сучасних користувачів, які цінують швидкість, зручність і автономність у прийнятті рішень. Таким чином, впровадження сучасного веб-сервісу для оренди автомобілів сприяє підвищенню конкурентоспроможності компанії, поліпшенню якості обслуговування та розширенню цільової аудиторії завдяки

послуг

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

Отже, цифрові сервіси у сфері прокату автомобілів знаходяться на етапі активного розвитку, що обумовлює потребу у створенні сучасного веб-застосунку, здатного забезпечити зручний користувацький інтерфейс та швидкий процес бронювання автомобіля.

Метою даної випускної кваліфікаційної роботи є розробка веб-сервісу для оренди автомобілів, який забезпечить просту, та надійну взаємодію між користувачем і системою. Для досягнення цієї мети необхідно вирішити задачі, що охоплюють як технічні аспекти реалізації, так і питання зручності використання, ефективності та безпеки сервісу. Необхідно провести аналіз існуючих рішень у сфері онлайн-оренди автомобілів та визначити функціональні та нефункціональні вимоги до веб-сервісу.

Однією з перших компаній, що реалізували повноцінну онлайн-модель оренди автомобілів є Zipcar. Заснована в 2000 році у Кембриджі, штат Массачусетс, США, компанія стала відповіддю на потребу в більш гнучкому та доступному підході до користування автомобілем у міських умовах.

Веб-застосунок має складатися з декількох взаємопов'язаних програмних модулів, кожен з яких реалізує окремий функціонал для користувачів та адміністраторів платформи. [4]

У структурі веб-сервісу для оренди авто одним із ключових елементів є модуль користувача, який реалізує процеси реєстрації, авторизації та управління персональними даними користувача. Його призначення полягає у тому, щоб створити персоналізоване середовище для кожного клієнта, що користується веб-сервісом, забезпечивши при цьому захист даних та зручність у використанні. Функціонал реєстрації дозволяє новим користувачам створити обліковий запис, ввівши необхідні дані для ідентифікації клієнта. Після успішної реєстрації користувач отримує доступ до особистого кабінету, який включає в себе можливість перегляду та редагування особистої інформації. Для захисту персональних даних використовуються методи токенизації а також захисту від типових загроз, таких як SQL-ін'єкції та CSRF-атаки. [8], [10]

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

Модуль каталогу авто є важливою складовою веб-сервісу, саме через нього користувачі взаємодіють з автопарком компанії. Цей модуль забезпечує зручну і структуровану подачу інформації про всі доступні автомобілі. Кожна одиниця автопарку відображається у вигляді окремого інформаційного блоку, який містить фотографії та ключові параметри автомобіля

Через модуль бронювання реалізується основна функція веб-сервісу, оформлення замовлення на оренду транспортного засобу. Після вибору авто на сторінці каталогу, система автоматично перевіряє в базі даних, чи не є транспортний засіб вже заброньованим на ці дати. Така перевірка запобігає конфліктам при одночасному бронюванні. Даний модуль дозволяє користувачу дату та час початку і завершення оренди, на основі введених даних система автоматично розраховує вартість послуги.

Невід’ємною складовою веб-застосунку для оренди автомобілів є модуль адміністрування, він забезпечує ефективне управління всіма ключовими аспектами функціонування платформи. Його метою є надання адміністратора інструментів для контролю над вмістом системи. Однією з головних функцій є додавання та редагування автомобілів, які відображаються в каталозі для користувачів із вказанням всіх необхідних параметрів. Також важливим аспектом є управління користувачами та обробка заявок на бронювання.

У результаті розробки повинен бути створений повнофункціональний веб-застосунок, що об’єднує в собі кілька взаємопов’язаних модулів і забезпечує зручний та безпечний сервіс для оренди автомобілів. Сервіс орієнтований на користувачів, які шукають авто для тимчасового використання та на адміністратора платформи, що відповідає за контент та обслуговування.

1.2. Комплексний аналіз функціональних можливостей та недоліків існуючих веб-платформ

Щоб виявити основні функціональні можливості, типові недоліки, а також створити сучасний та зручний сервіс, орієнтований на широку аудиторію користувачів, потрібно провести аналіз існуючих рішень на ринку оренди авто.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

Getmancar – одна з провідних каршеринг-компаній в Україні, заснована у 2017 році Тарасом Гетманським. Головна сторінка <https://getmancar.com/uk> зображена на рис. 1.1. Станом на 2025 рік сервіс активно працює в Україні (Київ та Дніпро), Грузії (Тбілісі, Батумі) та Молдові (Кишинів).

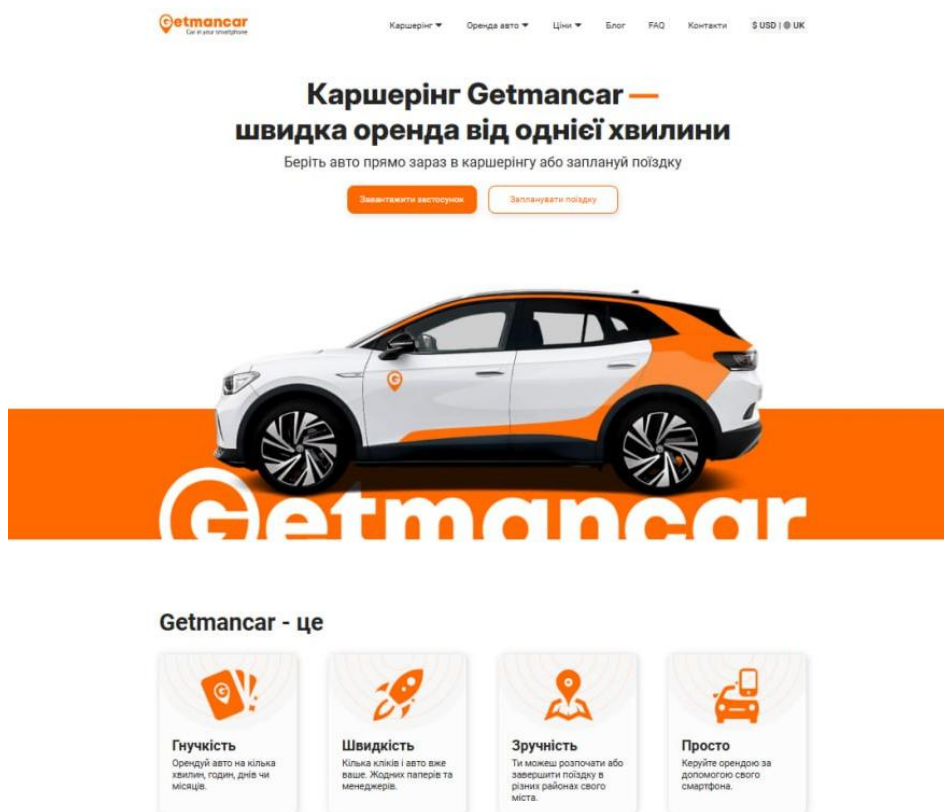


Рис. 1.1 – Головна сторінка веб-застосунку Getmancar

Основні переваги Getmancar:

Користувач може орендувати авто на будь-який термін, це може бути короткострокова поїздка тривалістю всього кілька хвилин так і довготривала оренда на декілька днів чи місяців. На сервісі представлено більше 500 автомобілів різних класів, від економ до бізнес-класу, також електромобілі та SUV. Клієнт може звернутися до служби підтримки, яка допомагає у вирішенні питань чи проблем під час оренди.

Недоліки:

Веб-сайт сервісу має досить обмежену функціональність, бронювання доступне лише після переходу на мобільний застосунок. Неможливо повноцінно управляти акаунтом, та редагувати профіль. Обмежене UX, загалом інтерфейс сайту орієнтований переважно на інформування, а не взаємодію.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

NarsCars – компанія з оренди авто в Україні, з власним автопарком, пропонує широкий вибір транспортних засобів. На рис. 1.2 та 1.3 зображено інтерфейс сайту <https://narscars.com.ua/ua>, який має сучасний адаптивний дизайн і дозволяє комфортно користуватись ним як на комп'ютері так і на мобільних пристроях. Також на сайті реалізована система реєстрації та особистий кабінет, який дозволяє клієнтам переглядати історію бронювань. Компанія працює з 2012 року, обслуговує приватних клієнтів, а також бізнес в найбільших містах країни.

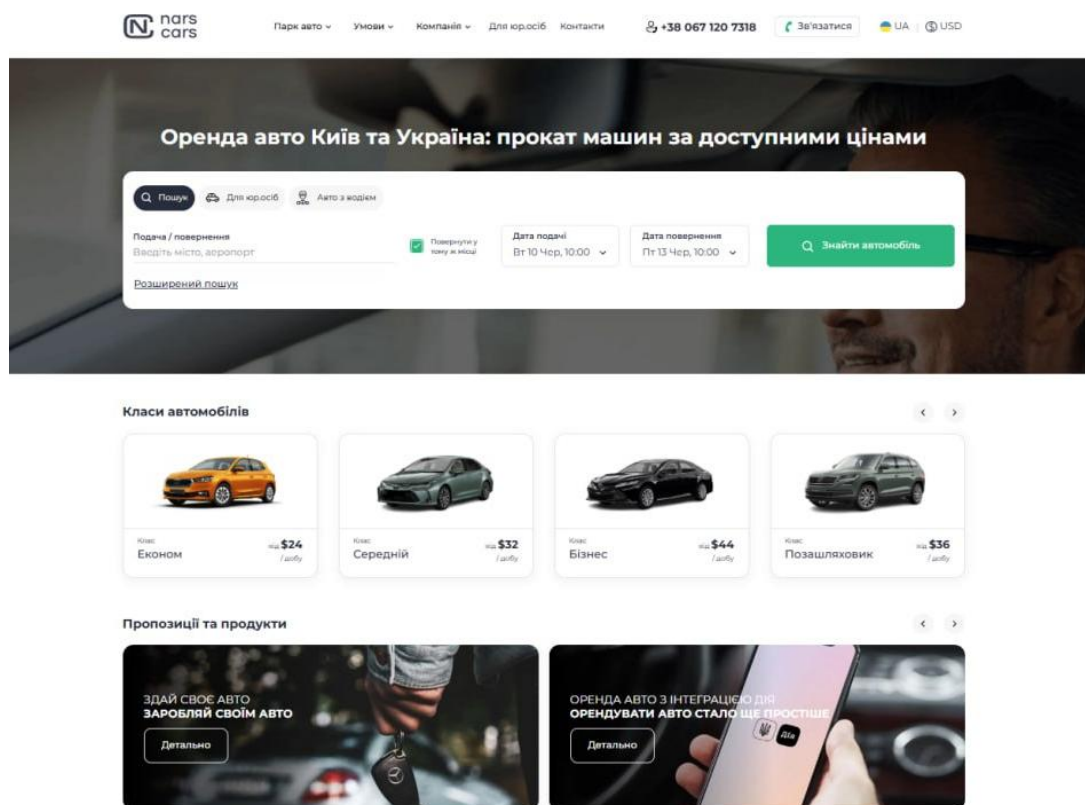


Рис. 1.2 – Головна сторінка сайту NarsCars

Основні переваги NarsCars:

Однією з основних переваг сервісу NarsCars є зручний та інтуїтивно зрозумілий інтерфейс для пошуку автомобіля. Користувачам надається можливість швидко відфільтрувати доступні авто за такими параметрами, як марка, модель, тип кузова, клас автомобіля. Також при оформленні бронювання можна замовити додаткове обладнання. На сайті представлені різні умови оренди;

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

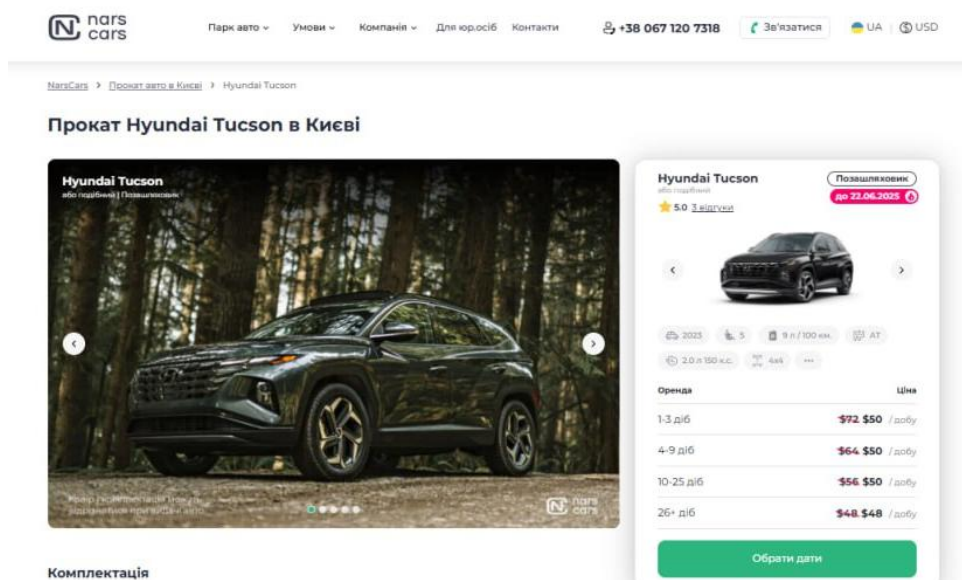


Рис. 1.3 – Сторінка бронювання авто на платформі NarsCars

Недоліки:

Додаткові платежі не завжди очевидні одразу. Немає повної автоматизації бронювання, для підтвердження замовлення часто необхідно телефонувати оператору. Інформація про наявність не завжди актуальна.

Rental.ua – це агрегатор послуг з оренди авто в Україні. Він об'єднує пропозиції від різних компаній, і це дозволяє користувачам орендувати авто онлайн у таких містах: Київ, Львів, Одеса, Харків, Дніпро. На рис. 1.4 та 1.5 зображені основні сторінки сайту <https://rental.ua/ua>.

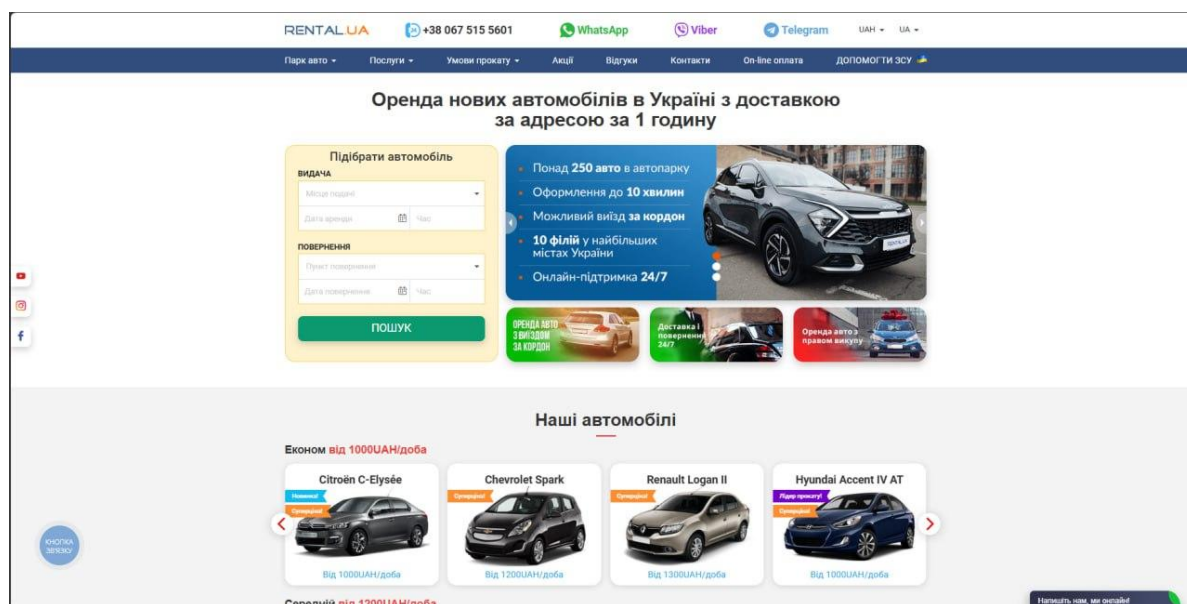


Рис. 1.4 – Головна сторінка веб-сервісу Rental.ua

					ПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15

Основні переваги Rental.ua:

Однією з головних особливостей є широкий вибір автомобілів, сервіс об'єднує пропозиції від великої кількості прокатних агентств по всій Україні, що дозволяє знайти оптимальний варіант як за ціною, так і за технічними характеристиками. Користувачі можуть повністю забронювати авто через сайт, без потреби у додаткових дзвінках. Кожне авто має окрему картку з ключовими характеристиками.

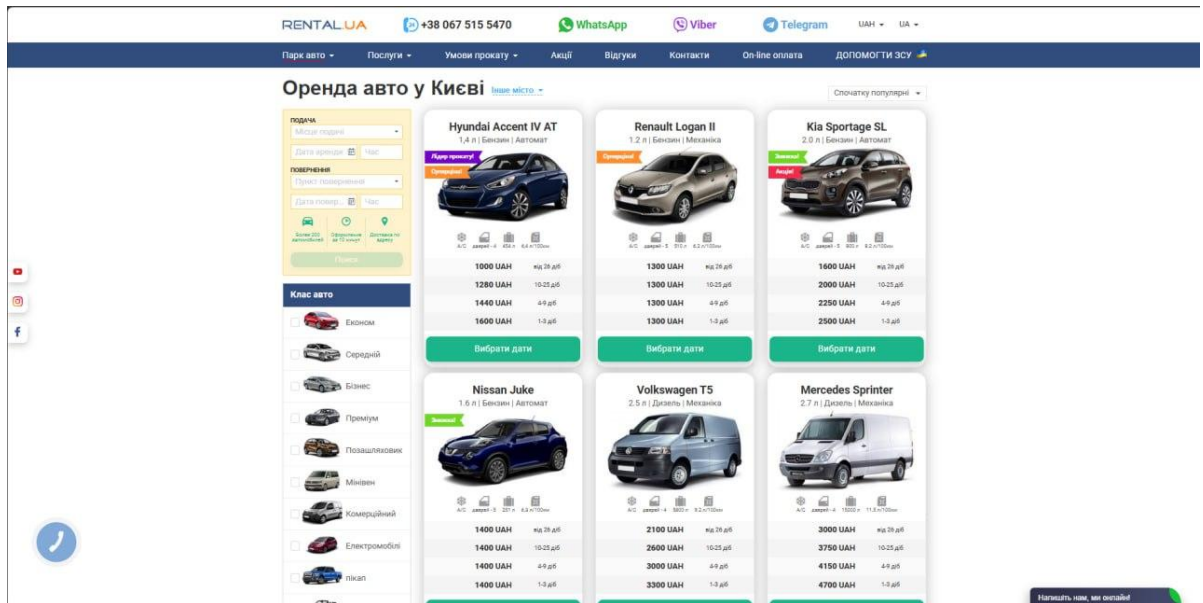


Рис. 1.5 – Сторінка бронювання авто на сайті Rental.ua

Недоліки:

Інтерфейс платформи виглядає застарілим та перенавантаженим інформацією. Відсутність особистого кабінету або профілю користувача. Іноді після завершення процедури бронювання з'ясовується, що обране авто фактично недоступне.

TopRent.ua – сервіс прокату автомобілів, працює переважно в Києві. На рис. 1.6 та 1.7 зображені головна сторінка та сторінка бронювання авто. Компанія пропонує широкий вибір авто різних класів, від економ до преміум. Сайт <https://toprent.ua/uk> дозволяє ознайомитись з автопарком, умовами оренди та залишити заявку онлайн.

					ПІЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

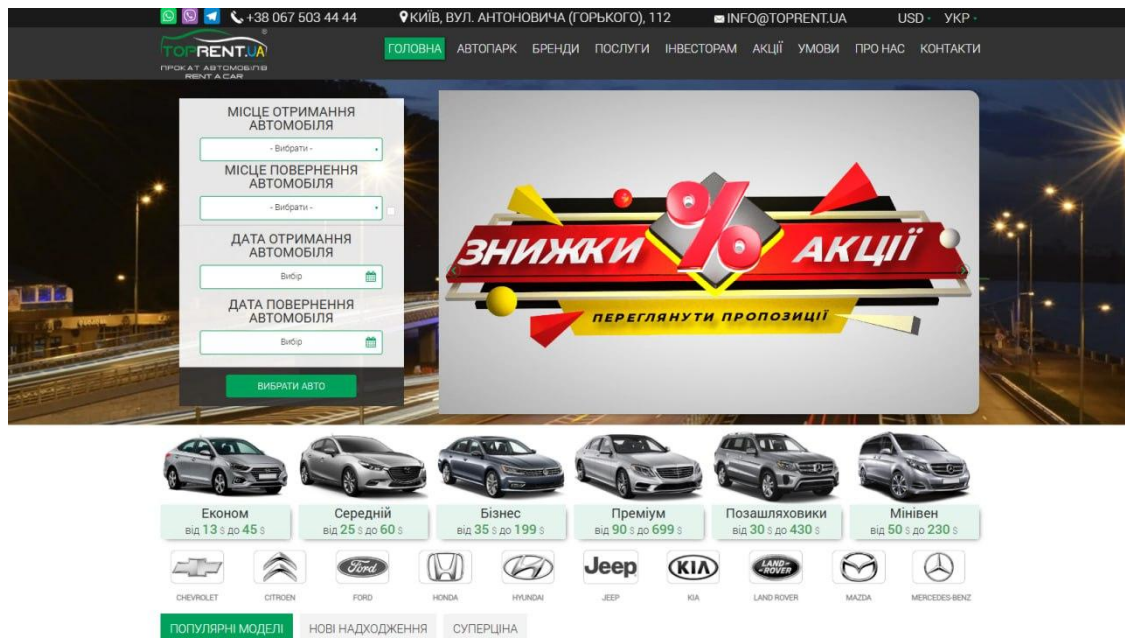


Рис. 1.6 – Головна сторінка сайту TopRent.ua

Основні переваги TopRent.ua:

Після успішного бронювання користувач автоматично отримує підтвердження на електронну пошту, це додає впевненості у статусі замовлення. Детальна інформація про авто: витрата палива, об'єм багажника. На сайті є відображення локації автопарку.

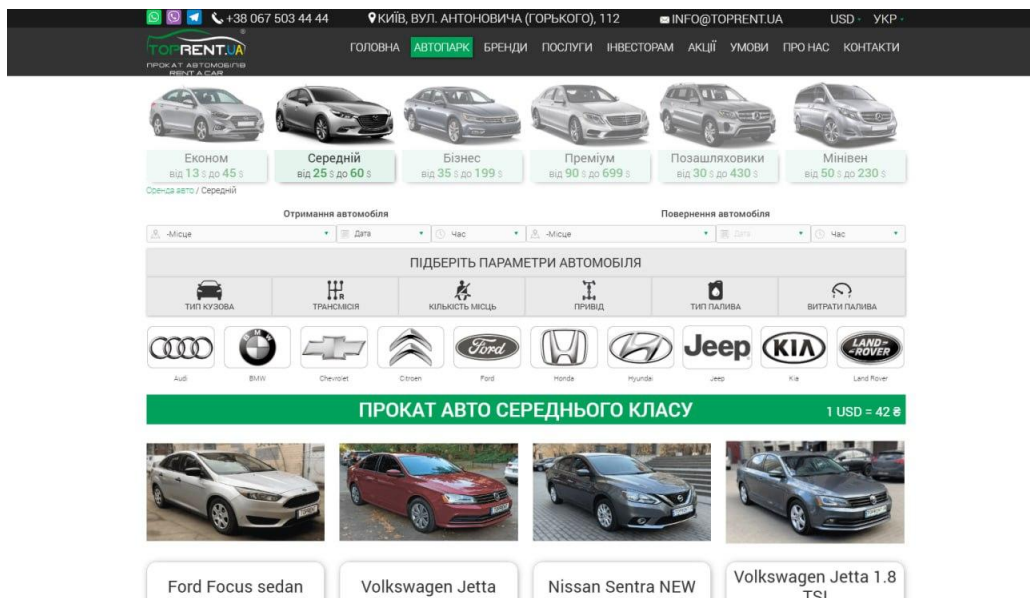


Рис. 1.7 – Сторінка бронювання авто на сайті TopRent.ua

Недоліки:

Остаточна вартість оренди уточняється по телефону з менеджером. Складна навігація по сайту, також сайт повільно завантажується.

					ПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

Sixt – один найбільших операторів оренди авто у світі, працює більш ніж у 100 країнах, включаючи Україну. Головна сторінка сайту <https://www.sixt.com/> зображена на рис. 1.8. Компанія має широкий автопарк, від бюджетних моделей до люксових автомобілів, має якісне обслуговування та цифровізацію процесу бронювання.

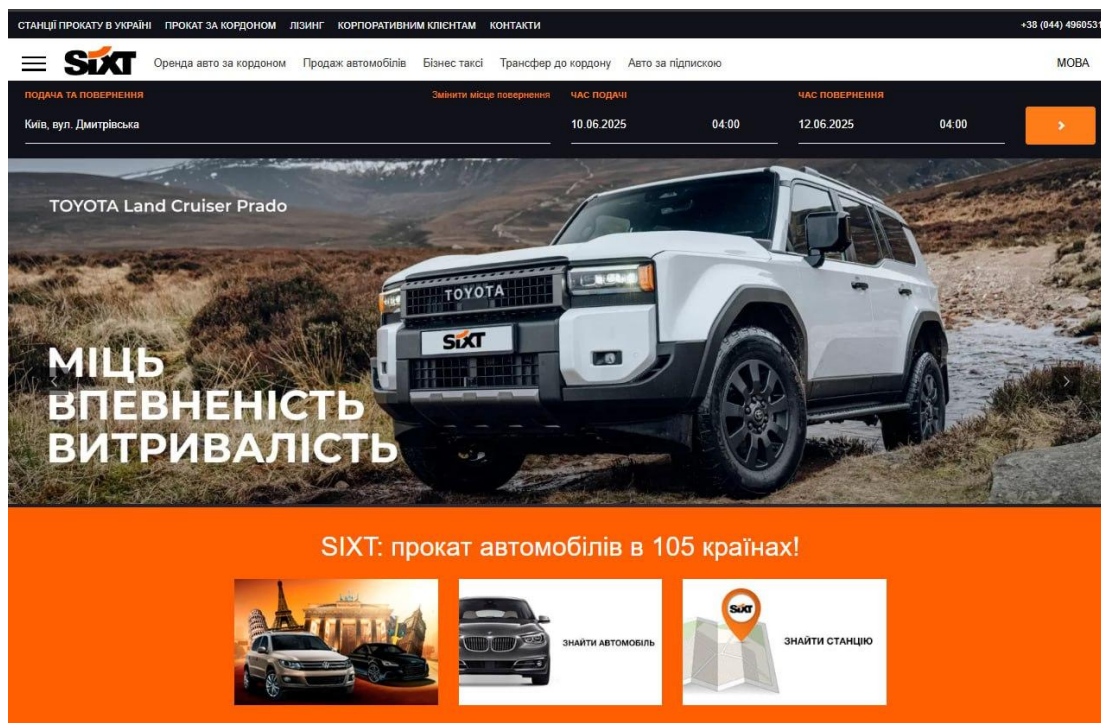


Рис. 1.8 – Головна сторінка веб-сервісу Sixt

Основні переваги Sixt:

Широкий вибір класів авто, від бюджетних моделей до преміум сегменту. Зрозумілий Інтерфейс платформи з чітким поділом на категорії. Окрему увагу приділено корпоративному сегменту, компанії мають можливість реєструвати корпоративні акаунти. З недоліків можна виділити великий обсяг умов оренди, також для замовлення обов'язково потрібна кредитна картка. Складна структура процесу бронювання, для нових клієнтів це може спричинити труднощі у користуванні сервісом.

На основі виявлених сильних і слабких сторін кожного з розглянутих сервісів, була сформована порівняльна таблиця 1.1, яка дозволить узагальнити їх функціональні можливості та зручність використання. Також в таблицю додано власну розробку, сайт RoyalRent.ua.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.1

Порівняльний аналіз веб-сервісів для оренди авто

Характеристика	Getmancar	NarsCars	Rental.ua	TopRent	Sixt	RoyalRent
Зрозумілий інтерфейс						
Категорії та фільтри авто						
Актуальність інформації про наявність						
Відкритість цінової політики						
Бронювання онлайн						

На основі аналізу існуючих веб-сервісів з оренди авто можна зробити висновок, що більшість із цих застосунків мають схожий функціонал, але відрізняються по якості реалізації, зручності в користуванні, актуальній інформації та прозорістю цін. На таких платформах користувачі можуть стикатися з труднощами при спробі забронювати автомобіль онлайн. Деякі сайти мають застарілий або перевантажений інтерфейс і це ускладнює навігацію.

З огляду на ці недоліки, у межах проекту було прийнято рішення розробити сучасний веб-сервіс, орієнтований на зручність та прозорість. Особлива увага буде приділена створенню інтуїтивного інтерфейсу з системою реєстрації та авторизації, пошуку автомобілів та можливістю оформляти бронювання онлайн. Також буде впроваджено профіль користувача з можливістю редагування особистої інформації.

1.3 Визначення концептуальних вимог до архітектури інформаційної системи

Архітектура інформаційної системи виконує ключову роль у забезпеченні її функціональності, надійності, масштабованості та зручності подальшого супроводу. Перед тим як перейти до проектування структури веб-застосунку, необхідно сформулювати вимоги, які визначають основні характеристики архітектури відповідно до потреб майбутніх користувачів та технічного середовища. Веб-сервіс для оренди автомобілів повинен забезпечувати безперебійну взаємодію між клієнтською частиною, сервером та базою даних. Крім того, система повинна бути готова до змін у майбутньому, включаючи масштабування функціональності, збільшення кількості користувачів, також інтеграції з зовнішніми сервісами. [7]

Концептуальні вимоги, які стануть основою для формування архітектури: функціональна відповідність, масштабованість, надійність і відмовостійкість, безпека, сучасний динамічний інтерфейс, продуктивність, сумісність та інтеграція.

Функціональна відповідність визначає здатність системи повною мірою реалізувати запланований функціонал. При розробці веб-сервісу для оренди автомобілів функціональна відповідність передбачає підтримку таких ключових можливостей, як реєстрація, та авторизація користувачів, перегляд транспортних засобів, оформлення бронювання, а також адміністрування автопарку. Архітектура повинна бути спроектована таким чином, щоб забезпечити розподіл навантаження між клієнтською та серверними частинами, що дозволить ефективно реалізовувати зазначені функції як з точки зору продуктивності, так і зручності користування. Необхідно побудувати збалансовану систему, де кожен компонент відповідає за виконання чітко визначених завдань, і в сукупності це забезпечить функціональну повноту та відповідність архітектурного рішення поставленим цілям. [7]

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

Масштабованість є властивістю архітектури інформаційної системи, яка визначає її здатність ефективно функціонувати при зростанні навантаження. Важливо враховувати можливе збільшення кількості користувачів, запитів до сервера, взаємодій з базою даних та обсягів збереженої інформації. Архітектура повинна передбачати гнучке розширення окремих її компонентів без необхідності повного перепроєктування. Це дозволить адаптувати систему до зростання бізнесу, регіонального розширення та інтеграції з додатковими платформами. Масштабована архітектура гарантує довготривалу ефективність функціонування системи в умовах зростаючого навантаження. [7]

Однією з важливих вимог до веб-сервісу для оренди авто, де збереження даних та забезпечення постійного доступу має важливе значення, є надійність та відмовостійкість. Архітектура повинна бути здатною витримувати різні збої, такі як втрата інтернет з'єднання або апаратні відмови, при цьому без порушення цілісності даних та доступності сервісу для користувача. Це досягається шляхом впровадження механізмів резервного копіювання, реплікації бази даних та автоматизованих стратегій відновлення після збоїв. Надійна архітектура також передбачає розмежування відповідальності між компонентами системи, щоб локалізувати проблеми без зупинки всього функціоналу.

Безпека є критично важливою складовою архітектури веб-сервісу для оренди автомобілів, де здійснюється обробка персональних даних користувачів та платіжної інформації. Архітектурні рішення повинні забезпечувати захист даних на всіх етапах обробки, від введення до зберігання й передачі. Необхідно реалізувати надійні механізми автентифікації та авторизації користувачів, що гарантує контроль доступу до функціоналу системи залежно від ролі та прав користувача. Важливим аспектом є впровадження шифрування даних під час зберігання їх в базі даних. При проектуванні системи необхідно дотримуватися сучасних стандартів безпеки, що дозволяють враховувати такі потенційні загрози, як SQL-ін'єкції, міжсайтове скриптування XSS та підроблені міжсайтові запити CSRF. Для запобігання цим загрозам потрібно

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

використовувати перевірені фреймворки, які мають вбудовані засоби захисту. [8], [9]

Важливою складовою архітектури веб-сервісу для оренди авто є сучасний динамічний інтерфейс, що реалізується за допомогою Single Page Application (SPA), це принцип, який виконує миттєве оновлення контенту без необхідності повного перезавантаження сторінки. Впровадження сучасного динамічного інтерфейсу дасть можливість легко інтегрувати додаткові функціональні модулі та розширення без значних змін у загальній структурі. Також це сприяє адаптивності інтерфейсу під різні типи пристроїв та зробить веб-сервіс доступнішим та зручнішим для широкого кола користувачів. [11]

Продуктивність є важливим аспектом в архітектурі інформаційної системи, він відповідає за здатність оперативно обробляти запити користувачів і забезпечує стабільну роботу в умовах підвищеного навантаження. Висока продуктивність є необхідною для забезпечення швидкого доступу до бази даних транспортних засобів, обробки замовлень та динамічного відображення змін. Досягнення високої продуктивності забезпечується поєднанням кількох підходів. Насамперед це оптимізація запитів до бази даних, яка зменшує кількість операцій читання та запису, використовуючи індекси, денормалізацію та механізми кешування. Ефективне кешування дозволяє зменшити навантаження на сервер та зменшуючи кількість повторних запитів прискорити вивід даних.

Важливими аспектами у проектуванні веб-сервісу для оренди авто є сумісність та інтеграція, вони визначають здатність системи ефективно взаємодіяти з іншими програмними рішеннями, платформами та технологіями. Висока сумісність системи досягається завдяки використанню відкритих стандартів обміну даними, зокрема форматів JSON та XML, а також підтримці RESTful API, що дозволяє взаємодіяти з системою через мережеві протоколи. Завдяки цьому веб-сервіс може адаптуватися до змін бізнес вимог і легко масштабуватися у майбутньому.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до першого розділу

У першому розділі було проведено комплексний аналіз розвитку цифрових веб-сервісів для оренди авто, також проведено дослідження функціональних можливостей та недоліків існуючих веб-сервісів. Було встановлено, що більшість сучасних сервісів забезпечують користувача базовими можливостями, такими як реєстрація користувачів, перегляд доступних транспортних засобів та оформлення замовлення. Були також виявлені обмеження, такі як відсутність можливості повноцінної онлайн-оренди автомобіля, обмежений функціонал профілю користувача, а також незручний інтерфейс, що ускладнює взаємодію користувача із системою.

В результаті дослідження виявлено ключові вимоги до архітектури майбутнього веб-сервісу. Було обговорено доцільність використання клієнт-серверної архітектури для забезпечення швидкого та надійного обміну даними між користувачами та сервером. Серед найголовніших вимог також є необхідність реалізувати такі функції, як реєстрація та аутентифікація користувачів, перегляд доступних автомобілів, можливість бронювання та адміністрування системи для управління контентом і користувачами. У першому розділі було складено основу для створення веб-сервісу з урахуванням потреб користувачів і сучасних вимог ринку автопрокату. Результати аналітичних досліджень першого розділу створили підґрунтя для наступного етапу розробки, проектування архітектури інформаційної системи, вибору технологій і практичної реалізації програмного продукту.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА РЕАЛІЗАЦІЯ ЇЇ ПРОГРАМНИХ КОМПОНЕНТІВ

2.1 Вибір архітектурно-технологічної платформи та інструментальних засобів

Етап вибору архітектури веб-застосунку є особливо важливим. Необхідно визначити основні компоненти програмного забезпечення, їхню структуру та способи взаємодії між ними. Вибір архітектури впливає на подальшу підтримку системи, її масштабованість, продуктивність та стабільність у роботі. У розробці веб-сервісу потрібно забезпечити взаємодію між користувачем, серверною частиною та базою даних. [33]

На сьогодні існує декілька основних архітектурних підходів, які можуть бути використані при створенні веб-сайтів. Найбільш поширеними серед них є монолітна архітектура, клієнт-серверна модель, REST-архітектура, архітектура мікро-сервісів та архітектура односторінкових застосунків.

Монолітна архітектура – це реалізація всієї бізнес-логіки, інтерфейсу користувача та доступу до даних в межах одного застосунку. Такий підхід є простим в реалізації та зручним для невеликих проєктів. Однак при масштабуванні або зміні окремих компонентів виникають проблеми, оскільки заміна одного модуля може вплинути на всю систему. [13]

Клієнт-серверна модель дозволяє чітко розділити відповідальність між серверною частиною, яка працює з бізнес-логікою, і клієнтською частиною, яка реалізовує інтерфейс взаємодії з користувачем. Така модель більш гнучка та контролює передачу даних. [1], [3]

REST-архітектура ґрунтується на принципах використання HTTP-протоколу та стандартних методів звернення до ресурсів. Вона є логічним продовженням клієнт-серверної моделі, забезпечує зручний спосіб обміну даними між frontend і backend, це дозволяє реалізувати гнучку взаємодію між частинами системи. [2]

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

Мікросервісна архітектура дозволяє розділити застосунок на окремі сервіси, кожен з них виконує власну функцію. Хоча цей підхід є сучасним і перспективним, він вимагає значних ресурсів для розробки, тестування і моніторингу. [15]

SPA (Single Page Application) архітектура односторінкових застосунків дозволяє реалізувати взаємодію користувача з додатком без повного перезавантаження сторінки. Завдяки сучасним JavaScript-фреймворків користувач отримує зручний та динамічний інтерфейс. [11]

Веб-сервіс для оренди авто буде побудований з використанням декількох архітектурних патернів: MVC (Model-View-Controller), REST та SPA (Single Page Application). Таке поєднання забезпечить гнучкість, зручність у розробці, масштабованість та високу продуктивність системи. Кожен із цих патернів виконує свою роль у структурі сайту та задовольняє функціональні потреби. [16]

Подання (View) відповідає за візуальне відображення даних, виконується фронтом у вигляді SPA. Контролери (Controller) обробляють запити користувача, взаємодіють з моделями, опрацьовують дані та передають результати у вигляді JSON-відповідей до фронту. Це забезпечить зручну маршрутизацію та централізовану логіку обробки запитів.

REST (Representational State Transfer) виступає в якості основного стилю архітектури для побудови інтерфейсу прикладного програмування (API). У рамках REST кожен ресурс, наприклад користувач або автомобіль, має унікальний URL-ідентифікатор, а взаємодія з цими ресурсами здійснюється через стандартні HTTP-методи: GET для отримання даних, POST для створення, PUT або PATCH для оновлення та DELETE для видалення. RESTful API реалізується за допомогою Laravel. [2], [19]

Патерн SPA дозволяє реалізувати односторінкову архітектуру інтерфейсу користувача. Основна HTML-сторінка завантажується один раз, а подальша взаємодія з додатком відбувається за рахунок динамічного оновлення контенту через JavaScript. Для реалізації SPA ми використовуємо React.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

Використання архітектури MVC, REST та SPA дозволить створити модульну та зручну для тестування систему з можливістю масштабування. MVC забезпечує внутрішню логічну організацію серверної частини. REST визначає правила взаємодії між частинами системи, у тому числі між фронтендом і бекендом. SPA реалізовує сучасний динамічний інтерфейс з мінімальними затримками та без повного перезавантаження сторінки. Така архітектура відповідає вимогам до високої доступності, масштабованості, гнучкості в розширенні функціоналу та зручності в обслуговуванні.

Користувач буде взаємодіяти лише з однією HTML-сторінкою, яка швидко оновлюється залежно від його дій. Наприклад, при бронюванні автомобіля користувач натискає відповідну кнопку, після чого відбувається відправка HTTP-запиту на API. Сервер обробляє запит, взаємодіє з базою даних та повертає результат, який React миттєво відображає на сторінці. Це зменшить навантаження на сервер. Реалізація API дозволяє в майбутньому інтегрувати систему з іншими платформами або мобільними застосунками без необхідності зміни бізнес-логіки. [14]

Серверна частина веб-застосунку буде розроблена як REST API за допомогою PHP-фреймворку Laravel, це забезпечить обробку запитів, автентифікацію, авторизацію та взаємодію з базою даних. Laravel – фреймворк для створення серверних вебдодатків мовою програмування PHP. Він базується на патерні Model-View-Controller (MVC) і включає в себе великий набір інструментів для побудови масштабованих, безпечних і підтримуваних API. Laravel дозволяє ефективно працювати з базами даних, завдяки вбудованому ORM Eloquent, а також має засоби маршрутизації, middleware, черги та систему автентифікації. Завдяки зручному синтаксису та великій екосистемі, Laravel дозволяє швидко реалізовувати складну бізнес-логіку. [26], [27]

Основними перевагами Laravel є читабельна та логічна структура коду також фреймворк містить багато модулів для роботи з сесіями, авторизацією, кешуванням, поштою та обробкою помилок. Laravel має вбудовані засоби захисту від поширених вразливостей, таких як SQL-ін'єкції, XSS та CSRF.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдяки системі міграцій є можливість легко керувати структурою бази даних і підтримувати її узгодженою в командній розробці. Завдяки вбудованим механізмам маршрутизації, підтримці ресурсних контролерів, автоматичній обробці запитів та відповіді у форматі JSON, Laravel дозволяє швидко реалізовувати RESTful API. [25]

Клієнтська частина веб-застосунку буде реалізована з використанням JavaScript-бібліотеки React, яка дозволяє створювати швидкий та зручний інтерфейс користувача. React – це бібліотека з відкритим кодом, призначена для побудови інтерфейсів користувача. React дозволяє створювати повторно використовувані компоненти, які реагують на зміни даних без перезавантаження сторінки. Також ця бібліотека дозволяє ефективно створювати SPA-додатки (Single Page Application) і забезпечує швидку взаємодію з інтерфейсом. [18]

Серед переваг бібліотеки React можна виділити компонентний підхід, увесь інтерфейс будується з невеликих незалежних компонентів, які легко тестувати та повторно використовувати. Також React дозволяє зручно керувати станом інтерфейсу за допомогою таких бібліотек як Redux або Context API. Завдяки розширенню синтаксису JSX, HTML та JavaScript можна поєднувати в одному файлі для кращої читабельності коду. React можна використовувати як окремо, так і в поєднанні з іншими бібліотеками або фреймворками, такими як Next.js. [17]

Для реалізації клієнтської та серверної частини було обрано Visual Studio Code – легке середовище розробки, яке надає розширені можливості для написання, налагодження та управління кодом. VS Code підтримує велику кількість розширень, включаючи засоби для роботи з PHP, JavaScript, HTML, CSS, базами даних та системами контролю версій. [30]

XAMPP – це універсальний пакет для локального розгортання вебсервера. Він включає в себе Apache, PHP та MySQL і дозволяє швидко розгорнути локальне середовище без необхідності ручної установки кожного компоненту.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

ХАМРР використовується для тестування серверної частини, а також взаємодії з базою даних під час розробки. [20], [24]

Для роботи з базою даних використовується phpMyAdmin – це веб-інтерфейс для адміністрування бази даних MySQL. Він дозволяє легко створювати таблиці, редагувати структуру бази, виконувати SQL-запити, керувати правами доступу та експортувати дані. [21]

MySQL – система керування базами даних, що підтримує реляційну модель. Вона забезпечує високу продуктивність при роботі з великими обсягами даних, підтримує транзакції, реплікацію, індекси, тригери та інші розвинені механізми. MySQL підходить для веб-проектів завдяки своїй стабільності, швидкодії та широкій сумісності з іншими інструментами. [22]

Для проведення навантажувального тестування було обрано програмне забезпечення Apache JMeter. Цей інструмент дає змогу моделювати реальні сценарії користувацької активності та оцінювати продуктивності. [32]

Для запуску та підтримки веб-сервісу для оренди автомобілів достатньо серверу з такими параметрами: двоядерний процесор, 4 ГБ оперативної пам'яті, не менше 20 ГБ вільного дискового простору та постійне інтернет-з'єднання. У разі збільшення навантаження передбачається можливість масштабування інфраструктури через хмарні сервіси або VPS.

2.2. Формалізація функціональних сценаріїв та побудова об'єктно-орієнтованої моделі

Веб-сервіс для оренди автомобілів надає інтерфейс, через який користувачі можуть переглядати доступні автомобілі, здійснювати бронювання, а адміністратори – керувати автопарком та користувачами. Система орієнтована на кінцевого користувача, що прагне швидко знайти відповідне авто для прокату, а також на адміністратора, який здійснює контроль за наявністю транспорту і обробкою замовлень.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

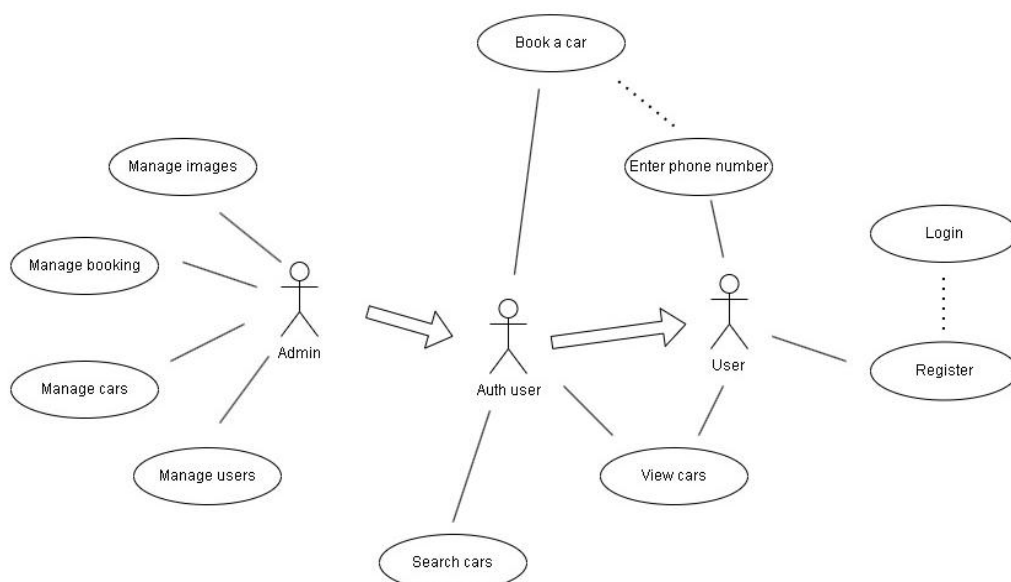


Рис. 2.1 – Діаграма варіантів використання

Діаграма варіантів використання, що зображена на рис. 2.1, відображає взаємодію користувачів із функціональністю веб-сервісу. Вона показує сценарії, у межах яких актори (користувачі) взаємодіють із системою.

Користувачі системи поділені на дві основні групи:

Користувачі (клієнти), переглядають каталог автомобілів, шукають авто за фільтрами, переглядають деталі (бренд, рік випуску, тип кузова, швидкість, колір, трансмісію тощо) та бронюють авто на вільні дати.

Адміністратори, мають доступ до панелі управління, де можуть створювати, редагувати та видаляти записи про автомобілі, бренди, категорії, користувачів і переглядати список бронювання. Опис діаграми варіантів використання наведено нижче в таблицях.

Таблиця 2.1

Прецедент «Вхід/Реєстрація»

Назва варіанту використання	Вхід/Реєстрація
Основні актори	Користувач, Адміністратор
Короткий опис	Користувачі можуть зареєструватися в системі для здійснення бронювання автомобілів, а адміністратори – для керування контентом та користувачами.

Таблиця 2.1 описує процес автентифікації користувачів у системі веб-сервісу. Цей функціонал є базовим і необхідним для розмежування доступу до функціональних можливостей застосунку. У системі передбачено два основних типи користувачів: клієнти, які здійснюють бронювання авто, та адміністратори, що керують контентом, бронюваннями та обліковими записами користувачів.

Для адміністратора передбачено окрему сторінку входу в систему, яка надає доступ до адмін-панелі. Це забезпечує інформаційну безпеку та розмежовує права доступу до даних і функцій веб-застосунку.

Таблиця 2.2

Прецедент «Пошук авто»

Назва варіанту використання	Пошук авто
Основні актори	Користувач
Короткий опис	Користувач має змогу здійснювати пошук автомобіля за назвою, брендом або категорією для швидкого доступу.

Варіант використання «Пошук авто», наведений в таблиці 2.2 описує функцію, яка дозволяє користувачам ефективно знаходити автомобілі в межах веб-сервісу оренди. Основна мета цього процесу забезпечити швидкий доступ до транспортного засобу на основі заданих критеріїв, таких як назва моделі, бренд або категорія (тип кузова, клас).

Таблиця 2.3

Прецедент «Перегляд автомобілів»

Назва варіанту використання	Перегляд автомобілів
Основні актори	Користувач, Користувач гість
Короткий опис	Усі користувачі можуть переглядати список доступних автомобілів з детальною інформацією про них.

Таблиця 2.3 описує базову функціональність веб-застосунку, що забезпечує користувачам доступ до перегляду автомобілів, доступних для оренди. Цей процес не потребує автентифікації та доступний для всіх відвідувачів платформи.

Таблиця 2.4

Прецедент «Бронювання автомобіля»

Назва варіанту використання	Бронювання автомобіля
Основні актори	Користувач
Короткий опис	Зареєстрований користувач може вибрати автомобіль із каталогу, переглянути його характеристики, обрати дату та здійснити бронювання.

У таблиці 2.4 описана ключова бізнес-функція веб-сервісу – оформлення бронювання транспортного засобу. Цей процес доступний після проходження автентифікації, що забезпечує персоніфіковану взаємодію та зв'язок даних про бронювання з конкретним обліковим записом.

Користувач має змогу вибрати автомобіль із каталогу, ознайомитися з його характеристиками, ціновою політикою, умовами та наявністю. Після цього система дозволяє вибрати бажану дату початку та завершення оренди. Перед підтвердженням запиту система перевіряє доступність автомобіля на вибраний період, звертаючись до бази вже існуючих бронювань.

Таблиця 2.5

Прецедент «Редагування автомобілів»

Назва варіанту використання	Редагування автомобілів
Основні актори	Адміністратор
Короткий опис	Адміністратор має змогу додавати нові автомобілі, також редагувати або видаляти наявні. Для кожного автомобіля заповнюються технічні характеристики, категорія та бренд.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

Варіант використання «Редагування автомобілів» у таблиці 2.5 описує функціональність, доступну лише для користувачів з роллю адміністратора, яка забезпечує повний цикл управління контентом автопарку в системі. Цей варіант включає можливість додавання нових автомобілів, редагування існуючих записів, а також видалення непотрібних або неактуальних одиниць.

Під час додавання або редагування автомобіля адміністратор заповнює розширену форму. Усі зміни передаються через запити до бекенд-частини системи, де здійснюється відповідна валідація та збереження в базі даних.

Таблиця 2.6

Прецедент «Керування користувачами»

Назва варіанту використання	Керування користувачами
Основні актори	Адміністратор
Короткий опис	Адміністратор може переглядати список користувачів, редагувати їхні дані або видаляти облікові записи.

У таблиці 2.6 описаний варіант використання «Керування користувачами», він передбачає надання адміністратору інструментів для контролю, моніторингу та адміністрування облікових записів у системі. Ця функціональність є важливою частиною адміністративного інтерфейсу та виконує роль засобу підтримки порядку й безпеки на платформі.

Таблиця 2.7

Прецедент «Перегляд бронювань»

Назва варіанту використання	Перегляд бронювань
Основні актори	Адміністратор
Короткий опис	Адміністратор має можливість переглядати всі бронювання здійснені користувачами. Виводиться інформація про авто, дати бронювання та користувача.

Таблиця 2.7 описує варіант використання «Перегляд бронювань», що реалізує адміністративну функцію контролю за активністю користувачів у

частині здійснених бронювань. Цей функціонал надає адміністратору доступ до всіх записів про бронювання автомобілів, що зберігаються в системі, з метою моніторингу, аналітики та оперативного втручання у разі необхідності.

Ціль розробки веб-сервісу для оренди авто полягає у створенні зручного онлайн-середовища, яке дозволяє користувачам здійснювати пошук та бронювати автомобілі на визначений період. Така система забезпечує взаємодію між власниками автопарку та клієнтами.

Основна мета сервісу полягає у наданні користувачу можливості швидко знайти авто відповідно до своїх потреб, за категорією, брендом, типом кузова, та оформити бронювання без необхідності звернення до офлайн-агентств. Платформа також надає адміністраторам інструменти для управління автопарком, контролю бронювань та оновлення інформації про авто.

Вимоги користувачів:

Внутрішній користувач – зареєстрований клієнт:

Після проходження процедури реєстрації, клієнт отримує доступ до персоналізованого середовища, що дозволяє йому здійснити авторизацію ввівши електронну пошту та пароль. Увійшовши до облікового запису, користувач переходить у свій профіль де може редагувати ім'я, номер телефону, електронну пошту та скинути пароль. Користувач може переглядати перелік автомобілів за категоріями, брендами, типами кузова та класом. Крім ознайомлення з характеристиками автомобіля зареєстрований клієнт має змогу здійснити бронювання обраного автомобіля на визначені дати, використовуючи форму, що автоматично розраховує вартість послуги. Після успішного бронювання користувача переадресує на сторінку оплати.

Внутрішній користувач – гість:

Користувач отримує доступ до базового функціоналу платформи без необхідності попередньої авторизації або створення облікового запису. Гість має змогу переглядати повний перелік автомобілів, які знаходяться в базі даних, також здійснювати пошук за назвою авто. Також гість може ознайомитись з детальною інформацією про кожен автомобіль. У разі

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

заповнення форми бронювання, система відправляє користувача на сторінку реєстрації, де він може створити обліковий запис, ввівши електронну пошту, ім'я, номер телефону, пароль та підтвердження паролю.

Внутрішній користувач – адміністратор:

Доступ адміністратора обмежено через обов'язкову процедуру авторизації, яка забезпечується шляхом введення електронної пошти та пароля через окремий інтерфейс адміністратора, що гарантує безпечний доступ до панелі керування. Після входу в систему адміністратор отримує розширений функціонал, що дозволяє повноцінно керувати контентом сервісу. Він може додавати нові автомобілі до каталогу, вказуючи всю необхідну інформацію. Також адміністратор може редагувати наявні записи або повністю видаляти автомобілі з системи. Окрім додавання автомобілів адміністратор має змогу створювати нові категорії та бренди з можливістю редагування. Важливим елементом є перегляд та обробка заявок на бронювання, адміністратор має доступ до переліку всіх активних замовлень. Також є можливість переглядати та редагувати профілі користувачів для змоги вручну коригувати помилки у даних клієнтів та виділення акаунтів у разі виявлення порушень або виявлення адміністративних запитів.

Функціональні вимоги:

Система повинна реалізовувати механізм реєстрації, який ґрунтується на введенні електронної пошти, імені, номеру телефону, паролю та підтвердження паролю, також авторизацію з введенням електронної пошти та паролю, цей механізм має охоплювати як клієнтів, так і користувачів з правами адміністратора, забезпечуючи захищений доступ до персоналізованих розділів сервісу. Має бути наявність функціоналу редагування профілю, яка дозволить користувачам самостійно оновлювати персональні дані: ім'я, електронну пошту, номер телефону та пароль.

Для авторизованих клієнтів надати можливість здійснення бронювання транспортного засобу з вибором конкретних дат. Система повинна забезпечити пошук автомобіля, сформувати замовлення та зберегти деталі бронювання в

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

базі даних. Для адміністратора передбачено розширений доступ до управління автопарком. Це включає створення нових записів про автомобілі, категорії та бренди, редагування існуючих даних а також повне видалення. Уся інформація про здійснені бронювання повинна зберігатися у системі. Запис включає такі параметри: дати бронювання, облікові дані користувача, ID автомобіля та розраховану вартість.

Нефункціональні вимоги:

Інтерфейс системи повинен бути побудований таким чином, щоб ознайомлення з ним для нових користувачів займало не більше 3-5 хвилин, для тих, хто має досвід взаємодії з подібними системами – не більше 2 хвилин. Це досягається завдяки інтуїтивно зрозумілому розміщенню елементів, простій навігації та використанню загальноприйнятих шаблонів взаємодії.

Час відгуку на користувацькі дії, такі як навігація між сторінками, відкриття детальної інформації про автомобіль, або заповнення форм, не повинен перевищувати 3 секунди. Запити до серверної частини мають буди оптимізованими кешуванням.

Інтерфейс користувача повинен відповідати вимогам сучасного UX-дизайну: мати чітку структуру, логічно згруповані блоки інформації, зручні підписані кнопки. Веб-сервіс має бути спроектований з урахуванням можливих збоїв у роботі серверного обладнання або комунікацій. Усі дані повинні зберігатися із застосуванням резервного копіювання. Також система повинна бути здатною обробляти запити щонайменше 50 одночасних користувачів без помітного зниження продуктивності. Передбачити механізми балансування навантаження, масштабованість на та оптимізація бази даних і запитів до неї.

Системні вимоги:

Вимоги до середовища виконання (клієнтська частина):

У мінімальній конфігурації платформа буде функціонувати на пристроях з оперативною пам'яттю не менше 2ГБ, процесором з тактовою частотою від 1.4GHz, операційною системою не нижче Windows Edge або macOS Catalina.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

У бажаній конфігурації рекомендується використовувати пристрої з оперативною пам'яттю від 4ГБ, чотирьохядерними процесорами з частотою від 2.0GHz, операційною системою Windows 11 або macOS Ventura.

Необхідною умовою є стабільне підключення до мережі інтернет. Оптимальна робота досягається у браузерях, що регулярно оновлюються та не перебувають під міжнародними санкціями.

Вимоги до серверного середовища:

Серверна частина веб-сервісу реалізована за допомогою Laravel (PHP) і повинна бути розгорнута на середовищі, що підтримує: PHP >= 8.1; MySQL або іншу сумісну реляційну базу даних; Composer для керування залежностями.

Аналіз функціональних вимог дозволив визначити сутності та зв'язки між ними, вони відображають основні процеси, що відбуваються в системі та необхідні для реалізації веб-сервісу оренди автомобілів. На рис. 2.2 та 2.3 представлено відповідні діаграми класів, які ілюструють основну об'єктно-орієнтовану структуру системи.

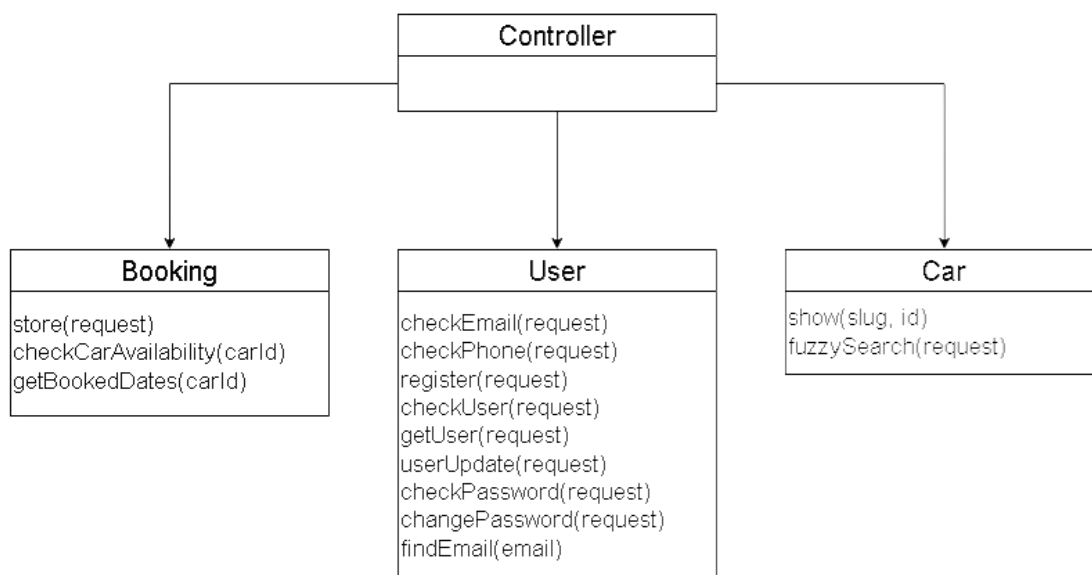


Рис. 2.2 – Діаграма класів, роль – User

Клас User є логічною складовою архітектури системи, що відповідає за обробку всіх запитів, пов'язаних із користувачем. Це включає як початкову реєстрацію, так і подальше управління акаунтом, перевірку унікальності даних та зміну пароля. У системі оренди авто даний клас реалізує функціонал, необхідний для взаємодії користувача з платформою.

Методи класу User: checkEmail(request) перевіряє, чи вказана електронна пошта вже зареєстрована в системі. Цей метод використовується при реєстрації або зміні email, щоб уникнути дублювання. Метод checkPhone(request) аналогічно до перевірки email, перевіряє унікальність номера телефону користувача.

Компонент register(request) реєструє нового користувача, використовуючи дані, передані через HTTP-запит. Створюється обліковий запис із базовими атрибутами (email, пароль, ім'я, номер телефону).

Метод checkUser(request) перевіряє наявність користувача в базі даних за переданими параметрами. Використовується для валідації під час входу або інших дій. Також getUser(request) повертає детальну інформацію про користувача. Метод updateUser(request) дозволяє оновити дані користувача: змінити ім'я, та контактні дані.

Метод checkPassword(request) перевіряє правильність поточного пароля, що важливо перед зміною пароля або виконанням входу в аккаунт, changePassword(request) змінює пароль користувача. Перед оновленням система перевіряє старий пароль та проводить валідацію нового, findEmail(email) дозволяє знайти користувача в базі за вказаною електронною адресою.

Клас Booking відповідає за логіку, пов'язану з бронюванням автомобілів. Це ключовий функціональний елемент системи, оскільки саме бронювання лежить в основі бізнес-моделі веб-сервісу оренди авто.

Методи класу Booking: store(request) який приймає запит з параметрами бронювання (користувач, автомобіль, дата початку, дата завершення) і створює нове бронювання в базі даних. Включає перевірку доступності авто, валідацію даних і збереження запису, checkCarAvailability(carId) перевіряє, чи доступний конкретний автомобіль (carId) у заданий проміжок часу та порівнює дати нового бронювання з уже існуючими бронюваннями, щоб уникнути конфлікту. Метод getBookedDates(carId) повертає список дат, коли автомобіль вже заброньований. Це дозволяє на фронтенді блокувати або підсвічувати зайняті дати в календарі, забезпечуючи кращий UX.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

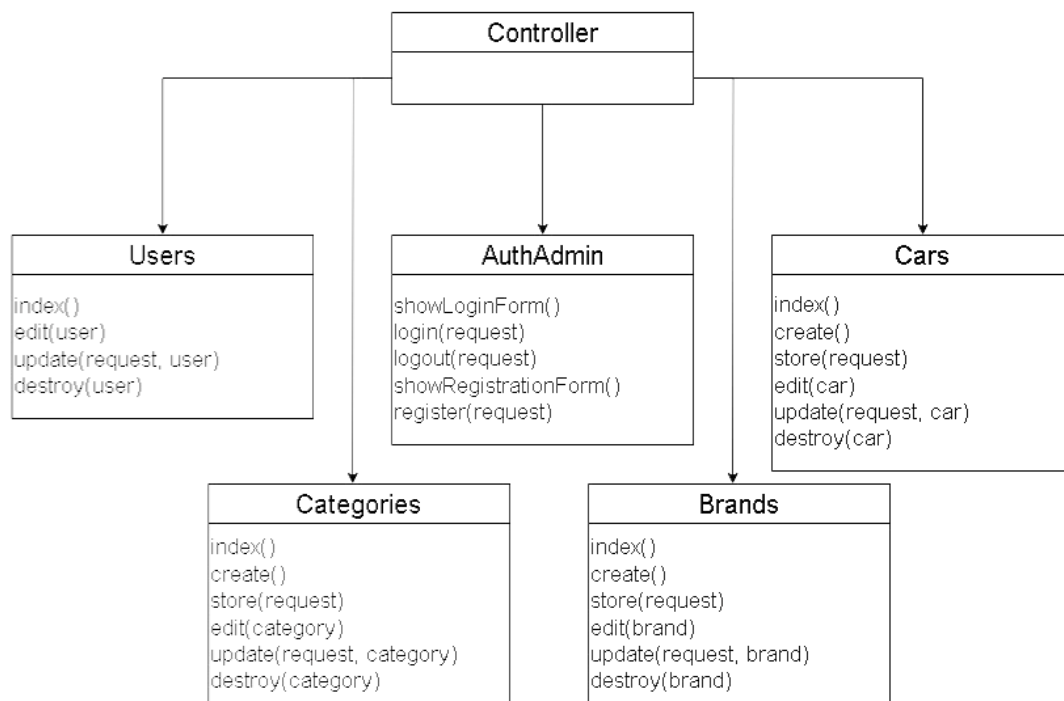


Рис. 2.3 – Діаграма класів (Admin)

Клас `AuthAdmin` відповідає за аутентифікацію та реєстрацію адміністратора. Це один із найважливіших методів в адміністраторській частині веб-застосунку, оскільки забезпечує контроль доступу до функцій керування вмістом сайту: авто, бренди, категорії, користувачі.

Методи класу `AuthAdmin`: `showLoginForm()` відповідає за відображення форми входу адміністратора. Це початкова точка для адміністратора, де він може ввести логін і пароль для доступу до закритих функцій панелі керування. Метод `login(request)` обробляє запит на вхід адміністратора в систему, виконує перевірку облікових даних (email і пароль), ініціює сесію та переадресовує на адмін-панель у разі успішної авторизації, у випадку помилки повертає відповідне повідомлення, `logout(request)` відповідає за вихід адміністратора із системи. Завершує сесію, очищує авторизаційні дані і перенаправляє назад на сторінку входу або головну. Метод `showRegistrationForm()` виводить форму для реєстрації нового адміністратора, `register(request)` приймає дані реєстраційної форми, створює нового адміністратора у базі даних. Перевіряє правильність введених даних, унікальність email-адреси, виконує хешування пароля і зберігає нового користувача з правами адміністратора.

Клас Users забезпечує керування обліковими записами користувачів у системі. Він дозволяє адміністратору переглядати, змінювати та видаляти дані користувачів, що використовують сервіс оренди автомобілів.

Методи класу Users: `index()` відповідає за виведення списку всіх користувачів. Адміністратор бачить загальний перелік зареєстрованих користувачів із базовими даними: ім'я, email, номер телефону. Метод `edit(user)` відображає форму редагування даних конкретного користувача. У формі адміністратор може змінити інформацію про користувача. Компонент `update(request, user)` Обробляє оновлення інформації про користувача, яку було внесено через форму. Здійснюється валідація введених даних, оновлення запису в базі даних і повідомлення про успішну зміну.

Метод `destroy(user)` відповідає за видалення користувача з системи. Може застосовуватись у випадку порушень правил сервісу, дублювання акаунтів або на запит користувача.

Клас Cars відповідає за управління всіма операціями, пов'язаними з автомобілями у системі, він надає повний набір CRUD операцій для управління інформацією про автомобілі у веб-сервісі.

Методи класу cars: `index()` виводить список усіх автомобілів, доступних у системі. Повертає JSON з переліком автомобілів для перегляду чи подальших дій. `create()` відображає форму для створення нового автомобіля. Ця форма дозволяє адміністратору ввести дані нового авто. Метод `store(request)` приймає дані з форми створення авто (об'єкт `request` містить введені користувачем дані), перевіряє їх, і зберігає новий автомобіль у базі даних, `edit(car)` відображає форму редагування для конкретного автомобіля, який передається у вигляді об'єкта `car`. Форма попередньо заповнюється існуючими даними цього авто для зручності редагування. Метод `update(request, car)` отримує оновлені дані автомобіля з `request` та оновлює інформацію про переданий у метод автомобіль `car` у базі даних, `destroy(car)` видаляє вказаний автомобіль з бази даних.

Такі методи для CRUD операцій також містять класи Categories Brands, які відповідають за інформацію про марку та категорію автомобіля

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

2.3. Розроблення клієнтської та серверної частин веб-сервісу на основі фреймворків React і Laravel

Опис алгоритму роботи сервісу. Веб-сервіс для оренди автомобілів має інтуїтивно зрозумілий інтерфейс, що дозволяє звичайним користувачам і адміністраторам взаємодіяти з системою починаючи з авторизації і завершуючи процесом оренди авто. Алгоритм роботи сервісу зображено на рис. 2.4.

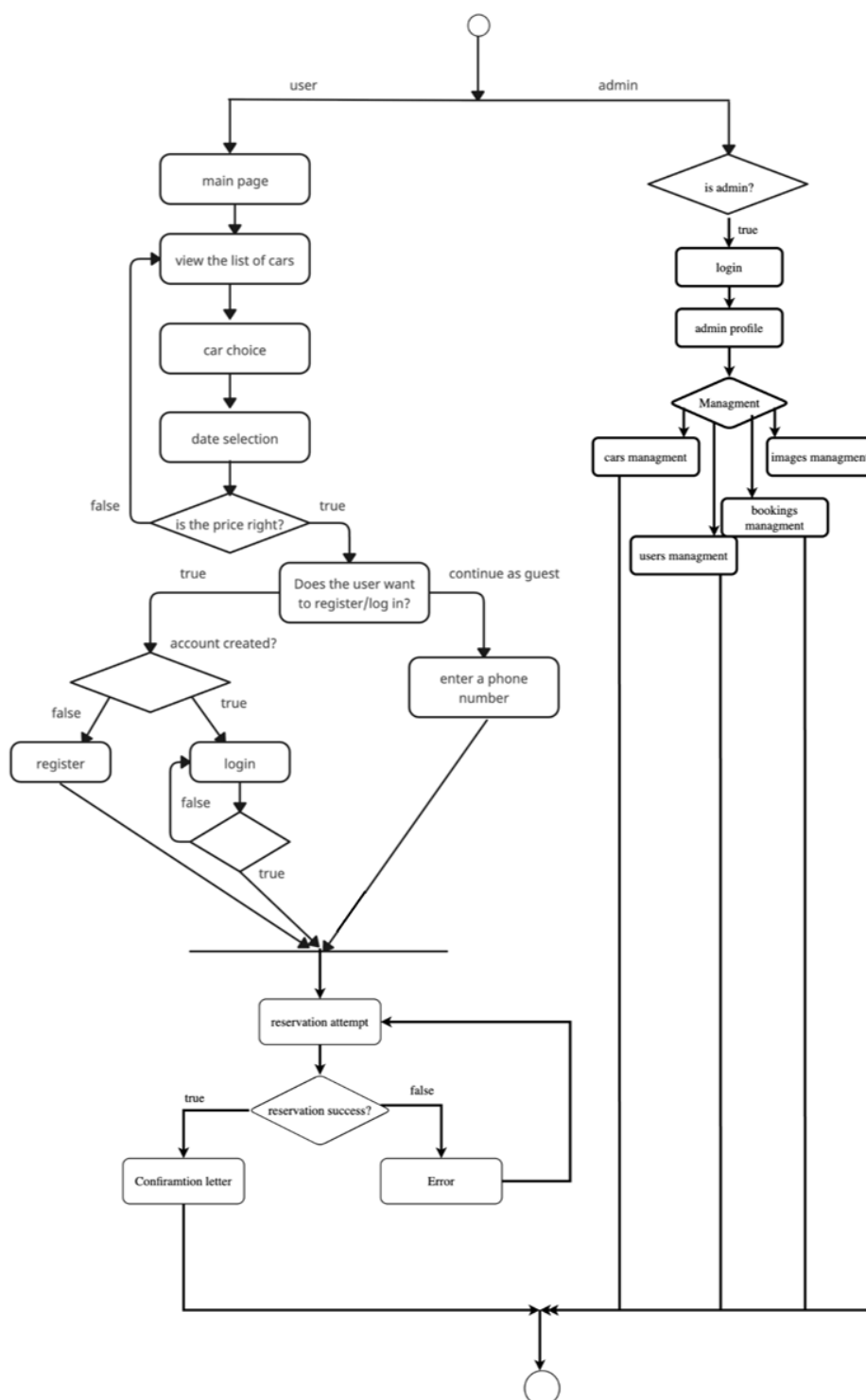


Рис. 2.4 – Діаграма алгоритму роботи сервісу

Змн.	Арк.	№ докум.	Підпис	Дата

Опис алгоритму роботи сервісу:

Якщо в систему намагається увійти користувач із правами адміністратора, на етапі авторизації система здійснює перевірку введених облікових даних, щоб переконатися, що цей користувач дійсно є адміністратором. Якщо інформація підтверджується, користувачу відкривається доступ до адміністративної панелі, яка містить розширений набір інструментів для управління сервісом.

Після входу адміністратор отримує можливість здійснювати повноцінне керування автопарком. Він може додавати нові автомобілі до бази даних, вносити зміни до вже наявних записів, а також видаляти ті транспортні засоби, які більше не надаються в оренду. Крім цього, адміністратор має змогу додавати зображення автомобілів, оновлювати їх або повністю видаляти за потреби. Також адміністратор може керувати обліковими записами користувачів, з редагувати їхні дані, видаляти профілі. Усі ці дії спрямовані на підтримку ефективної роботи веб-сервісу та забезпечення якісного обслуговування клієнтів. Важливою функцією адміністратора є також перегляд заявок на бронювання автомобілів, які надходять від зареєстрованих користувачів.

Якщо сервісом користується клієнт: користувач потрапляє на головну сторінку і має змогу переглянути список доступних автомобілів. Після вибору авто користувач переходить до сторінки автомобіля з усіма його характеристиками та формою бронювання з вибором початкової та кінцевої дати оренди. Система обчислює вартість оренди та на цьому етапі його запитують про бажання авторизуватися або продовжити як гість.

Якщо обрано реєстрацію: система перевіряє, чи створено обліковий запис. У разі відсутності акаунту пропонується реєстрація, після якої можливий вхід. Якщо ж обрано режим гостя, користувачу потрібно ввести номер телефону для продовження. Після успішної ідентифікації відбувається спроба бронювання авто. Якщо бронювання проходить успішно, користувач отримує

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

підтвердження. У разі помилки в процесі виводиться повідомлення про помилку, після чого користувач може повторити дію.

Реалізація маршрутизації API-запитів та логіки серверної частини з використанням фреймворку Laravel. У межах серверної частини веб-застосунку, було налаштовано маршрутизацію REST API, яка забезпечує взаємодію між клієнтським інтерфейсом та сервером. Для обробки HTTP-запитів використовуються контролери, які відповідають за отримання даних, створення бронювання, автентифікацію тощо. [6], [23]

У додатку (А) наведено фрагмент коду, який відповідає за конфігурацію маршрутів, що реалізують ключову частину API-функціоналу веб-сервісу з оренди автомобілів. Ці маршрути забезпечують взаємодію між клієнтською частиною застосунку та сервером.

Перший функціональний блок API відповідає за роботу з автомобілями. Реалізовано маршрут для отримання повного списку всіх доступних авто, яке здійснюється за унікальним ідентифікатором (id) та читабельним URL-ідентифікатором (slug), що дозволяє будувати зручні для користувача посилання. Також маршрут для пошуку автомобілів за ключовим словом, що забезпечує швидкий доступ до потрібної інформації з урахуванням введених параметрів.

Другий блок API стосується фільтрації автопарку. З цією метою реалізовані запити, які дозволяють отримати списки брендів, категорій авто та типів кузова. Ці дані використовуються для динамічного формування фільтрів на стороні клієнта, що значно полегшує навігацію по каталогу й підбір авто відповідно до уподобань користувача. [34]

Окрему функціональність реалізовано для реєстрації нових користувачів. Перед створенням облікового запису система виконує перевірку унікальності електронної пошти та номера телефону, щоб уникнути дублювання акаунтів. Також маршрут для пошуку користувача за e-mail. Після успішної авторизації користувач отримує доступ до особистого кабінету, де має змогу переглядати

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

та редагувати власні дані, включаючи зміну особистої інформації та пароля. Ця частина API захищена відповідними middleware для автентифікації, що гарантує, що лише авторизований користувач має доступ до власного профілю.

З метою ефективного розмежування ролей користувачів, забезпечення безпеки доступу та організованого керування ресурсами в межах веб-сервісу було реалізовано окрему групу маршрутів з префіксом /admin, яка відповідає за функціональність адміністративної панелі. Ця структура дозволяє централізовано управляти всіма адміністративними діями, забезпечивши чітке відокремлення логіки користувача та адміністратора.

Доступ до маршрутів групи /admin обмежено, вони активуються лише після проходження автентифікації користувача та обов'язкової перевірки його ролі. Для цього використовується механізм перевірки відповідності ролі is_admin, який гарантує, що жоден неавторизований або звичайний користувач не зможе потрапити до розділу керування системою.

Усі маршрути цієї групи захищені спеціальним middleware, який не лише перевіряє автентичність та права доступу, але й запобігає можливості повторного входу адміністратора, якщо він уже авторизований. Це дозволяє уникнути дублювання сесій та гарантує стабільну роботу адміністративного інтерфейсу.

Архітектура маршрутів, представлена у додатку (А), охоплює повний спектр адміністративних функцій. До неї входять маршрути для авторизації адміністратора (вхід і вихід із системи), доступ до головної сторінки панелі керування та персонального профілю. Окремим блоком реалізовані CRUD-операції для автомобілів, які дозволяють адміністратору додавати нові транспортні засоби, редагувати вже наявні, видаляти автомобілі з бази або переглядати повну інформацію про них.

Також передбачено маршрути для керування обліковими записами користувачів, адміністратор має змогу змінювати або видаляти профілі клієнтів у разі потреби. Окрему увагу приділено маршрутам, які дають змогу переглядати та обробляти заявки на оренду: адміністратор може бачити всі

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

запити, змінювати їхній статус, підтверджувати або відхиляти замовлення, а також переглядати детальну інформацію про кожну заявку.

Розробка клієнтської частини веб-сервісу з використанням бібліотеки React. Розглянемо основні дії користувача на сайті.

Код в додатку (Б) є частиною форми реєстрації, він виконує перевірку введених даних, потім відправляє їх на сервер.

Функція `handleSubmit` викликається при надсиланні форми, у ній перевіряються дані з форми `validateForm` та номер телефону через `handleValidatePhone`. Якщо дані правильні – відбувається POST-запит до API для створення нового користувача. Якщо API повертає помилку (наприклад, email уже зайнятий) – показується повідомлення про помилку.

Після успішної реєстрації користувача перенаправляють на сторінку входу `/login`. Код наведений у додатку(Б). Функція `validateForm` перевіряє поля `email` і `пароль` з об'єкта `formData` та створює порожній об'єкт `errors` для збереження помилок. Відправляється запит до API, щоб переконатися, що користувач із такими даними існує. Якщо поле порожнє або формат неправильний, або якщо поле паролю порожнє – записується помилка. Якщо відповідь від API – "success", у `localStorage` зберігається токен авторизації, щоб надалі ідентифікувати користувача.

Користувач також може змінити інформацію в профілі, наведений в додатку(Б) код обробляє подію натискання кнопки «Зберегти» в формі редагування. Поточні дані користувача до змін: `currEmail`, `currName`, `currPhone`. Якщо користувач нічого не змінив, тоді немає сенсу відправляти запит, виводиться помилка: "Нічого змінювати!". Якщо всі валідації пройдені, надсилається PUT-запит на `/api/user-update`. Він передає: оновлені дані користувача, токен з `localStorage`, який підтверджує, що користувач авторизований.

Показується повідомлення про успіх та через 1 секунду оновлюється сторінка, щоб побачити зміни. За успішне бронювання при натисканні кнопки

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

«Забронювати» відповідає наступна функція, наведена в додатку (Б). Функція `handleValidate` перевіряє чи заповнено всі обов'язкові поля. Якщо перевірка не пройдена – нічого не відправляється. Laravel API очікує формат дат уууу-мм-дд (2025-06-04), для цього використовуємо бібліотеку `date-fns`.

Створюється об'єкт з даними для бронювання: `start_date`, `end_date` – період оренди, `number` – номер телефона, `from_time`, `end_time` – час початку/кінця оренди, `car_id` – ID машини. Дані надсилаються в API через `axios.post` на маршрут `/api/bookings`. Laravel обробляє ці дані та створює відповідний запис у базі даних.

В ході розробки клієнтської частини веб-сервісу було створено діаграму компонентів оренди авто (рис. 2.5) на якій визначено основні компоненти. Вона ілюструє, як окремі елементи взаємодіють для забезпечення функціональності веб-застосунку.

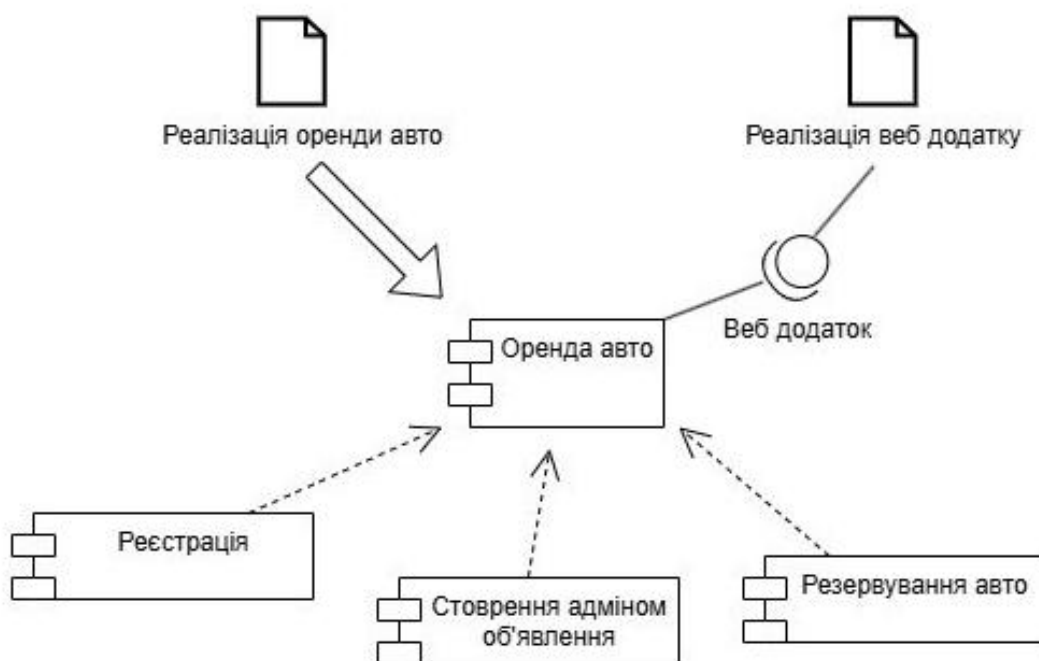


Рис. 2.5 – Діаграма компонентів оренди авто

2.4 Моделювання структури бази даних та організація взаємодії з API

Процес моделювання структури бази даних включає визначення таблиць, атрибутів, типів даних та побудову логічних зв'язків між об'єктами. Для збереження файлів та інформації у додатку використовується MySQL, це система управління базами даних, що базується на реляційній моделі. Для зручності адміністрування бази даних застосовується графічний веб- інтерфейс phpMyAdmin. На рис. 2.6 зображена діаграма бази даних та зв'язки між таблицями 2.8 – 2.20, описаними нижче, вони містять перелік полів, типи даних, вказівки для перевірки зовнішніх ключів, а також опис бізнес-логіки. [35], [12]

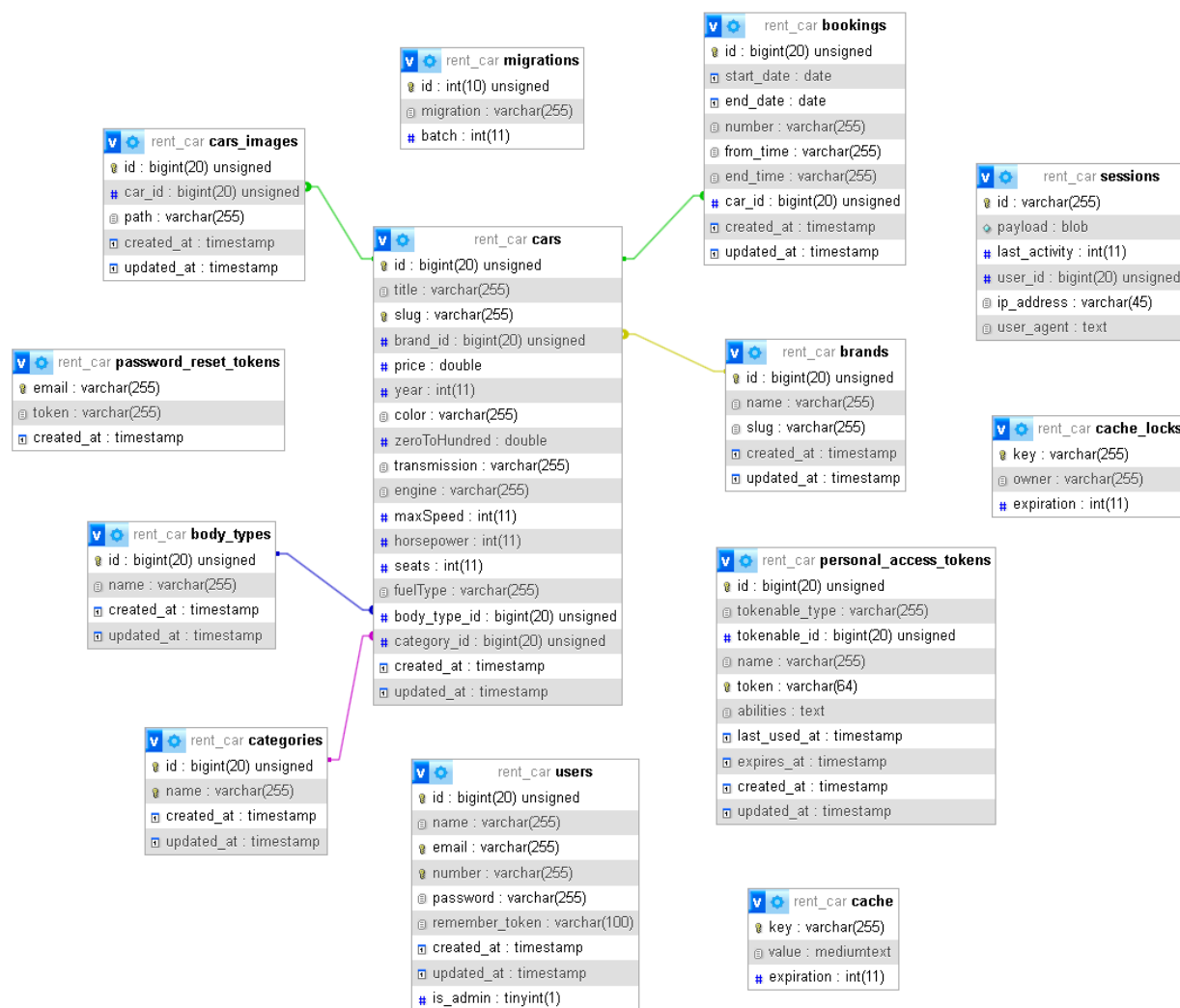


Рис. 2.6 – Діаграма бази даних

Таблиця 2.8

Опис таблиці body_types

Поле	Тип	Призначення
id	bigint(20)	ID Кузова
name	varchar(255)	Назва
created_at	timestamp	Коли створено
updated_at	timestamp	Коли змінено

Дана таблиця зберігає дані про типи кузовів автомобілів. Кожен запис представляє окремий тип кузова. Використовується для упорядкування автомобілів за форм-фактором. Кожен запис ідентифікується полем id, яке виконує роль первинного ключа. Фіксується час створення та останнього оновлення запису.

Таблиця 2.9

Опис таблиці bookings

Поле	Тип	Призначення
id	bigint(20)	ID бронювання
start_date	date	Дата початку
end_date	date	Дата завершення
number	varchar(255)	Номер телефону
from_time	varchar(255)	Час початку
end_time	varchar(255)	Час завершення
car_id	bigint(20)	ID Автомобіля
created_at	timestamp	Коли створено
updated_at	timestamp	Коли змінено

Таблиця bookings зберігає всю необхідну інформацію про бронювання автомобілів користувачами. Містить дату і час початку оренди, дату і час завершення оренди, ідентифікатор авто, а також номер телефону користувача.

Таблиця 2.10

Опис таблиці brands

Поле	Тип	Призначення
id	bigint(20)	ID Бренду
name	varchar(255)	Назва
slug	varchar(255)	URL-адреса
created_at	timestamp	Коли створено
updated_at	timestamp	Коли змінено

Таблиця brands використовується для зв'язку з таблицею cars, містить список автомобільних брендів, представлених у системі дозволяє користувачам знаходити транспортні засоби за певною маркою. Кожен бренд має ідентифікатор, назву та URL-дружню назву бренду для побудови зручних URL адрес, наприклад: /brands/ford.

Таблиця 2.11

Опис таблиці cache

Поле	Тип	Призначення
key	bigint(20)	ID запису
value	varchar(255)	Значення
expiration	varchar(255)	Дата закінчення строку

Таблиця 2.12

Опис таблиці cache_locks

Поле	Тип	Призначення
key	bigint(20)	ID запису
owner	varchar(255)	Власник
expiration	int(11)	Дата закінчення строку

Таблиці `cache` та `cache_locks` використовуються для кешування та тимчасового зберігання даних, завдяки ним менше навантажуються основні таблиці бази даних. Таблиця `cache` зберігає попередньо обчислені результати та проміжні значення, які не потребують постійного зберігання, `cache_locks` створена для керування блокуванням кешу з синхронізації доступу до даних між різними процесами. Поле `key` є ідентифікатором запису та слугує ключем для пошуку відповідного кешованого значення.

Таблиця 2.13

Опис таблиці `cars`

Поле	Тип	Призначення
<code>id</code>	<code>bigint(20)</code>	ID Автомобіля
<code>title</code>	<code>varchar(255)</code>	Назва
<code>slug</code>	<code>varchar(255)</code>	URL-адреса
<code>brand_id</code>	<code>bigint(20)</code>	ID Бренду
<code>price</code>	<code>double</code>	Ціна
<code>year</code>	<code>int(11)</code>	Рік випуску
<code>color</code>	<code>varchar(255)</code>	Колір
<code>zeroToHundred</code>	<code>double</code>	Час від 0 до 100 км/ч
<code>transmission</code>	<code>varchar(255)</code>	Трансмісія
<code>engine</code>	<code>varchar(255)</code>	Двигун
<code>maxSpeed</code>	<code>int(11)</code>	Максимальна швидкість
<code>horsepower</code>	<code>int(11)</code>	Кількість кінських сил
<code>seats</code>	<code>int(11)</code>	Кількість сидячих місць
<code>fuelType</code>	<code>varchar(255)</code>	Тип палива
<code>body_type_id</code>	<code>bigint(20)</code>	ID Типу кузова
<code>category_id</code>	<code>bigint(20)</code>	ID Категорії
<code>created_at</code>	<code>timestamp</code>	Коли створено
<code>updated_at</code>	<code>timestamp</code>	Коли змінено

Таблиця cars є основною в структурі бази даних веб-сервісу для оренди автомобілів, вона має зв'язки з іншими таблицями, які містять інформацію про автомобіль, а також зв'язана з бронюванням. Вона призначена для зберігання загальних відомостей про авто та їх технічних характеристик. Поле brand_id встановлює зв'язок із таблицею brands, вказуючи, до якого бренду належить автомобіль, category_id і body_type_id використовуються для класифікації авто за категорією наприклад, преміум, економ та типом кузова.

Таблиця 2.14

Опис таблиці cars_images

Поле	Тип	Призначення
id	bigint(20)	ID Фотографії автомобіля
car_id	varchar(255)	ID Автомобіля
path	varchar(255)	Посилання
created_at	timestamp	Коли створено
updated_at	timestamp	Коли змінено

Таблиця cars_images призначена для зберігання інформації про зображення автомобілів, представлених у сервісі оренди. Вона виконує допоміжну функцію у зв'язку з таблицею cars, до якої прив'язується через поле car_id. Поле Path містить шлях до файлу зображення, який використовується для завантаження фотографії авто на інтерфейс користувача.

Таблиця 2.15

Опис таблиці categories

Поле	Тип	Призначення
id	bigint(20)	ID Категорії
name	varchar(255)	Назва
created_at	timestamp	Коли створено
updated_at	timestamp	Коли змінено

Таблиця 2.16

Опис таблиці migrations

Поле	Тип	Призначення
id	int(10)	ID міграції
migration	varchar(255)	Міграція
batch	int(11)	Партія

Таблиця migrations виконує службову роль у процесі керування структурою бази даних під час розробки та підтримки веб-сервісу для оренди авто. Вона відстежує виконані міграції які реалізуються за допомогою фреймворку Laravel. [28]

Таблиця 2.17

Опис таблиці password_reset_tokens

Поле	Тип	Призначення
email	varchar(255)	Електронна пошта
token	varchar(255)	Токен
created_at	timestamp	Коли створено

Таблиця 2.18

Опис таблиці personal_access_tokens

Поле	Тип	Призначення
id	bigint(20)	ID Токена персонального доступу
tokenable_type	varchar(255)	Тип токену
tokenable_id	bigint(20)	ID запису моделі токену
name	varchar(255)	Назва
token	varchar(64)	Токен
abilities	text	Здібності
last_used_at	timestamp	Коли останній раз використовувався
expires_at	timestamp	Закінчення строку
created_at	timestamp	Коли створено
updated_at	timestamp	Коли змінено

Таблиця password_reset_tokens призначена для тимчасового зберігання даних, необхідних для відновлення паролю користувачів системи. Вона містить електронну адресу користувача, що ініціює запит на скидання паролю, та унікальний токен, який використовується для підтвердження права на зміну паролю.

Таблиця personal_access_tokens зберігає персональні токени доступу, які забезпечують автентифікацію користувачів. Сам токен зберігається у вигляді хешованого значення у полі token, також зберігається час останнього використання токена та термін його дії. Ця таблиця є одним з основних елементів при реалізації системи авторизації.

Таблиця 2.19

Опис таблиці sessions

Поле	Тип	Призначення
id	bigint(20)	ID Сесії
payload	blob	Корисне навантаження
last_activity	int(11)	Остання активність
user_id	bigint(20)	ID Користувача
ip_address	varchar(64)	IP адреса
user_agent	text	Агент користувача

Таблиця sessions використовується для зберігання даних про активні та нещодавні сесії користувачів веб-сервісу. Основне поле payload містить дані сесії та зміни тану, які були записані під час взаємодії користувача з системою. Поле last_activity відстежує момент останньої дії користувача, що допомагає при реалізації автоматичного завершення сесій. Ідентифікатор користувача user_id пов'язує сесію з конкретним користувачем, а поля ip_address та user_agent зберігають технічну інформацію про пристрій, з якого здійснено вхід у веб-застосунок.

Опис таблиці users

Поле	Тип	Призначення
id	bigint(20)	ID запису
name	varchar(255)	Ім'я
email	varchar(255)	Електронна пошта
number	varchar(255)	Номер телефону
password	varchar(255)	Пароль
remember_token	varchar(100)	Токен пам'яті
created_at	timestamp	Коли створено
updated_at	timestamp	Коли змінено
is_admin	tinyint(1)	Чи адмін

Таблиця users містить усю інформацію про користувачів, які взаємодіють з веб-сервісом для оренди авто. У таблиці зберігається ID користувача, ім'я, електронна пошта, номер телефону та токен пам'яті. Поле is_admin служить для розмежування рівнів доступу, значення 1 позначає адміністратора, а 0 – звичайного користувача. Подібним чином реалізована таблиця is_admin.

Таблиця 2.21

Опис таблиці is_admin

Поле	Тип	Призначення
id	bigint(20)	ID запису
name	varchar(255)	Назва
email	varchar(255)	Електронно пошта
number	varchar(255)	Номер телефону
password	varchar(255)	Пароль
remember_token	varchar(100)	Токен пам'яті
created_at	timestamp	Коли створено
updated_at	timestamp	Коли змінено
is_admin	tinyint(1)	Чи адмін

Таблиці в базі даних взаємодіють між собою через зв'язки (ключі). Кожна таблиця містить певний тип інформації, що відповідає логічній частині системи: автомобіль, користувач, бронювання. Основною перевагою реляційної бази даних є централізоване зберігання даних на сервері, це дозволяє забезпечити цілісність, узгодженість і контроль доступу до інформації. Кожен запис у таблиці є унікальним і пов'язаний із записами в інших таблицях за допомогою зовнішніх ключів. [5]

Основним файлом для конфігурації навігації та взаємодії з сервером виступає компонент App.jsx В додатку Б представлено фрагмент коду, який відповідає за: отримання даних з API, маршрутизацію сторінок, відображення відповідних компонентів інтерфейсу. Було використано хук useEffect, який викликається один раз після першого рендеру компонента. Він ініціює GET-запит до бекенду Laravel через /api/cars за допомогою бібліотеки axios. Отримані дані зберігаються в стані компонента через useState, у змінній data. Таким чином забезпечується динамічне завантаження автопарку при відкритті головної сторінки. [29]

Підсумовуючи було створено діаграму розгортання (рис. 2.7), яка ілюструє фізичне розміщення компонентів системи для оренди авто на серверному обладнанні та відображає взаємодію між апаратними вузлами та програмними елементами.

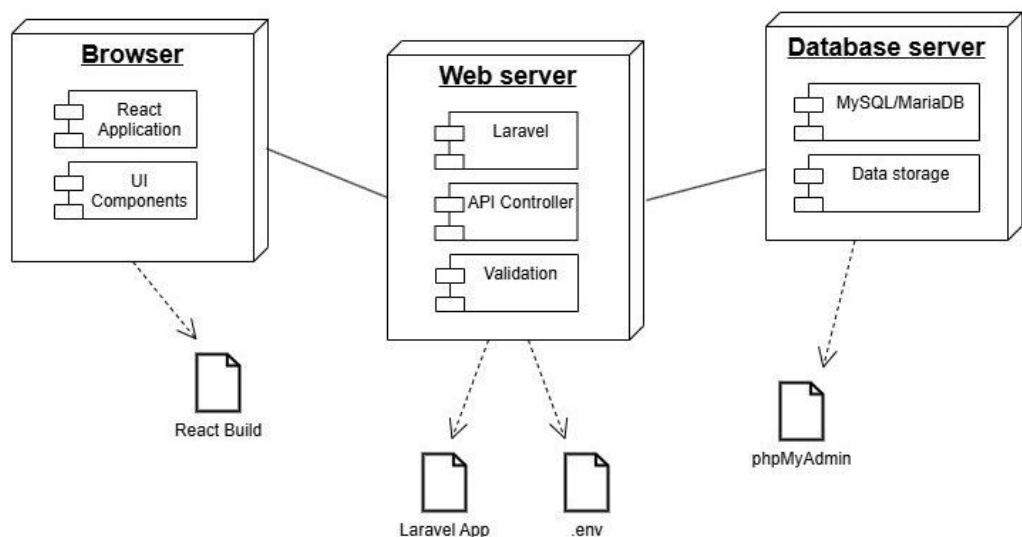


Рис. 2.7 – Діаграма розгортання веб-сервісу для оренди авто

Висновки до другого розділу

В результаті процесу проектування веб-сервісу для оренди автомобілів було зазначено основні сценарії використання сервісу що відповідає вимогам програмного забезпечення. На основі цього була складена архітектура клієнтської та серверної частин системи. Реалізовано частина фронтенду за допомогою React, а також RESTful API на базі Laravel, що обробляє запити та взаємодіє з базою даних. В ході проектування веб-сервісу для оренди автомобілів було визначено основні сценарії використання сервісу відповідно до вимог до програмного забезпечення. На основі цього була побудована архітектура клієнтської та серверної частин системи. Реалізована структура фронтенду за допомогою React, а також RESTful API на базі Laravel, що обробляє запити та взаємодіє з базою даних. Підсумовуючи, результатом реалізації проекту є досягнення поставленої мети – створити функціональний веб-сервіс для оренди автомобілів, який забезпечує зручну взаємодію користувача з системою, зберігання та обробку даних, а також працю всіх ключових модулів. Архітектура застосунку відповідає сучасним вимогам веб-розробки, що дозволяє легко масштабувати систему та інтегрувати додаткові можливості в майбутньому.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ФУНКЦІОНУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ РОЗРОБНЕЛОГО ВЕБ-СЕРВІСУ

3.1 Організація тестування та опис методик перевірки працездатності системи

Тестування програмного забезпечення є невід’ємною частиною життєвого циклу розробки інформаційних систем. Його основною метою є верифікація та валідація функціональності, стабільності та відповідності розробленого веб-застосунку вимогам користувача. Проведення тестування дозволяє визначити рівень готовності веб-застосунку до впровадження у продуктивне середовище.

Було описано план тестування, визначено цілі, об’єкти та критерії приймання, зазначено засоби тестування, середовище виконання, а також методики перевірки працездатності системи. Комплексна перевірка дозволила оцінити як функціональні можливості сервісу, так і його продуктивність, зручність інтерфейсу та відповідність сучасним вимогам до веб-застосунків. Проведене тестування дозволило не лише виявити й усунути критичні помилки, а й загалом оцінити якість розробленого програмного продукту перед його впровадженням у реальне середовище експлуатації.

План тестування веб-сервісу для оренди авто:

Метою тестування веб-сервісу для оренди авто є дослідження працездатності, надійності та відповідності функціональних і нефункціональних характеристик системи встановленим вимогам. Тестування проводиться для верифікації реалізованого функціоналу, перевірки правильності взаємодії між клієнтською та серверною частинами, а також виявлення можливих помилок у роботі програмного забезпечення.

Об’єктом тестування є веб-сервіс для оренди автомобілів, який складається з двох основних компонентів. Клієнтська частина реалізована за

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

допомогою бібліотеки React, серверна частина побудована на основі PHP-фреймворку Laravel, що відповідає за обробку запитів, логіку бізнес-процесів, роботу з базою даних, та забезпечення безпеки системи.

Види тестування: тестування інтерфейсу користувача, функціональне тестування основних сценаріїв взаємодії користувача з системою та тестування продуктивності. Середовище тестування: Операційна система Windows 11, Браузер: Google Chrome.

Інструментальні засоби: для тестування продуктивності застосовується інструмент Apache JMeter, який дозволяє змодельовати середовище з багатьма одночасними користувачами. Середовище розробки VS Code використовується для налагодження і ручного тестування окремих модулів як клієнтської так і серверної частини застосунку. Для контролю змін у базі даних та верифікації коректності виконання запитів застосовується веб- інтерфейс phpMyAdmin.

Вимоги до програмного забезпечення:

Node.js, npm, PHP 8+, Composer, MySQL, Laravel, React, XAMPP.

Вимоги до апаратного забезпечення: процесор не нижче Intel Core i5, оперативна пам'ять від 8 ГБ, місце на диску від 1 ГБ.

Критерії приймання веб-сервісу для оренди автомобілів визначають перелік умов, за дотримання яких система вважається готовою до впровадження в експлуатаційне середовище. Основним критерієм є повна відповідність реалізованого функціоналу вимогам, та коректна робота всіх компонентів. Прийнятною вважається наявність лише незначних недоліків, які не впливають на загальну працездатність системи. Ще одним обов'язковим критерієм є забезпечення можливості безперешкодного проходження повного користувацького сценарію: вхід до системи, вибір авто, оформлення бронбвання.

Для перевірки працездатності веб-сервісу були застосовані різні методики тестування, які дозволяють оцінити функціональні і нефункціональні характеристики системи.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

Функціональне тестування – це тип тестування програмного забезпечення, спрямований на перевірку функцій або функціональних вимог програми. Метою є переконатися, що система виконує завдання, визначені у технічних вимогах, і все працює так, як це передбачено.

Для оцінки продуктивності системи було здійснено навантажувальне тестування – автоматизоване тестування, яке імітує роботу певної кількості користувачів на загальному ресурсі. Тестування стабільності та надійності – має на меті перевірку працездатності додатка під час тривалого тестування з середнім рівнем навантаження.

Тестування інтерфейсу користувача – це процес перевірки функціональності графічного інтерфейсу програмного забезпечення. Основна мета полягає у забезпеченні коректної роботи всіх візуальних елементів, таких як кнопки, меню, поля для введення тексту, лейбли, панелі навігації та інші, а також правильність їх взаємодії з користувачем. [31]

3.2 Оцінювання зручності використання, продуктивності та надійності сервісу

Основною метою цього етапу є перевірка функціональності, зручності інтерфейсу, швидкодії та надійності сервісу в умовах стандартного навантаження. Було здійснено комплексне тестування, спрямоване на верифікацію наступних сценаріїв: реєстрація та авторизація користувачів, реєстрація з невалідними даними, бронювання автомобіля.

Тестування виконувалося вручну за попередньо складеними тест-кейсами, які описані в таблицях 3.1 – 3.4 в них описані дії, які необхідно виконати для перевірки конкретної функціональності програмного забезпечення.

Опис тест-кейсу №1.

Тестовий сценарій: Зареєструватися на сайті RoyalRent.ua.

Вхідні дані:

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

- Ім'я користувача: Владислав Кузьмін;
- Електронна пошта: testuser@gmail.com;
- Номер телефону: +380686840258;
- Пароль: Password123.

Таблиця 3.1

Тест-кейс №1

Крок №	Інструкції	Очікувані результати	Результат	Статус
1	Відкрити головну сторінку сайту	Сторінка відкриється	Успішно	Пройдено
2	Натиснути на кнопку "Реєстрація"	Сторінка відкриється	Успішно	Пройдено
3	Ввести електронну пошту, номер телефону пароль і підтвердження паролю	Значення будуть коректними	Успішно	Пройдено
4	Натиснути кнопку "Зареєструватися"	Користувач зареєстрований	Успішно	Пройдено

Метою даного тест-кейсу є перевірка працездатності функціоналу реєстрації користувача через клієнтський інтерфейс веб-сервісу. Тестування орієнтоване на забезпечення коректної взаємодії між формою реєстрації, механізмами валідації та бекендом, що обробляє запити на створення нового облікового запису. Тестовий сценарій охоплює повну послідовність дій від моменту відкриття головної сторінки до успішного завершення реєстрації.

Результати показали, що процедура реєстрації нового користувача виконується у повній відповідності до функціональних вимог. Усі компоненти інтерфейсу спрацювали стабільно, а форма валідувала дані належним чином.

В результаті після проходження реєстрації в базі даних (рис. 3.2) з'явився новий користувач з електронною поштою testuser@gmail.com.

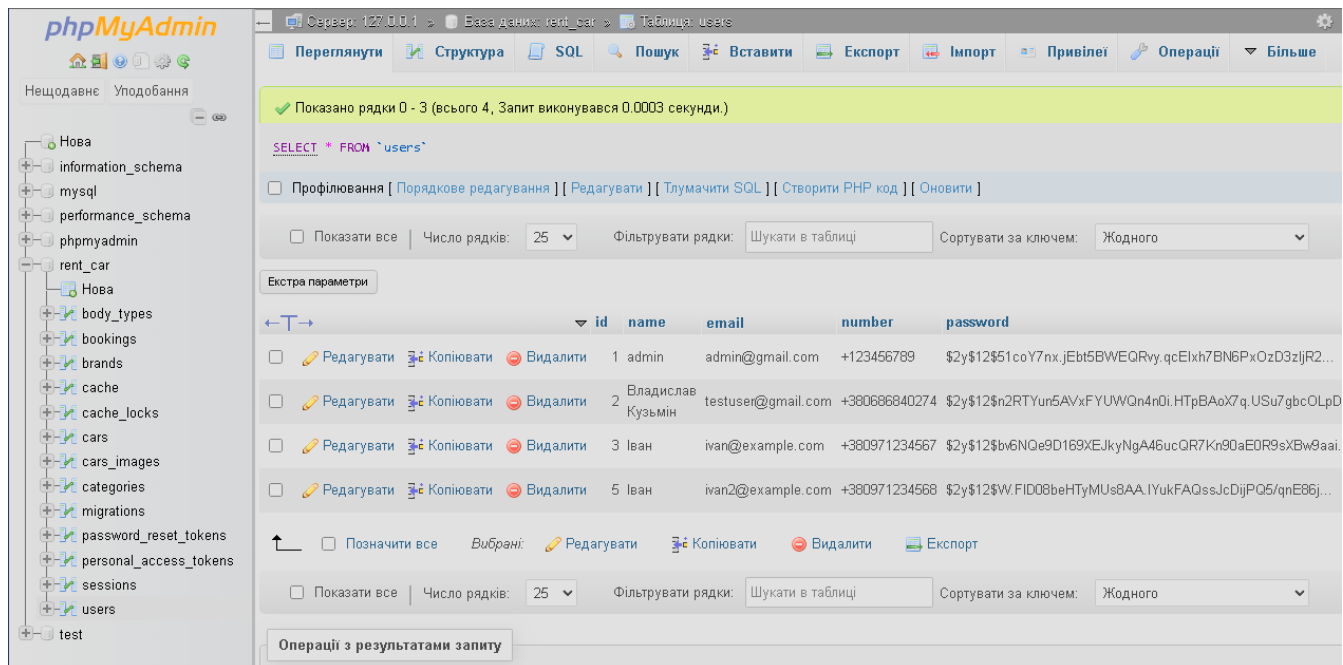


Рис. 3.2 – Таблиця користувачів

Для захисту паролів у базі даних виконується функція хешування, завдяки цьому навіть у разі злому бази даних або отримання доступу до неї сторонніми особами, реальні паролі користувачів залишаються недоступними, оскільки зловмисник бачить лише хешовані значення, які не мають прямого зв'язку з оригінальними паролями.

Після реєстрації нового користувача відкривається сторінка входу в аккаунт, перевірка авторизації описана в таблиці 3,2.

Опис тест-кейсу №2.

Тестовий сценарій: Виконати вхід в аккаунт

Вхідні дані:

- Ім'я користувача: Владислав Кузьмін;
- Електронна пошта: testuser@gmail.com;
- Пароль: Password123.

Тест-кейс №2

Крок №	Інструкції	Очікувані результати	Результат	Статус
1	Відкрити головну сторінку сайту	Сторінка відкриється	Успішно	Пройдено
2	Натиснути на кнопку "Увійти"	Сторінка відкриється	Успішно	Пройдено
3	Ввести електронну пошту і пароль	Значення будуть коректними	Успішно	Пройдено
4	Натиснути кнопку "Увійти"	Вхід у профіль користувача	Успішно	Пройдено

Цей тест-кейс має на меті перевірку функціональності модуля автентифікації веб-сервісу, а саме можливості входу в систему під обліковим записом зареєстрованого користувача. Тестування передбачає послідовне виконання дій, які дозволяють авторизуватися у системі, перевіряючи коректність валідації введених облікових даних.

Після успішної авторизації користувача направляє на сторінку особистого профілю (рис. 3.3), де він може переглянути та змінити свої дані.

Рис. 3.3 – Профіль користувача

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		61

Також був протестований сценарій вводу некоректних даних, щоб оцінити рівень валідації і переконатися в тому, що система правильно реагує на непередбачувані дії користувача. Результати наведені в рис. 3.4.

Опис тест-кейсу №3.

Тестовий сценарій: Заповнити поля реєстрації невалідними даними

Вхідні дані:

- Ім'я користувача: Пустий рядок;
- Електронна пошта: vladkuzmin@gb;
- Номер телефону: +380686840258;
- Пароль: 1234.

Таблиця 3.3

Тест-кейс №3

Крок №	Інструкції	Очікувані результати	Результат	Статус
1	Відкрити головну сторінку сайту	Сторінка відкриється	Успішно	Пройдено
2	Натиснути на кнопку "Реєстрація"	Сторінка відкриється	Успішно	Пройдено
3	Ввести електронну пошту, номер телефону пароль і підтвердження паролю	Значення будуть некоректними, повідомлення про неправильний формат та обов'язкові поля	Успішно	Пройдено

Метою даного тест-кейсу є перевірка роботи валідації при спробі заповнення форми реєстрації некоректними даними, оцінка здатності системи виявляти й обробляти помилки введення та захисту від потенційно небезпечної інформації. У межах тестового сценарію користувач переходить до форми реєстрації та вводить зазначені невалідні дані. Результати виконання тесту підтвердили коректну роботу механізмів перевірки вхідних даних.

РЕЄСТРАЦІЯ

EMAIL

vladkuzmin@gb

Невірний формат електронної пошти

НОМЕР ТЕЛЕФОНУ

+380 50 123 4567

Обов'язкове поле

ІМ'Я

Введіть ваше ім'я

Ім'я обов'язкове

ПАРОЛЬ

Пароль має бути не менше 6 символів

Рис. 3.4 – Введення некоректних даних у форму реєстрації

Наступним важливим сценарієм є процес бронювання авто, необхідно перевірити чи зберігаються дані про нове бронювання у базі даних, результат зображено на рис. 3.5 та 3.6.

Опис тест-кейсу №4.

Тестовий сценарій: Здійснити бронювання автомобіля

Вхідні дані:

- Ім'я користувача: Владислав Кузьмін;
- Електронна пошта: testuser@gmail.com;
- Номер телефону: +380686840274;
- Дата і час початку: 19.06.2025, 13:00;
- Дата завершення: 22.06.2025.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 3.4

Тест-кейс №4

Крок №	Інструкції	Очікувані результати	Результат	Статус
1	Відкрити головну сторінку сайту	Сторінка відкриється	Успішно	Пройдено
2	Натиснути на категорію "Спортивні"	Відкриється список авто	Успішно	Пройдено
3	Натиснути на перший автомобіль	Відкривається сторінка з інформацією про авто з вікном бронювання	Успішно	Пройдено
4	Обрати дату початку та завершення прокату	Значення будуть коректними	Успішно	Пройдено
5	Натиснути кнопку "Орендувати"	Перехід на сторінку оплати	Успішно	Пройдено

Цей тест-кейс має на меті перевірити коректність функціоналу бронювання автомобіля. Перевіряється працездатність взаємодії з інтерфейсом користувача, обробка даних та завершення бронювання переадресацією на сторінку оплати. Після заповнення всіх полів, користувач натискає кнопку «Орендувати». У разі коректного функціонування застосунку та відсутності помилок на клієнтській і серверній стороні, система обробляє запит і направляє користувача на сторінку оплати.

У результаті виконання всіх дій, функціональність бронювання автомобіля спрацювала як очікувалося. Інформація про нове бронювання зберіглася у базі даних. Усі етапи пройдено успішно, жодних збоїв або помилок не виявлено, це свідчить про стабільну роботу ключового модуля застосунку.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

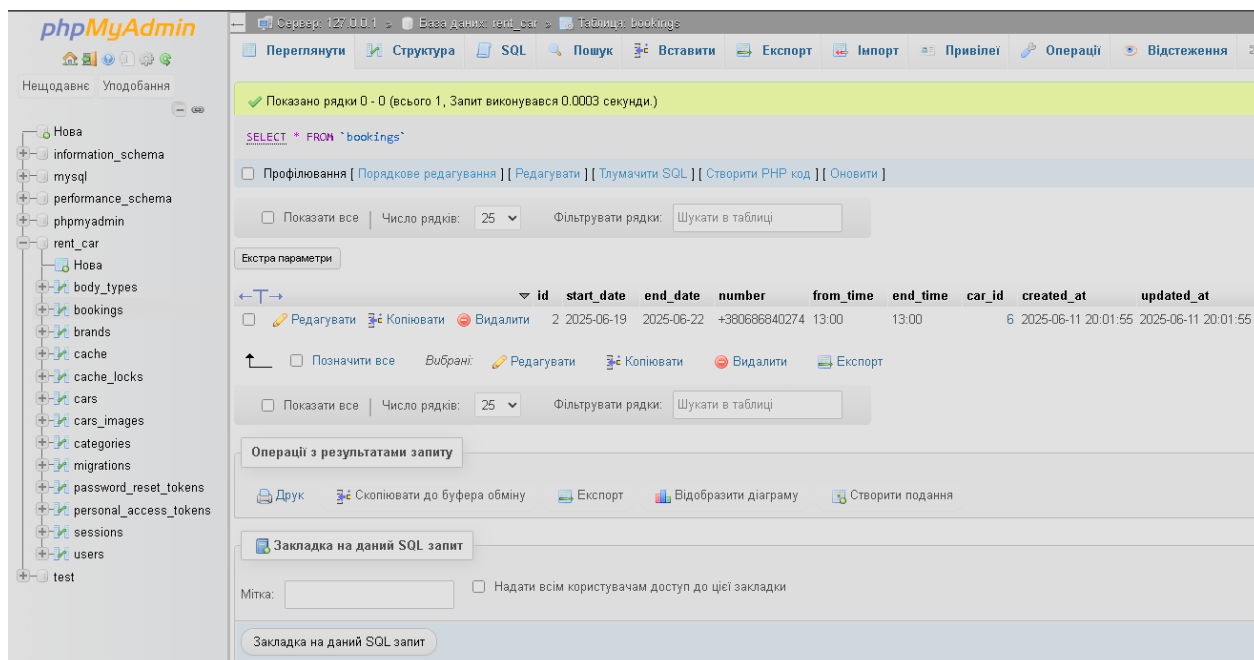


Рис. 3.5 – Інформація про бронювання в БД

Для забезпечення якості функціонування веб-сервісу було проведено навантажувальне тестування та тестування стабільності і надійності. Ці види тестування є важливими для визначення здатності системи працювати в умовах реальної експлуатації.

За допомогою інструменту Apache JMeter було імітовано різні сценарії, що включали збільшення кількості віртуальних користувачів і частоти запитів. Тестування стабільності і надійності спрямоване на перевірку безперервної роботи сервісу протягом тривалого часу за помірного рівня навантаження. Це тестування дало змогу оцінити, як система витримує тривалі цикли використання, наскільки стабільним залишається час відповіді та чи виникають збої або деградація продуктивності в процесі роботи. (Див. додаток В) [32]

Налаштування першої групи користувачів:

Кількість користувачів (Threads): 50, час розгону (Ramp-Up Period): 10 секунд, повторів (Loop Count): 10.

Результати навантажувального тесту в таблиці Summary Report (рис. 3.3):

Кількість запитів: 500, середній час відповіді: 10.204сек., мінімальний час: 363мс., максимальний час: 11.801сек., відсоток помилкових запитів: 0%; продуктивність: 4.3 запитів/сек.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	500	10.04	343	11801	2618.66	0.00%	4.3/sec	46.22	0.51
TOTAL	500	10.04	343	11801	2618.66	0.00%	4.3/sec	46.22	0.51

Рис. 3.3 – Результат навантажувального тесту

Результати тестування показують, що за умови запуску через вбудований сервер php artisan serve, сайт продемонстрував повну відсутність помилкових відповідей при навантаженні в 50 одночасних клієнтів, але варто зазначити, що час відповіді був доволі високим – понад 10 секунд у середньому, це є ознакою обмеженої обробної здатності в умовах локального середовища. Ці результати свідчать про працездатність API, і також підкреслюють необхідність розгортання сервісу у повноцінному веб-серверному середовищі для досягнення прийнятної швидкодії в умовах реальної експлуатації.

Налаштування другої групи користувачів:

Кількість користувачів (Threads): 20, час розгону (Ramp-Up Period): 30 секунд, повторів (Loop Count): безкінечність.

Результати тестування стабільності і надійності протягом 2 годин, 36 хвилин в таблиці Summary Report (рис. 3.4):

Кількість запитів: 38898, середній час відповіді: 4.815сек., мінімальний час: 200мс., максимальний час: 8.391сек., відсоток помилкових запитів: 0,05%, продуктивність: 4.1 запитів/сек.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

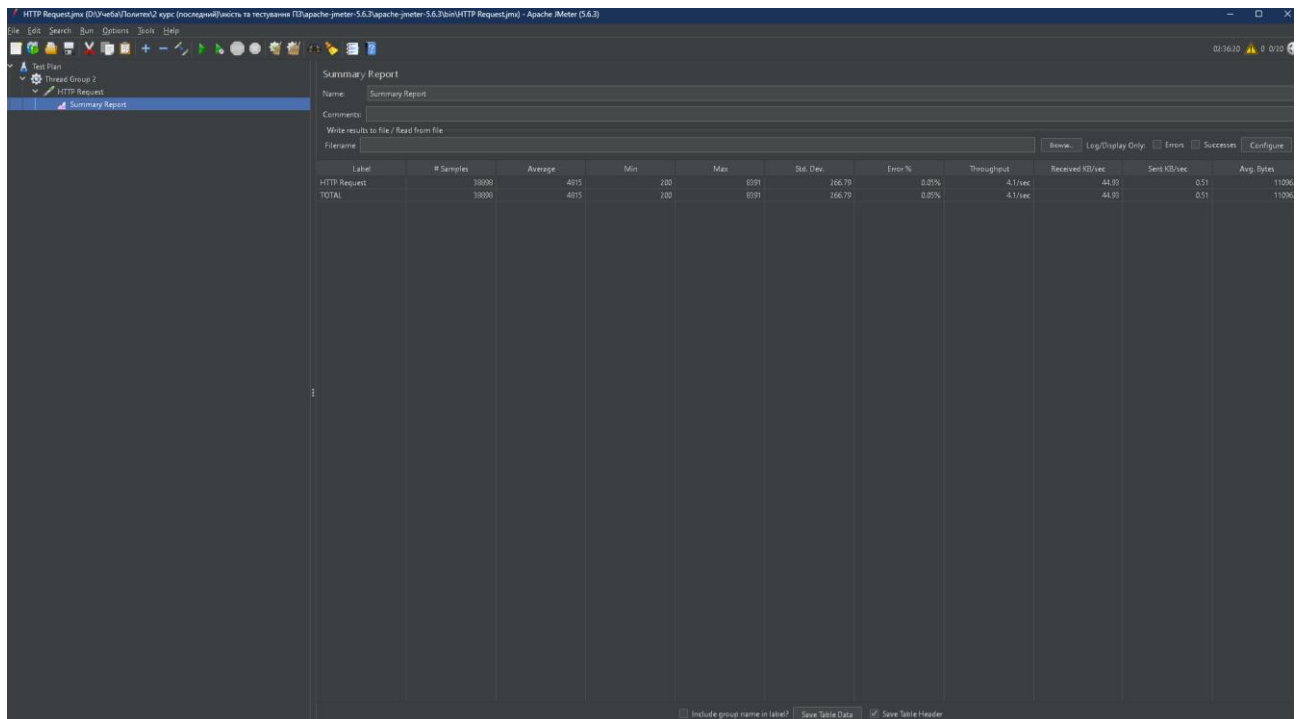


Рис. 3.4 – Результат тестування стабільності і надійності

Результати тестування стабільності і надійності свідчать про здатність системи обробляти велику кількість звернень за тривалий час. Середній час відповіді становив 4.815 секунди, цей показник є прийнятним для більшості веб-застосунків, але свідчить про певну затримку, через навантаження.

3.3 Напрями подальшого вдосконалення програмного продукту

У процесі розробки веб-додатку для оренди автомобілів були успішно реалізовані ключові функціональні модулі, які забезпечують базову взаємодію користувачів із системою. Серед них створення облікового запису шляхом реєстрації, можливість входу до системи з використанням авторизації, перегляд доступних транспортних засобів із відповідною інформацією, а також оформлення замовлення на оренду автомобіля. Реалізований функціонал дозволяє забезпечити основні потреби користувачів та досягти належного рівня автоматизації процесу бронювання.

Одним із перспективних напрямів удосконалення веб-сервісу для оренди автомобілів є створення повноцінного мобільного застосунку, який дозволить підвищити доступність системи, забезпечити зручність користування з будь-

якого місця та задовольнити очікування сучасних користувачів. У контексті зростання попиту на мобільні платформи, наявність відповідного застосунку є важливою умовою для конкурентоспроможності бізнесу у сфері прокату авто. З точки зору користувацького досвіду, мобільний застосунок дозволяє не лише зручно здійснювати бронювання в дорозі, а й отримувати push-сповіщення про статус замовлення та нагадування про завершення оренди.

Також перспективним напрямом розвитку є вдосконалення функціоналу особистого кабінету користувача. На поточному етапі особистий кабінет забезпечує базову взаємодію з системою, проте його розширення дозволить значно покращити індивідуальний досвід користувачів. Зокрема, доцільно реалізувати можливість перегляду детальної статистики витрат на оренду автомобілів, історії поїздок із відображенням дати, тривалості та вартості кожної поїздки, а також запровадити систему накопичення бонусів або знижок за часте використання сервісу. Окрім цього, важливо надати користувачеві зручні інструменти для управління своїми замовленнями: перегляд поточних і завершених бронювань, можливість скасування або продовження оренди, а також отримання сповіщень про зміни в статусі замовлення. Такий функціонал не лише підвищить зручність користування сервісом, а й сприятиме формуванню довготривалих відносин із клієнтами, що позитивно вплине на рівень їхньої лояльності та загальну конкурентоспроможність веб-додатку.

Інтеграція з платіжними є дуже важливим вдосконаленням веб-сервісу для оренди автомобілів, вона дасть змогу автоматизувати процес оплати і зменшити кількість ручних операцій, це значно підвищить зручність користування для клієнтів. Запровадження можливості здійснювати онлайн-оплати безпосередньо через інтерфейс застосунку сприятиме підвищенню довіри з боку користувачів та скороченню часу оформлення оренди. На сьогоднішній день найбільш поширеними платіжними сервісами, з якими можлива інтеграція, є такі платформи: LiqPay, Stripe, Fondy, PayPal, Apple Pay та Google Pay. Вибір конкретної платіжної системи залежить від цільового ринку, законодавчих вимог, а також технічної сумісності з існуючою

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

архітектурою проекту. Впровадження онлайн-оплати може вимагати перегляду частини логіки бекенду, зокрема створення webhook-обробників для реакції на події з боку платіжної системи, таких як підтвердження платежу, відмова або повернення коштів. У контексті Laravel цей процес може бути реалізований з використанням бібліотек налаштування взаємодії з REST API відповідного платіжного сервісу. Інтеграція з платіжною системою передбачає реалізацію механізмів верифікації платіжних даних, захисту інформації користувача, а також фіксації результатів транзакцій у базі даних з можливістю подальшої перевірки.

Важливим напрямом удосконалення є підвищення рівня інформаційної безпеки, що має критичне значення для забезпечення надійності системи, захисту персональних даних користувачів та запобігання несанкціонованому доступу до внутрішніх ресурсів. У зв'язку з цим у веб-сервіс для оренди авто планується впровадження двофакторної автентифікації, яка суттєво знизить ризики компрометації облікових записів навіть у випадку втрати або викрадення пароля. Двофакторної автентифікація передбачає додаткову перевірку особи користувача через підтвердження за допомогою одноразового коду, що надсилається на мобільний телефон або електронну пошту.

Висновки до третього розділу

В третьому розділі відбулось проведення експериментальної перевірки функціонування веб-сервісу для оренди авто та оцінювання його ефективності, було проведено тестування, яке включало перевірку функціоналу та продуктивності сайту. Для навантажувального тестування було використано інструмент Apache JMeter, також проведено ручне тестування функціоналу на основі прописаних сценаріїв. Під час тестування було визначено, що всі основні функції системи працюють відповідно до визначених вимог, таких як успішна реєстрація користувача, перегляд доступних автомобілів, здійснення бронювання і проходження повного циклу оренди без технічних збоїв. Виявлено стабільну роботу сервісу під середніми навантаженнями, що каже про

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

його надійність і готовність до практичного використання. Також оцінено зручність використання системи, яка була успішно сприйнята тестовими користувачами завдяки інтуїтивно зрозумілому інтерфейсу та швидкому відгуку додатку. На основі отриманих результатів визначено напрями подальшого вдосконалення, які вміщують покращення продуктивності при високих навантаженнях, інтеграцію додаткових функцій та підвищення безпеки користувацьких даних.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Під час виконання випускної кваліфікаційної роботи бакалавра було проведено проектування та реалізація веб-сервісу для оренди авто. У першому розділі здійснено теоретичні дослідження й аналітичне обґрунтування потреби у створенні такого сервісу. Розглянуто сучасний стан розвитку цифрових сервісів у сфері автопрокату, здійснено порівняльний аналіз функціональних можливостей і недоліків існуючих веб-платформ. Результатом проведеного аналізу є визначені основні вимоги до архітектури інформаційної системи, що стали основою практичної реалізації.

У другому розділі проведено проектування інформаційної системи та реалізацію її основних програмних компонентів. Обґрунтовано вибір архітектурно-технологічної платформи та технологічних засобів, якими є Laravel для серверної частини та React для клієнтської. Було створено функціональні сценарії роботи користувачів та побудовано об'єктно-орієнтовану модель системи. Реалізовано клієнтську та серверну частини веб-сервісу, забезпечено обробку запитів і відповіді через REST API. Розроблено структуру бази даних та забезпечено взаємодію з нею для зберігання та обробки інформації про автомобілі, користувачів та бронювання.

У третьому розділі в ході експериментальної перевірки працездатності розробленого веб-сервісу було отримано результати відповідності розробленого продукту сучасним вимогам швидкості завантаження, зручності інтерфейсу та стабільності роботи. Також обговорено напрями подальшого вдосконалення веб-сервісу, що передбачають інтеграцію нових функцій, підвищення безпеки та масштабування системи. Таким чином, у результаті виконання кваліфікаційної роботи досягнуто поставленої мети – розробка веб-сервісу для оренди автомобілів, що відповідає вимогам сучасного ринку та забезпечує зручність користування як для клієнтів, так і для адміністраторів. Завдання дослідження, аналіз існуючих сервісів, визначення архітектури та реалізація функціональних компонентів, було виконано у повному обсязі.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sandeep J. "Client-Server Architecture". Geeksforgeeks. Client-Server Architecture - System Design. 16.07.2024. URL: <https://www.geeksforgeeks.org/client-server-architecture-system-design/> (дата звернення: 02.05.2025).
2. Lokesh G. "REST API". Restfulapi. What is REST?. 01.04.2025. URL: <https://restfulapi.net/> (дата звернення: 10.05.2025).
3. Gamze K. "Client-Server". Medium. The Client-Server Communication in The Context of React Frontend. 17.09.2024. URL: <https://medium.com/@gamzekandara/the-client-server-communication-in-the-context-of-react-frontend-and-flask-backend-75c540911807> (дата звернення: 03.05.2025).
4. Salma A.-N. "Modular architecture". Netlify. Modular web design architecture: what it is and why you need it. 17.05.2023. URL: <https://www.netlify.com/blog/modular-web-design-architecture/> (дата звернення: 02.05.2025).
5. Bill G. "Database design". Microsoft. Database design basics. 22.06.2023. URL: <https://support.microsoft.com/en-us/office/database-design-basics-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5> (дата звернення: 03.05.2025).
6. Arjun D. "Laravel Routing". Laravel. Routing. 03.11.2022. URL: <https://laravel.com/docs/12.x/routing#main-content> (дата звернення: 04.05.2025).
7. Sandeep J. "Architecture". Geeksforgeeks. Architecture of a System. 31.01.2023. URL: <https://www.geeksforgeeks.org/architecture-of-a-system/> (дата звернення: 05.05.2025).
8. Jan E. R. "SQL Injection". W3schools. SQL Injection. 24.08.2019. URL: https://www.w3schools.com/sql/sql_injection.asp (дата звернення: 05.05.2025).
9. Vikram S. "Cross-site scripting (XSS)". MND web docs. 15.02.2024. URL: <https://developer.mozilla.org/en-US/docs/Web/Security/Attacks/XSS> (дата звернення: 06.05.2025).

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

10. Sandeep J. “CSRF”. What is Cross Site Request Forgery (CSRF). Geeksforgeeks. 11.06.2025. URL: <https://www.geeksforgeeks.org/what-is-cross-site-request-forgery-csrf/> (дата звернення: 06.05.2025).

11. Побережний О. “SPA”. Asabix. Що таке SPA додаток?. 06.03.2024. URL: <https://asabix.com.ua/what-is-a-single-page-application/> (дата звернення: 08.05.2025).

12. Освітній портал ДУ “Житомирська політехніка” – курс “Бази даних”. – URL: <https://learn.ztu.edu.ua/course/view.php?id=40> (дата звернення: 10.05.2025).

13. Sandeep J. “Monolithic Architecture”. Geeksforgeeks. Monolithic Architecture - System Design. 05.10.2024. URL: <https://www.geeksforgeeks.org/monolithic-architecture-system-design/> (дата звернення: 07.05.2025).

14. Jan E. R. “HTTP”. w3schools. What is HTTP?. 13.10.2022. URL: https://www.w3schools.com/whatis/whatis_http.asp (дата звернення: 10.06.2025).

15. Sandeep J. “Microservice Architecture”. Geeksforgeeks. Microservice Architecture - Introduction, Challenges & Best Practices. 03.05.2025. URL: <https://www.geeksforgeeks.org/microservice-architecture-introduction-challenges-best-practices/> (дата звернення: 11.05.2025).

16. Шайтан В. MVC. “Ithillel”. Побудова динамічних вебзастосунків за допомогою MVC. 14.11.2023. URL: <https://blog.ithillel.ua/articles/building-dynamic-web-applications-using-mvc> (дата звернення: 13.05.2025).

17. Pranav K. “JavaScript”. Geeksforgeeks. Introduction to JavaScript. 12.05.2025. URL: <https://www.geeksforgeeks.org/javascript/introduction-to-javascript/> (дата звернення: 15.05.2025).

18. Deepak H. “React learn”. React. Quick Start. 06.02.2025. URL: <https://react.dev/learn> (дата звернення: 17.05.2025).

19. Lokesh G. “HTTP Methods”. Restfulapi. REST guidelines suggest using a specific HTTP method on a particular type of call made to the server i.e. GET,

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

POST, PUT or DELETE.. 04.11.2023. URL: <https://restfulapi.net/http-methods/>
(дата звернення: 14.06.2025).

20. Joomla. “XAMPP Introduction”. 14.05.2020. URL:
<https://docs.joomla.org/XAMPP> (дата звернення: 03.05.2025).

21. Abhishek Y. “phpMyAdmin’s documentation”. 09.10.2024. URL:
<https://docs.phpmyadmin.net/en/latest/> (дата звернення: 15.05.2025).

22. Rajat B. “MySQL”. MySQL 8.0 Reference Manual. 04.02.2024.
URL: <https://dev.mysql.com/doc/refman/8.0/en/> (дата звернення: 20.05.2025).

23. Devansh F. “Apache HTTP Server Documentation”. 12.07.2024. URL:
<https://httpd.apache.org/docs/> (дата звернення: 16.05.2025).

24. Neeraj X. “XAMPP official website”. 16.09.2023. URL:
<https://www.apachefriends.org/> (дата звернення: 23.05.2025).

25. Mohit W. “Laravel. Configuration”. 22.04.2023. URL:
<https://laravel.com/docs/11.x/configuration> (дата звернення: 09.05.2025).

26. Tarun Z. “Laravel”. Eloquent: Getting Started. 19.07.2023. URL:
<https://laravel.com/docs/12.x/eloquent> (дата звернення: 18.05.2025). 28. IEEE
Access – “Microservice Architectures” in Edge/Fog IoT. – URL:
<https://ieeexplore.ieee.org/document/XYZ> (дата звернення: 26.05.2025).

27. Anuj C. “Laravel. Middleware”. 03.11.2024. URL:
<https://laravel.com/docs/12.x/middleware> (дата звернення: 22.05.2025).

28. Rahul P. “Laravel. Database”: Migrations. 15.03.2023. URL:
<https://laravel.com/docs/12.x/migrations> (дата звернення: 23.05.2025).

29. Nikhil R. “JSX”. reactjs. Introducing JSX. 08.04.2025. URL:
<https://legacy.reactjs.org/docs/introducing-jsx.html> (дата звернення: 28.05.2025).

30. Arjun D. “Visualstudio”. Visual Studio Code documentation.
23.11.2023. URL: <https://code.visualstudio.com/docs> (дата звернення: 26.04.2025).

					ПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

31. Rohan M. “UI Testing”: A Detailed Guide. Browserstack. 14.01.2025.
URL: <https://www.browserstack.com/guide/ui-testing-guide> (дата звернення: 03.06.2025).
32. Anuj C. “Jmeter.apache”. User’s Manual. 18.04.2022. URL: <https://jmeter.apache.org/usermanual/index.html> (дата звернення: 03.06.2025).
33. Мартін Р. “Чистий код” : навч. посіб. Київ : Діалектика, 2019. – ст. 83–99.
34. Джін Б., Сахні С., Шеват А. “Designing Web APIs” : навч. посіб. Київ : O'Reilly / Лабораторія, 2021. – ст. 112–119.
35. Больє А. “Learning SQL” : навч. посіб. Київ : O'Reilly / Лабораторія, 2020. – ст. 42–2.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

Додаток А містить маршрути до API, реалізовані за допомогою фреймворку Laravel, які виконують основну логіку роботи бекенду та є частиною REST API для веб-сервісу, кожен маршрут відповідає за обробку конкретних запитів та викликає відповідний метод контролера.

Запити, що стосуються клієнтської частини застосунку:

```
Route::get('/cars', [GetCarsData::class, 'index']);
Route::get('/cars/{slug}/{id}/detail', [CarController::class, 'show']);
Route::post('/bookings', [BookingController::class, 'store']);
Route::get('/brands', [BrandController::class, 'index']);
Route::get('/car_images', [CarImagesController::class, 'index']);

Route::get('/cars/search', [CarController::class, 'fuzzySearch']);

Route::get('/brands/{brand}', [BrandController::class, 'api']);
Route::get('/types/{type}', [TypeController::class, 'api']);
Route::get('/body-types/{bodyType}', [BodyTypeController::class, 'api']);
Route::get('/check-car-availability/{carId}', [BookingController::class, 'checkCarAvailability']);
Route::get('/booked-dates/{carId}', [BookingController::class, 'getBookedDates']);

Route::post('/register', [UserController::class, 'register']);
Route::post('/check-email', [UserController::class, 'checkEmail']);
Route::post('/check-phone', [UserController::class, 'checkPhone']);

Route::post('/check-user', [UserController::class, 'checkUser']);
Route::post('/find-email/{email}', [UserController::class, 'findEmail']);
Route::middleware('auth:sanctum')->get('/user', [UserController::class, 'getUser']);
Route::middleware('auth:sanctum')->put('/user-update', [UserController::class, 'userUpdate']);
Route::middleware('auth:sanctum')->post('/check-password', [UserController::class, 'checkPassword']);
Route::middleware('auth:sanctum')->put('/change-password', [UserController::class, 'changePassword']);

Route::post('/forgot-password', [PasswordResetLinkController::class, 'store']);
Route::post('/reset-password', [NewPasswordController::class, 'store']);
```

```
function App() {
  const [data, setData] = useState([]);

  useEffect(() => {
    axios
      .get("http://127.0.0.1:8000/api/cars")
      .then((response) => {
        setData(response.data);
      })
      .catch((error) => {
        console.error("There was an error fetching the data!", error);
      });
  }, []);

  return (
    <HelmetProvider>
      <Router>
        <Routes>
          <Route path="/" element={<Index data={data} /> } />
          <Route path="/cars/:slug/:id/detail" element={<CarPage /> } />
          <Route path="/privacy-policy" element={<PrivacyPolicy /> } />
          <Route path="/terms-conditions" element={<TermsConditions /> } />
          <Route path="/login" element={<Login /> } />
          <Route path="/register" element={<Register /> } />
          <Route path="/password-reset" element={<PasswordReset /> } />
          <Route path="/reset-password/:token" element={<PasswordResetForm /> } />
          <Route path="/brands/:brand" element={<Brand /> } />
          <Route path="/types/:type" element={<Type /> } />
          <Route path="/body-types/:bodyType" element={<BodyType /> } />
          <Route path="/all-brands" element={<Brands /> } />
          <Route path="/user-dashboard" element={<UserDashboard /> } />

          <Route path="*" element={<NotFound /> } />
          <Route path="/page/not-found" element={<NotFound /> } />
          <Route path="/logout" element={<Logout /> } />
        </Routes>
      </Router>
    </HelmetProvider>
  );
}
```

Запити панелі адміністратора веб-сервісу для оренди авто:

```
Route::get('/access_denied', fn() => view('admin.access_denied'))->name('admin.access_denied');

Route::prefix('admin')->namespace('App\Http\Controllers\Admin')->group(function () {
    Route::middleware([RedirectIfAuthenticated::class])->group(function () {
        Route::get('/login_admin', [AuthAdminController::class, 'showLoginForm'])->name('admin.login');
        Route::post('/login_admin', [AuthAdminController::class, 'login']);
    });

    Route::middleware([Authenticate::class, 'is_admin'])->group(function () {
        Route::get('/', [DashboardController::class, 'index'])->name('admin.dashboard');
        Route::get('/profile', ProfileController::class->name('admin.profile'));

        // for cars
        Route::get('/cars', [CarsController::class, 'index'])->name('admin.cars.index');
        Route::post('/cars', [CarsController::class, 'store'])->name('admin.cars.store');
        Route::get('/cars/create', [CarsController::class, 'create'])->name('admin.cars.create');
        Route::get('/cars/{car}/edit', [CarsController::class, 'edit'])->name('admin.cars.edit');
        Route::put('/cars/{car}', [CarsController::class, 'update'])->name('admin.cars.update');
        Route::delete('/cars/{car}', [CarsController::class, 'destroy'])->name('admin.cars.destroy');

        // for users
        Route::get('/users', [UsersController::class, 'index'])->name('admin.users.index');
        Route::get('/users/{user}/edit', [UsersController::class, 'edit'])->name('admin.users.edit');
        Route::put('/users/{user}', [UsersController::class, 'update'])->name('admin.users.update');
        Route::delete('/users/{user}', [UsersController::class, 'destroy'])->name('admin.users.destroy');

        // for brands
        Route::get('/brands', [BrandsController::class, 'index'])->name('admin.brands.index');
        Route::post('/brands', [BrandsController::class, 'store'])->name('admin.brands.store');
        Route::get('/brands/create', [BrandsController::class, 'create'])->name('admin.brands.create');
        Route::get('/brands/{brand}/edit', [BrandsController::class, 'edit'])->name('admin.brands.edit');
        Route::put('/brands/{brand}', [BrandsController::class, 'update'])->name('admin.brands.update');
        Route::delete('/brands/{brand}', [BrandsController::class, 'destroy'])->name('admin.brands.destroy');

        // for bookings
        Route::get('/bookings', [BookingsController::class, 'index'])->name('admin.bookings.index');
        Route::get('/bookings/{booking}/edit', [BookingsController::class, 'edit'])->name('admin.bookings.edit');
        Route::put('/bookings/{booking}', [BookingsController::class, 'update'])->name('admin.bookings.update');
        Route::delete('/bookings/{booking}', [BookingsController::class, 'destroy'])->name('admin.bookings.destroy');

        //for categories
        Route::get('/categories', [CategoriesController::class, 'index'])->name('admin.categories.index');
        Route::post('/categories', [CategoriesController::class, 'store'])->name('admin.categories.store');
        Route::get('/categories/create', [CategoriesController::class, 'create'])->name('admin.categories.create');
        Route::get('/categories/{category}/edit', [CategoriesController::class, 'edit'])->name('admin.categories.edit');
        Route::put('/categories/{category}', [CategoriesController::class, 'update'])->name('admin.categories.update');
        Route::delete('/categories/{category}', [CategoriesController::class, 'destroy'])->name('admin.categories.destroy');

        // for body_types
        Route::get('/body_types', [BodyTypesController::class, 'index'])->name('admin.body_types.index');
        Route::post('/body_types', [BodyTypesController::class, 'store'])->name('admin.body_types.store');
        Route::get('/body_types/create', [BodyTypesController::class, 'create'])->name('admin.body_types.create');
        Route::get('/body_types/{body_type}/edit', [BodyTypesController::class, 'edit'])->name('admin.body_types.edit');
        Route::put('/body_types/{body_type}', [BodyTypesController::class, 'update'])->name('admin.body_types.update');
        Route::delete('/body_types/{body_type}', [BodyTypesController::class, 'destroy'])->name('admin.body_types.destroy');

        // for cars images
        Route::get('/car_images/{id}', [CarsImagesController::class, 'images'])->name('admin.cars_images.images');
        Route::get('/cars_images', [CarsImagesController::class, 'index'])->name('admin.cars_images.index');
        Route::get('/cars_images/{car}/create', [CarsImagesController::class, 'create'])->name('admin.cars_images.create');
        Route::post('/cars_images/{car}', [CarsImagesController::class, 'store'])->name('admin.cars_images.store');
        Route::delete('/cars_images/{car}/image/{image}', [CarsImagesController::class, 'destroyImage'])->name('admin.cars_images.destroyImage');
        Route::delete('/cars_images/{car}/destroyAll', [CarsImagesController::class, 'destroyAll'])->name('admin.cars_images.destroyAll');
    });
});

Route::post('/logout', [AuthAdminController::class, 'logout'])->name('admin.logout');
});

Route::get('/reset-password/{token}', function ($token) {
    $email = request()->query('email');
    return redirect("http://localhost:5173/reset-password/{token}?email={$email}");
})->name('password.reset');
```

					ПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

Додаток Б – Фрагменти коду клієнтської частини

У додатку наведено фрагменти коду клієнтської частини застосунку, розробленої з використанням бібліотеки React, що відповідають за інтерфейс користувача, а саме модулі: реєстрації та авторизації, редагування профілю, бронювання автомобіля.

```
const handleSubmit = async (e) => {
  e.preventDefault();
  const isValid = await validateForm();
  const isPhoneValid = await handleValidatePhone();

  if (isValid && isPhoneValid) {
    const { confirmPassword, ...userData } = formData;
    try {
      const response = await axios.post(
        "http://127.0.0.1:8000/api/register",
        {
          ...userData,
          number: phoneNumber,
          password_confirmation: formData.confirmPassword,
        }
      );
      setSuccessMessage("Реєстрація пройшла успішно!");

      setTimeout(() => {
        navigate("/login");
      }, 1000);
    } catch (error) {
      setSuccessMessage("Реєстрація не пройшла :(");
      return false;
    }
  } else {
    return false;
  }
};
```

```
const validateForm = async () => {
  const errors = {};
  const { email, password } = formData;

  if (!email.trim()) {
    errors.email = "Електронна пошта обов'язкова";
  } else if (!/^[a-zA-Z0-9+@-]+\.[a-zA-Z0-9+@-]+\.[a-zA-Z0-9+@-]+$/i.test(email)) {
    errors.email = "Невірний формат електронної пошти";
  }

  if (!password.trim()) {
    errors.password = "Потрібен пароль";
  }

  if (Object.keys(errors).length > 0) {
    setFormErrors(errors);
    return false;
  }

  try {
    const response = await axios.post(
      "http://127.0.0.1:8000/api/check-user",
      {
        email,
        password,
      }
    );

    if (response.data.status === "success") {
      localStorage.setItem("token", response.data.token);
      return true;
    } else {
      errors.form = "Неправильний email або пароль!";
    }
  } catch (error) {
    console.error("Error validating email:", error);
    errors.form = "Неправильний email або пароль!";
  }

  setFormErrors(errors);

  return Object.keys(errors).length === 0;
}
```

Змн.	Арк.	№ докум.	Підпис	Дата

```

const handleSubmit = async (e) => {
  e.preventDefault();
  const errors = {};
  const token = localStorage.getItem("token");
  if(formData.email === currEmail && formData.name.trim() === currName && changedNumber === currPhone) {
    errors.formError = "Нічого змінювати!";
    setFormErrors(errors);
    return;
  }

  const isValid = await validateForm();
  const isPhoneValid = await handleValidatePhone();

  if (isValid && isPhoneValid) {
    const { ...userData } = formData;

    try {
      // eslint-disable-next-line no-unused-vars
      const response = await axios.put(
        "http://127.0.0.1:8000/api/user-update",
        {
          ...userData,
          number: changedNumber ? changedNumber : phoneNumber,
        },
        {
          headers: {
            Authorization: `Bearer ${token}`,
          }
        }
      );
      setSuccessMessage("Профіль успішно оновлено!");

      setTimeout(() => {
        window.location.reload();
      }, 1000);
    } catch (er) {
      errors.formError = "Не вдалося оновити профіль :(";
    }
  } else {
    return false;
  }
};

```

```

const handleSubmit = async (e) => {
  e.preventDefault();

  if (!handleValidate()) {
    return;
  }

  setIsSubmitted(true);
  setIsButtonDisabled(true);

  const formatDate = (date) =>
    date ? format(new Date(date.dateInstance), "yyyy-MM-dd") : "";

  const formData = {
    start_date: formatDate(dates.startDate),
    end_date: formatDate(dates.endDate),
    number: phoneNumber,
    from_time: time,
    end_time: time,
    car_id: data.id,
  };

  try {
    const response = await axios.post(
      "http://127.0.0.1:8000/api/bookings",
      formData
    );

    if (response.data.error) {
      setBookingErr(true);
      return false;
    }

    setSuccess(true);
    setIsButtonDisabled(true);
  } catch (error) {
    console.error(error);
  }
};

```

Змн.	Арк.	№ докум.	Підпис	Дата

ПЗ.КР.Б – 121 – 25 – ПЗ

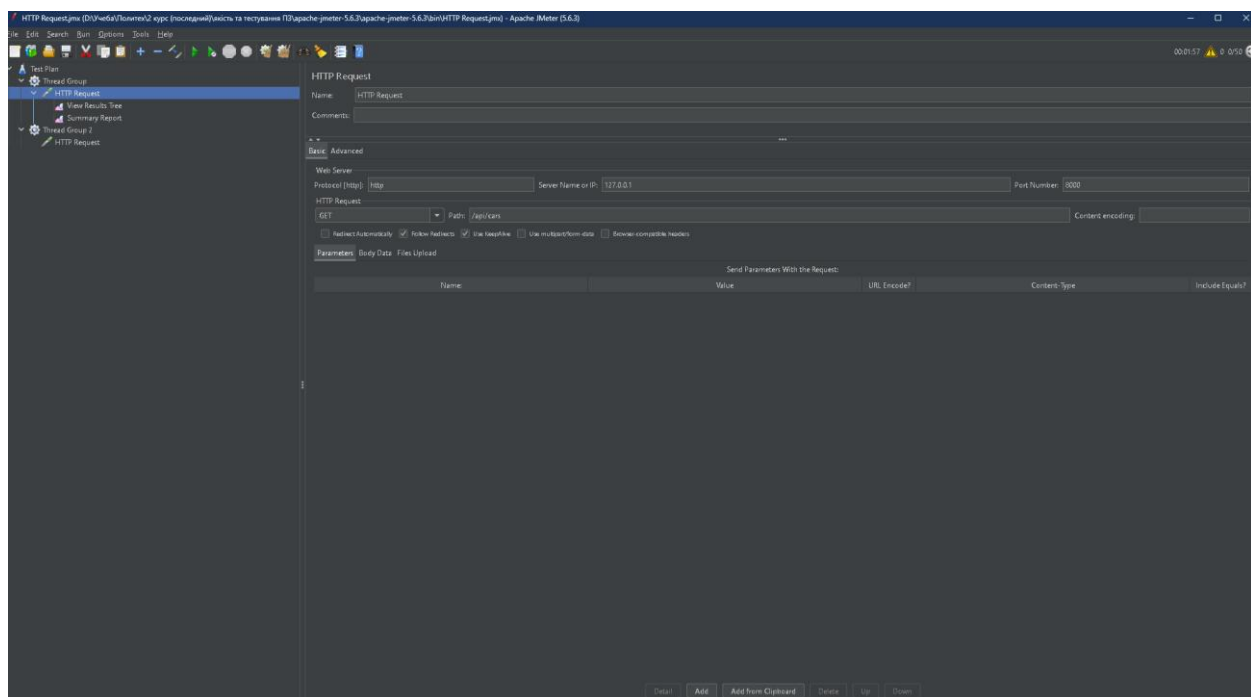
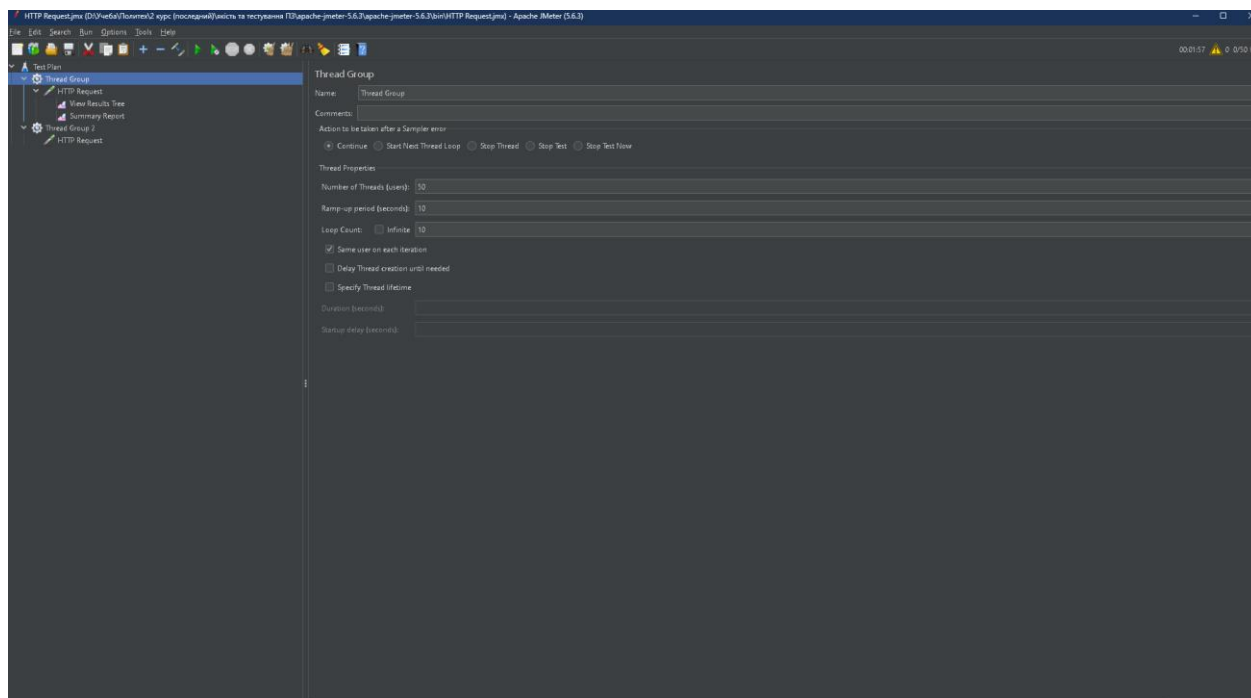
Арк.

80

Додаток В – Налаштування тестування продуктивності в JMeter

У цьому додатку описано налаштування та результати тестів в JMeter з двома групами віртуальних користувачів, для оцінки продуктивності сайту.

Threads Group 1:



Змн.	Арк.	№ докум.	Підпис	Дата

HTTP Request.png [D:\web\2\server\2\app\localhost\apache-jmeter-3.6.2\bin\HTTP_Request.png] - Apache JMeter (3.6.2)

FileEditSearchRunOptionsTestHelp

Test PlanThread GroupHTTP RequestView Results TreeSummary ReportThread Group 2HTTP Request

00:01:570 27%

Summary Report

Name:Summary Report

Comments:

Write results to file / Read from file

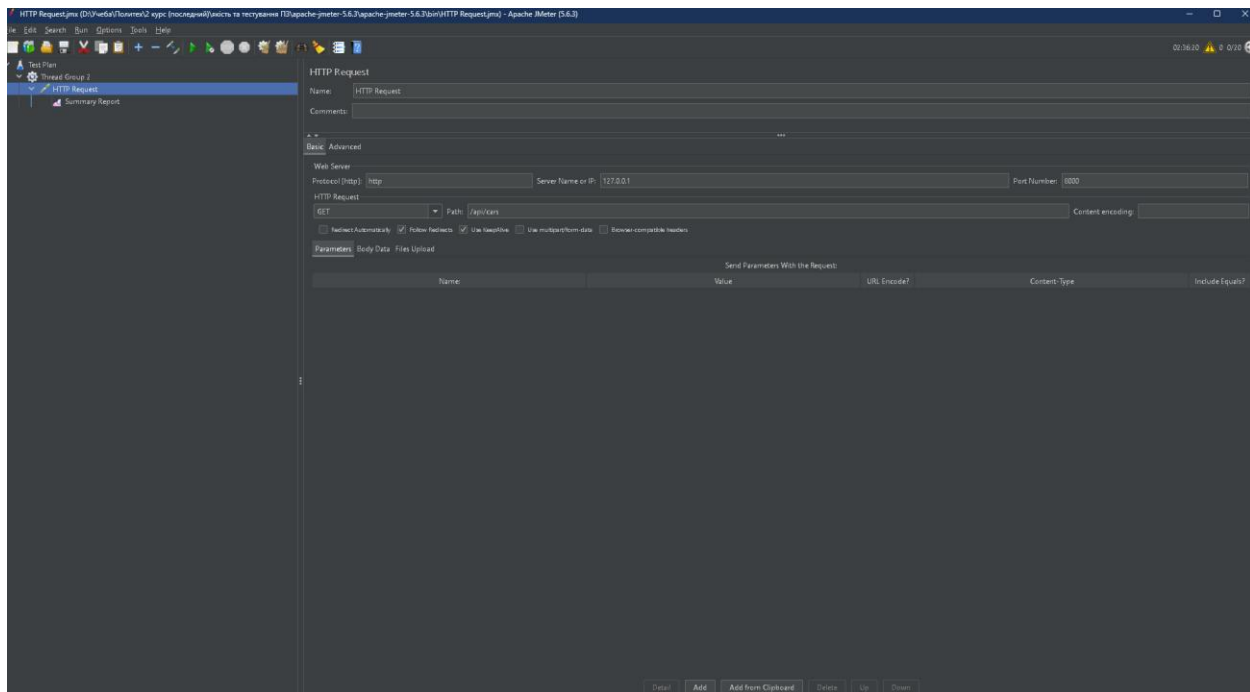
Filename

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	500	10104	363	11801	2618.66	0.00%	4.3/sec	46.22	0.53
TOTAL	500	10104	363	11801	2618.66	0.00%	4.3/sec	46.22	0.53

☐ Include group name in label?

Threads Group 2:

Thread Group										
Name: Thread Group 2										
Comments:										
Action to be taken after a sampler error										
☒ Continue ☐ Start Next Thread Loop ☐ Stop Thread ☐ Stop Test ☐ Stop Test Now										
Thread Properties										
Number of Threads (each): 10										
Ramp-up period (seconds): 30										
Loop Count: ☒ Infinite										
☒ Same user on each iteration										
☐ Delay Thread creation until needed										
☐ Specify Thread lifetime										
Duration (seconds):										
Startup delay (seconds):										



Summary Report Table:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received Kbytes/sec	Sent Kbytes/sec	Avg. Bytes
HTTP Request	10000	4015	100	6191	266.79	0.01%	4.1/sec	44.09	0.51	11096
TOTAL	10000	4015	100	6191	266.79	0.01%	4.1/sec	44.09	0.51	11096