Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Інститут комп'ютерних наук та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри, циклової комісії)

Пояснювальна записка

до дипломної роботи

перший (бакалаврський)
(рівень вищої освіти)

на тему:«Розроблення інформаційної вебсистеми для оренди автомобілів»

Виконав:студент 4 курсу групи КН-41
спеціальності 122 "Комп'ютерні науки"
(шифр і назва спеціальності)

Лутчин М.А
(прізвище та ініціали)

Керівник Мокрицька О.В
(прізвище та ініціали)

Рецензент Флуд Л.О.
(прізвище та ініціали)

Львів – 2025

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Інститут комп'ютерних наук та інформаційних технологій
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КН

Борецька І.Б
"10" червня 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Лутчин Максим Андрійович
(прізвище, ім'я по батькові)

1. Тема роботи Розроблення інформаційної вебсистеми для оренди автомобілів/Development of an information web system for car rental керівник роботи Мокрицька О.В Доцент кафедри інженерії програмного забезпечення, кандидат технічних наук,
(прізвище, ім'я по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "15" листопада 2024 року №С-882
2. Термін подання студентом роботи 10 червня 2025 р.
3. Вихідні дані до роботи Вихідними даними до роботи є вимоги до функціональності системи оренди автомобілів, включаючи управління користувачами, автопарком, бронюваннями та відгуками, а також технічні вимоги до реалізації веб-додатку з використанням PHP, MariaDB та сучасних веб-технологій.
4. Зміст пояснювальної записки (перелік пиатнь, які потрібно розробити)
ВСТУП
РОЗДІЛ 1. Стан проблемної області
РОЗДІЛ 2. Інформаційне та математичне забезпечення
РОЗДІЛ 3. Програмне та технічне забезпечення
ВИСНОВКИ
5. перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Підготовка матеріалу до доповіді

6. Дата видачі завдання 18 листопада 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Збір та опрацювання матеріалу за темою дипломної роботи	18.11.2024-26.11.2024	Виконано
2	Проєктування програми	26.11.2024-14.12.2024	Виконано
3	Розробка програми	14.12.2024-24.12.2024	Виконано
4	Тестування програми	24.12.2024-20.01.2025	Виконано
5	Внесення правок у програму	20.01.2025-30.01.2025	Виконано
6	Розробка першого розділу пояснювальної записки	30.01.2025-20.02.2025	Виконано
7	Розробка другого розділу пояснювальної записки	20.02.2025-15.02.2025	Виконано
8	Розробка третього розділу пояснювальної записки	15.02.2025-20.03.2025	Виконано
9	Оформлення пояснювальної записки	20.03.2025-10.04.2025	Виконано

Студент _____ (підпис) Лутчин М.А (прізвище та ініціали)

Керівник роботи _____ (підпис) Мокрицька О.В (прізвище та ініціали)

АНОТАЦІЯ

Розглянуто процес створення вебсистеми для оренди автомобілів, який охоплює повний цикл управління автопарком і бронюванням. Основна мета полягає у розробці надійної, захищеної та масштабованої системи, яка включає модуль управління користувачами з диференційованим доступом, каталог автомобілів з розширеною інформацією, систему бронювання з автоматичним підрахунком вартості, а також адміністративну панель для комплексного керування платформою. Інтерфейс користувача реалізовано з використанням HTML5, CSS3, JavaScript і Bootstrap 5, а серверну логіку - за допомогою PHP. Зберігання даних здійснюється у базі MariaDB(MySQL) з оптимізованою структурою, що складається з шести взаємопов’язаних таблиць.

Ключові слова: оренда авто, веб-застосунок, PHP, MariaDB(MySQL), Bootstrap, захист, бронювання, система оцінювання.

ABSTRACT

The process of developing a web-based car rental system is examined, covering the full cycle of fleet management and booking. The main goal is to create a reliable, secure, and scalable system that includes a user management module with differentiated access levels, a car catalog with detailed information, a booking system with automatic cost calculation, and an administrative panel for comprehensive platform management. The user interface is implemented using HTML5, CSS3, JavaScript, and Bootstrap 5, while the server-side logic is built with PHP. Data is stored in a MariaDB (MySQL) database with an optimized structure consisting of six interrelated tables.

Keywords: car rental, web application, PHP, MariaDB (MySQL), Bootstrap, security, booking, evaluation system.

ТЕХНІЧНЕ ЗАВДАННЯ

Створити сучасну, безпечну та зручну у використанні веб-систему для управління процесом оренди автомобілів, що забезпечить ефективну взаємодію між клієнтами та адміністраторами системи.

Система повинна забезпечувати повний цикл оренди автомобілів, включаючи каталог автомобілів з детальною інформацією, систему бронювання, управління користувачами та адміністрування. Інтерфейс системи має бути інтуїтивно зрозумілим, адаптивним для різних пристроїв та підтримувати українську мову.

Функціональні вимоги:

Реєстрація та автентифікація користувачів з різними рівнями доступу (адмін, клієнт)

Каталог автомобілів з можливістю фільтрації та пошуку

Детальна інформація про кожен автомобіль (технічні характеристики, фотографії, ціна)

Система бронювання з автоматичним розрахунком вартості

Управління бронюваннями (створення, редагування, скасування)

Система відгуків та рейтингів

Адміністративна панель для управління контентом

Система повідомлень для зв'язку з адміністрацією

Технічні вимоги:

Мова програмування: PHP

База даних: MariaDB(MySQL)

Frontend: HTML5, CSS3, JavaScript, Bootstrap 5

Безпека: захист від SQL-ін'єкцій, XSS-атак, CSRF

Продуктивність: час завантаження сторінок не більше 2 секунд

Масштабованість: підтримка до 1000 одночасних користувачів

Система має бути розроблена з урахуванням сучасних стандартів веб-розробки, забезпечувати надійне зберігання даних та мати зручний інтерфейс для користувачів. Особливу увагу приділити безпеці системи та захисту персональних даних користувачів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

API Application Programming Interface – інтерфейс програмування додатків

CRUD Create, Read, Update, Delete – базові операції створення, читання, оновлення й видалення даних

DBMS Database Management System – система управління базами даних

ER-діаграма Entity–Relationship Diagram – діаграма «сутність–зв'язок»

GET HTTP GET – метод запиту у протоколі HTTP для отримання ресурсів

HTML HyperText Markup Language – мова розмітки гіпертексту

HTTP HyperText Transfer Protocol – протокол передачі гіпертексту

MVC Model–View–Controller –архітектурний шаблон «модель–подання–контролер»

СУБД MariaDB(MySQL) – реляційна система управління базами даних з відкритим кодом

ORM Object–Relational Mapping – механізм відображення даних між об'єктами програми й реляційною БД

PDO PHP Data Objects – інтерфейс для роботи з базами даних у PHP

PHP Hypertext Preprocessor – серверна мова сценаріїв

POST HTTP POST – метод запиту у протоколі HTTP для відправлення даних на сервер

SQL Structured Query Language – мова структурованих запитів до реляційної бази даних

URI Uniform Resource Identifier – унікальний ідентифікатор ресурсу (адреса веб-сторінки або **API**)

URL Uniform Resource Locator – локатор ресурсу в Інтернеті (специфічний тип **URI**)

XAMPP Cross-Platform Apache MariaDB(MySQL) PHP Perl – локальне середовище для розробки (**веб-сервер** + **СУБД** + **PHP**)

ВСТУП

У сучасних умовах цифровізації економіки та розвитку інформаційних технологій автоматизація бізнес-процесів стає критично важливою для підвищення ефективності та конкурентоспроможності підприємств. Сфера оренди автомобілів не є винятком - перехід від традиційних методів управління автопарком до інформаційних систем дозволяє оптимізувати процеси бронювання, зменшити операційні витрати та покращити якість обслуговування клієнтів.

Існуючі комерційні рішення для автоматизації оренди автомобілів часто є надто складними та дорогими для малих та середніх підприємств, а також можуть містити зайвий функціонал, що не відповідає конкретним потребам бізнесу. Крім того, багато з цих систем не враховують специфіку українського ринку та потреб місцевих компаній. Тому розробка спеціалізованої веб-інформаційної системи для оренди автомобілів з використанням сучасних веб-технологій є актуальним та перспективним напрямком дослідження.

Об'єкт дослідження

Об'єктом дослідження є веб-інформаційна система для автоматизації процесів оренди автомобілів, яка включає в себе серверну частину на основі PHP та MariaDB(MySQL), клієнтський інтерфейс з використанням HTML, CSS та JavaScript, а також адміністративний модуль для управління системою. Система забезпечує повний цикл обслуговування клієнтів: від реєстрації та авторизації користувачів до оформлення бронювань, управління автопарком та обробки відгуків.

Предмет дослідження

Предметом дослідження є методи та технології проектування, розробки та впровадження веб-інформаційних систем для автоматизації бізнес-процесів оренди автомобілів, включаючи моделювання структури даних, реалізацію бізнес-логіки, забезпечення безпеки та створення зручного користувацького інтерфейсу без використання готових фреймворків.

Мета роботи

Розробити та впровадити функціональну, безпечну та масштабовану веб-інформаційну систему для автоматизації процесів оренди автомобілів, яка забезпечить ефективну взаємодію між клієнтами та адміністраторами, автоматизацію основних бізнес-процесів та підвищення якості обслуговування користувачів.

Практичне значення

Розроблена веб-інформаційна система може бути успішно впроваджена малими та середніми підприємствами, що надають послуги оренди автомобілів, забезпечуючи їм конкурентні переваги через автоматизацію процесів та покращення якості обслуговування клієнтів. Система демонструє практичне застосування сучасних веб-технологій та може служити основою для подальшого розширення функціоналу, включаючи інтеграцію з платіжними системами, розробку мобільних додатків або створення REST API для зовнішніх інтеграцій.

Результати роботи можуть бути використані в навчальному процесі для демонстрації принципів проектування веб-систем, роботи з базами даних та реалізації безпечних веб-додатків.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Постановка задачі та загальна характеристика проблеми

У сучасному суспільстві спостерігається стрімкий розвиток цифрових технологій, які активно проникають у різні сфери послуг, зокрема транспортну. Сервіси онлайн-оренди авто дозволяють спростити доступ до транспортних засобів, зробити процес оренди прозорим і зручним, а також мінімізувати людський фактор при обробці замовлень [6].

Класичні підходи до оренди автомобілів зазвичай передбачають безпосередній контакт із менеджером, ручну перевірку доступності авто та паперову документацію. Така модель застаріла й недостатньо ефективна в умовах сучасного ринку. Користувачі очікують можливості швидко знайти авто, переглянути його характеристики, оформити бронювання онлайн, залишити відгук або поставити запитання - усе безпосередньо через браузер.

У зв’язку з цим виникає потреба у розробці спеціалізованих вебдодатків, які дозволять автоматизувати процеси бронювання, адміністрування автопарку та взаємодії з клієнтами [10].

1.2. Аналіз існуючих аналогів

На ринку IT-рішень існують різноманітні системи для оренди автомобілів. Серед найбільш відомих міжнародних прикладів можна виділити такі сервіси, як Rentalcars, Avis, Hertz, Sixt, які мають розвинену архітектуру, підтримку мобільних застосунків, інтеграцію з глобальними базами даних та платіжними шлюзами.

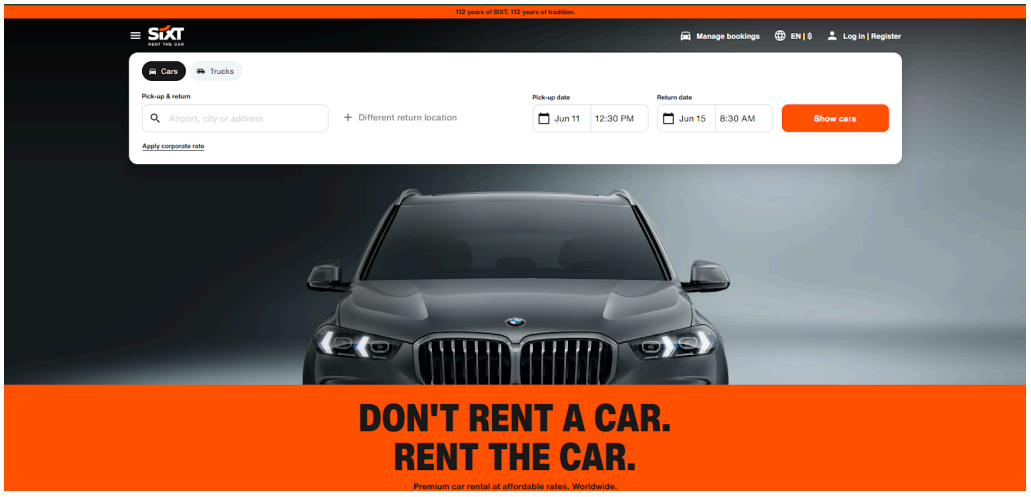


Рисунок 1.1 - Зразок популярного сервісу

Водночас такі сервіси не підходять для невеликих локальних компаній через високу вартість інтеграції, складність налаштування й надмірний функціонал. Українські локальні сервіси, як-от Getmancar або Uklon Rent, хоч і забезпечують базову функціональність, однак не завжди відкриті до адаптації під потреби малого бізнесу чи індивідуальних власників транспорту. Більшість із них не надають доступу до вихідного коду і побудовані на закритих платформах.

Таким чином, є потреба у створенні відкритого, гнучкого і масштабованого вебдодатку для оренди авто, який би дозволив реалізувати повний цикл: від реєстрації користувача до завершення бронювання й отримання відгуку.

1.3. Сильні та слабкі сторони існуючих рішень

На основі аналізу можна зробити висновок, що найбільша прогалина на ринку - це відсутність простих і водночас функціональних рішень для невеликих компаній або фізичних осіб, які хочуть надавати свої транспортні засоби в оренду без посередників. У таблиці 1.1 наведено порівняння основних харектиристик існуючих сервісів які надають свої транспортні засоби в оренду.

Таблиця 1.1

Порівняння основних характеристик деяких існуючих сервісів

Сервіс	**Переваги**	**Недоліки**
Rentalcars	Великий вибір авто, інтеграція API	Висока вартість, складна інтеграція
Getmancar	Зручний мобільний застосунок	Закрита архітектура, неможливість модифікації
Local Rent CMS	Доступ до коду, базова панель адміна	Обмежена функціональність, застарілий інтерфейс

1.4. Обґрунтування вибору технологій

Проєкт реалізується з використанням відкритих технологій:

- PHP - мова серверної логіки, яка має велику підтримку й інтегрується з Apache [11].
- MariaDB(MySQL) - реляційна СУБД, що забезпечує збереження та обробку замовлень, відгуків, автомобілів [9].
- HTML5/CSS3 - для створення адаптивного інтерфейсу [3] .
- XAMPP - середовище для локального розгортання.

Відмова від фреймворків типу Laravel чи Symfony зумовлена бажанням зберегти простоту, мінімальну залежність від сторонніх компонентів і повне розуміння коду. Такий підхід також спрощує навчання для студентів або молодих розробників, які хочуть зрозуміти логіку роботи вебзастосунків "зсередини"[2] .

1.5. Формулювання задачі розробки

Враховуючи вищенаведене, формулюється наступна задача - розробити вебдодаток для онлайн-оренди автомобілів, який має включати:

- модуль реєстрації, авторизації та управління профілем користувача;
- каталог авто з можливістю фільтрації й перегляду характеристик;
- механізм бронювання з обліком дат і статусів;
- систему зворотного зв’язку (відгуки та коментарі);
- адміністративну панель для керування всіма сутностями;
- захист системи (валідація, хешування паролів, захист форм);
- базу даних, яка забезпечить логічні зв’язки між усіма сутностями.

1.6. Очікувані переваги розробки

Розроблений вебдодаток вирішує такі задачі:

- автоматизує процеси, що раніше виконувались вручну;
- підвищує ефективність взаємодії з клієнтами;
- зменшує навантаження на адміністратора автопарку;
- дозволяє зберігати історію бронювань та відгуків;
- може бути розгорнутий на будь-якому локальному сервері безкоштовно.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Виділення об’єктів дослідження

Система оренди автомобілів, що розглядається у даній роботі, є комплексним програмним продуктом, який забезпечує взаємодію між клієнтами, адміністраторами та автопарком. Для ефективного функціонування такої системи необхідно чітко визначити основні об’єкти дослідження, їх властивості, взаємозв’язки та ролі у загальній структурі.

До основних об’єктів дослідження належать:

1. Користувачі системи - це фізичні особи, які взаємодіють із системою. Вони поділяються на дві основні категорії: адміністратори та клієнти. Адміністратори мають розширені права доступу, можуть керувати автопарком, бронюваннями, користувачами, відгуками та повідомленнями. Клієнти – це звичайні користувачі, які мають змогу переглядати доступні автомобілі, здійснювати бронювання, залишати відгуки та коментарі.

2. Автомобілі - це об’єкти, які є предметом оренди. Кожен автомобіль має унікальні характеристики: марка, модель, рік випуску, тип трансмісії, кількість пасажирів, тип пального, наявність кондиціонера, об’єм багажника, кількість дверей, об’єм двигуна, пробіг, а також декілька фотографій для візуального представлення.

3. Бронювання - це процес резервування автомобіля клієнтом на певний період часу. Кожне бронювання містить інформацію про користувача, автомобіль, дати початку та завершення оренди, загальну вартість, статус (очікує підтвердження, підтверджено, відхилено) та унікальний номер тікету.

4. Відгуки та коментарі - це текстові повідомлення, які залишають користувачі після завершення оренди або для загального обговорення. Відгуки містять оцінку (рейтинг) та коментар, а також прив’язуються до конкретного бронювання та автомобіля. Коментарі можуть бути загальними, не прив’язаними до конкретного бронювання.

5. Повідомлення – це звернення користувачів через контактну форму. Вони містять ім’я, email, телефон (необов’язково), текст повідомлення та дату створення.

6. Адміністративні дії – це операції, які виконують адміністратори для підтримки та розвитку системи: додавання, редагування, видалення автомобілів, керування користувачами, обробка бронювань, модерація відгуків та коментарів, обробка повідомлень.

Виділення цих об’єктів дозволяє чітко структурувати інформаційну модель системи, визначити основні сценарії взаємодії та закласти фундамент для подальшого концептуального та фізичного проектування.

2.2. Побудова інформаційної моделі

Інформаційна модель системи оренди автомобілів визначає структуру даних, які зберігаються, обробляються та передаються між різними компонентами системи. Вона є основою для розробки бази даних, програмної логіки та інтерфейсів користувача [12].

2.2.1. Основні сутності та їх атрибути

До основних сутностей користувача (User) належить :

id – унікальний ідентифікатор користувача (ціле число, автоінкремент)

username – унікальний логін користувача (рядок)

password – хешований пароль (рядок)

role – роль користувача (admin/client)

full_name – повне ім’я користувача (рядок)

email – електронна пошта (рядок)

phone – номер телефону (рядок)

created_at – дата та час створення облікового запису (timestamp)

reset_token – токен для відновлення паролю (рядок, може бути null)

reset_expires – термін дії токену (timestamp, може бути null)

	#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
☐	1	id	int(11)			Ні	*Немає*		AUTO_INCREMENT
☐	2	username	varchar(50)	utf8mb4_general_ci		Ні	*Немає*		
☐	3	password	varchar(255)	utf8mb4_general_ci		Ні	*Немає*		
☐	4	role	enum('admin', 'client')	utf8mb4_general_ci		Так	client		
☐	5	full_name	varchar(100)	utf8mb4_general_ci		Так	*NULL*		
☐	6	email	varchar(100)	utf8mb4_general_ci		Так	*NULL*		
☐	7	phone	varchar(20)	utf8mb4_general_ci		Так	*NULL*		
☐	8	created_at	timestamp			Ні	current_timestamp()		
☐	9	reset_token	varchar(255)	utf8mb4_general_ci		Так	*NULL*		
☐	10	reset_expires	varchar(255)	utf8mb4_general_ci		Так	*NULL*		

Рисунок 2.1 - Структура таблиці користувачів

До основних сутностей автомобіля (Car) належить :

car_id – унікальний ідентифікатор автомобіля (ціле число, автоінкремент)

brand – марка автомобіля (рядок)

model – модель автомобіля (рядок)

year – рік випуску (ціле число)

price_per_day – ціна за добу оренди (десяткове число)

transmission – тип трансмісії (manual/automatic)

available – доступність для оренди (булевий тип)

image, image2, image3, image4 – фотографії автомобіля (рядки)

passengers – кількість пасажирів (ціле число)

fuel_consumption – витрати пального (рядок)

fuel_type – тип пального (рядок)

air_conditioning – наявність кондиціонера (булевий тип)

luggage_capacity – об’єм багажника (ціле число)

doors – кількість дверей (ціле число)

engine_volume – об’єм двигуна (десяткове число)

range_km – запас ходу (ціле число)

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	car_id	int(11)			Ні	*Немає*		AUTO_INCREMENT
2	brand	varchar(100)	utf8mb4_general_ci		Ні	*Немає*		
3	model	varchar(100)	utf8mb4_general_ci		Ні	*Немає*		
4	year	int(11)			Ні	*Немає*		
5	price_per_day	decimal(10,2)			Ні	*Немає*		
6	transmission	enum('manual', 'automatic')	utf8mb4_general_ci		Так	manual		
7	available	tinyint(1)			Так	1		
8	image	varchar(255)	utf8mb4_general_ci		Так	*NULL*		
9	passengers	int(11)			Так	*NULL*		
10	fuel_consumption	varchar(10)	utf8mb4_general_ci		Так	*NULL*		
11	fuel_type	varchar(50)	utf8mb4_general_ci		Так	*NULL*		
12	air_conditioning	tinyint(1)			Так	*NULL*		
13	luggage_capacity	int(11)			Так	*NULL*		
14	doors	int(11)			Так	*NULL*		
15	engine_volume	decimal(3,1)			Так	*NULL*		
16	range_km	int(11)			Так	*NULL*		
17	image2	varchar(255)	utf8mb4_general_ci		Так	*NULL*		
18	image3	varchar(255)	utf8mb4_general_ci		Так	*NULL*		
19	image4	varchar(255)	utf8mb4_general_ci		Так	*NULL*		

Рисунок 2.2 - Структура таблиці авто

До основних сутностей бронювання (Booking) належить :

id – унікальний ідентифікатор бронювання (ціле число, автоінкремент)

car_id – ідентифікатор автомобіля (ціле число)

user_id – ідентифікатор користувача (ціле число)

start_date – дата початку оренди (date)

end_date – дата завершення оренди (date)

total_price – загальна вартість оренди (десяткове число)

status – статус бронювання (pending/approved/rejected)

ticket_id – унікальний номер тікету (рядок)

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	id	int(11)			Ні	*Немає*		AUTO_INCREMENT
2	car_id	int(11)			Ні	*Немає*		
3	user_id	int(11)			Ні	*Немає*		
4	start_date	date			Ні	*Немає*		
5	end_date	date			Ні	*Немає*		
6	total_price	decimal(10,2)			Ні	*Немає*		
7	status	enum('pending', 'approved', 'rejected')	utf8mb4_general_ci		Так	pending		
8	ticket_id	varchar(50)	utf8mb4_general_ci		Ні	*Немає*		

Рисунок 2.3 -Структура таблиці бронювань

До основних сутностей відгуку (Review) належить :

id – унікальний ідентифікатор відгуку (ціле число, автоінкремент)

car_id – ідентифікатор автомобіля (ціле число)

user_id – ідентифікатор користувача (ціле число)

rating – оцінка (ціле число від 1 до 5)

comment – текстовий коментар (текст)

created_at – дата створення (timestamp)

booking_id – ідентифікатор бронювання (ціле число)

approved – статус модерації (булевий тип)

	#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
☐	1	**id**	int(11)			Ні	*Немає*		AUTO_INCREMENT
☐	2	**car_id**	int(11)			Ні	*Немає*		
☐	3	**user_id**	int(11)			Ні	*Немає*		
☐	4	**rating**	tinyint(4)			Ні	*Немає*		
☐	5	**comment**	text	utf8mb4_general_ci		Так	*NULL*		
☐	6	**created_at**	timestamp			Ні	current_timestamp()		
☐	7	**booking_id**	int(11)			Ні	*Немає*		
☐	8	**approved**	tinyint(1)			Так	0		

Рисунок 2.4 - Структура таблиці відгуків

До основних сутностей коментар (Comment) належить :

id – унікальний ідентифікатор коментаря (ціле число, автоінкремент)

user_id – ідентифікатор користувача (ціле число)

comment – текст коментаря (текст)

rating – оцінка (ціле число від 1 до 5)

created_at – дата створення (timestamp)

	#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
☐	1	**id**	int(11)			Ні	*Немає*		AUTO_INCREMENT
☐	2	**user_id**	int(11)			Ні	*Немає*		
☐	3	**comment**	text	utf8mb4_general_ci		Ні	*Немає*		
☐	4	**rating**	tinyint(4)			Ні	*Немає*		
☐	5	**created_at**	timestamp			Ні	current_timestamp()		

Рисунок 2.5 - Структура таблиці коментарів

До основних сутностей повідомлення (Message) належить :

id – унікальний ідентифікатор повідомлення (ціле число, автоінкремент)

name – ім’я відправника (рядок)

email – електронна пошта (рядок)

phone – телефон (рядок, може бути null)

message – текст повідомлення (текст)

created_at – дата створення (timestamp)

	#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
☐	1	id	int(11)			Ні	*Немає*		AUTO_INCREMENT
☐	2	name	varchar(255)	utf8mb4_general_ci		Ні	*Немає*		
☐	3	email	varchar(255)	utf8mb4_general_ci		Ні	*Немає*		
☐	4	phone	varchar(20)	utf8mb4_general_ci		Так	*NULL*		
☐	5	message	text	utf8mb4_general_ci		Ні	*Немає*		
☐	6	created_at	timestamp			Ні	current_timestamp()		

Рисунок 2.6 - Структура таблиці повідомлень

2.2.2. Взаємозв’язки між сутностями

У проекті передбачено, що один користувач може мати багато бронювань, відгуків, коментарів та повідомлень. Один автомобіль може бути заброньований багато разів, мати багато відгуків. Одне бронювання пов’язане з одним користувачем та одним автомобілем, а також може мати один відгук. А відгук пов’язаний з конкретним бронюванням, автомобілем та користувачем.

2.3. Опис існуючих обмежень на вхідні та вихідні дані

2.3.1. Вхідні дані

Вхідні дані - це інформація, яку користувачі або адміністратори вводять у систему для виконання певних дій. До основних обмежень на вхідні дані належать: Реєстрація користувача: логін та email мають бути унікальними, пароль - не менше 6 символів, email - у форматі електронної пошти, ім’я - не порожнє.

Бронювання автомобіля: автомобіль має бути доступним для оренди, дати бронювання не повинні перетинатися з іншими бронюваннями цього користувача для того ж автомобіля, дати мають бути коректними (**start_date ≤ end_date**).

Відгуки: рейтинг – ціле число від 1 до 5, коментар – не порожній, відгук можна залишити лише після завершення бронювання, один користувач може залишити лише один відгук на одне бронювання. Коментарі: рейтинг – від 1 до 5, текст – не порожній. Повідомлення: ім’я, email та текст повідомлення – обов’язкові поля.

2.3.2. Вихідні дані

Вихідні дані – це інформація, яку система генерує у відповідь на дії користувачів або адміністраторів: Списки автомобілів: відображаються з урахуванням фільтрів (марка, рік, трансмісія, ціна, кількість пасажирів, тип пального, кондиціонер). Історія бронювань: користувач бачить лише свої бронювання, адміністратор – всі. Відгуки та коментарі: відображаються лише після модерації (для відгуків), містять ім’я користувача, рейтинг, дату. Повідомлення про помилки: у разі некоректного введення даних, дублювання логіна/email, перетину дат бронювання, некоректного рейтингу тощо. Підтвердження успішних дій: повідомлення про успішну реєстрацію, бронювання, додавання відгуку, скасування бронювання тощо.

2.3.3. Обмеження на рівні бази даних

Унікальність: логін та email користувача, ticket_id у бронюваннях.

Цілісність: зовнішні ключі (user_id, car_id) у таблицях бронювань, відгуків, коментарів. Діапазони: рейтинг - від 1 до 5, статуси - лише допустимі значення (pending, approved, rejected).

Автоінкремент: для ідентифікаторів сутностей.

2.4. Концептуальне проектування системи

2.4.1. Побудова концептуальної моделі

Концептуальна модель системи – це абстрактне уявлення про основні сутності, їх властивості та взаємозв’язки. Вона дозволяє виявити причинно-наслідкові зв’язки, визначити структуру даних та основні сценарії взаємодії [7].

2.4.1.1. ER-діаграма (Entity-Relationship Diagram)

ER-діаграма відображає сутності системи та зв’язки між ними. Основні сутності: User, Car, Booking, Review, Comment, Message.

User (1) — (M) Booking

Car (1) — (M) Booking

Booking (1) — (1) Review

User (1) — (M) Review

Car (1) — (M) Review

User (1) — (M) Comment

User (1) — (M) Message

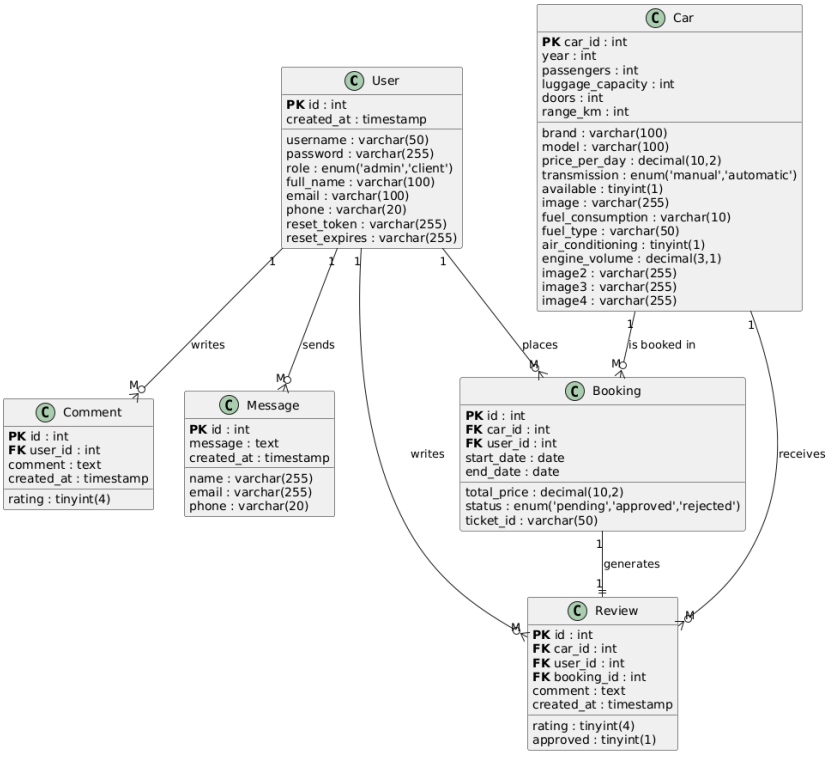


Рисунок 2.7 - ER діаграма зв’язків бази даних

2.4.1.2. UML-діаграма класів

UML-діаграма класів відображає структуру класів, їх атрибути, методи та зв’язки.

User: login(), register(), getUserById(), getUserBookings(), getUserReviews(), cancelBooking(), generatePasswordResetLink(), checkResetToken(), updatePassword()
Car: getAllCars(), getCarById(), bookCar(), getFilteredCars(), getUniqueBrands(), getUniqueYears(), getUniqueTransmissions(), getCarReviews(), addReview(), hasUserReviewedBooking(), updateReview(), deleteReview(), getUniqueFuelTypes(), getUniquePassengers()
Booking: властивості та методи для створення, оновлення, скасування бронювань
Review: властивості та методи для додавання, оновлення, видалення відгуків
Comment: властивості та методи для додавання, видалення коментарів
Message: властивості та методи для обробки повідомлень

Рисунок 2.8 - UML діаграма класів проєкту

2.5. Опис моделей реалізації математичних та алгоритмічних методів

2.5.1. Математичні моделі

2.5.1.1. Модель розрахунку вартості оренди

Вартість оренди автомобіля розраховується за формулою:

$$C = P * D \quad (2.1)$$

де C - загальна вартість оренди, P - ціна за добу, D - кількість днів оренди.

Ця модель дозволяє швидко та однозначно визначити суму, яку має сплатити клієнт за користування автомобілем у вибраний період.

2.5.1.2. Модель перевірки перетину бронювань

Для уникнення дублювання бронювань на одні й ті самі дати для одного користувача та автомобіля використовується наступний підхід:

Нове бронювання можливе, якщо для вибраного автомобіля не існує іншого бронювання з датами, що перетинаються з новими.

Формально: для кожного існуючого бронювання перевіряється, чи виконується умова:

new_end_date < existing_start_date або **new_start_date > existing_end_date**

Якщо умова не виконується, бронювання не дозволяється.

2.5.2. Алгоритмічні моделі

2.5.2.1. Алгоритм реєстрації користувача

1. Користувач вводить логін, пароль, email, ім’я, телефон.
2. Система перевіряє унікальність логіна та email.
3. Якщо дані унікальні, пароль хешується.
4. Дані користувача зберігаються у базі.
5. Користувач отримує підтвердження про успішну реєстрацію.

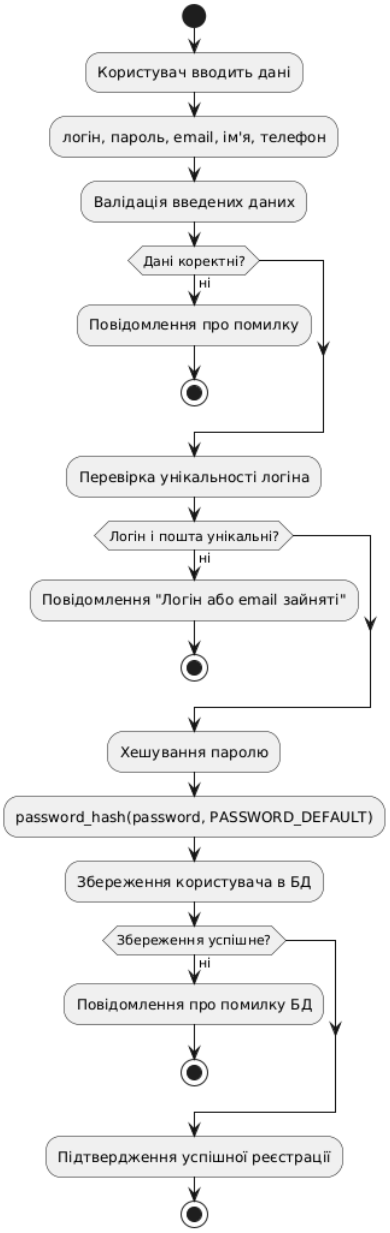


Рисунок 2.9 - Блок схема реєстрації

2.5.2.2. Алгоритм авторизації користувача

1. Користувач вводить логін та пароль.
2. Система знаходить користувача за логіном.
3. Перевіряється відповідність паролю (password_verify).
4. У разі успіху користувач авторизується, інакше - повідомлення про помилку.

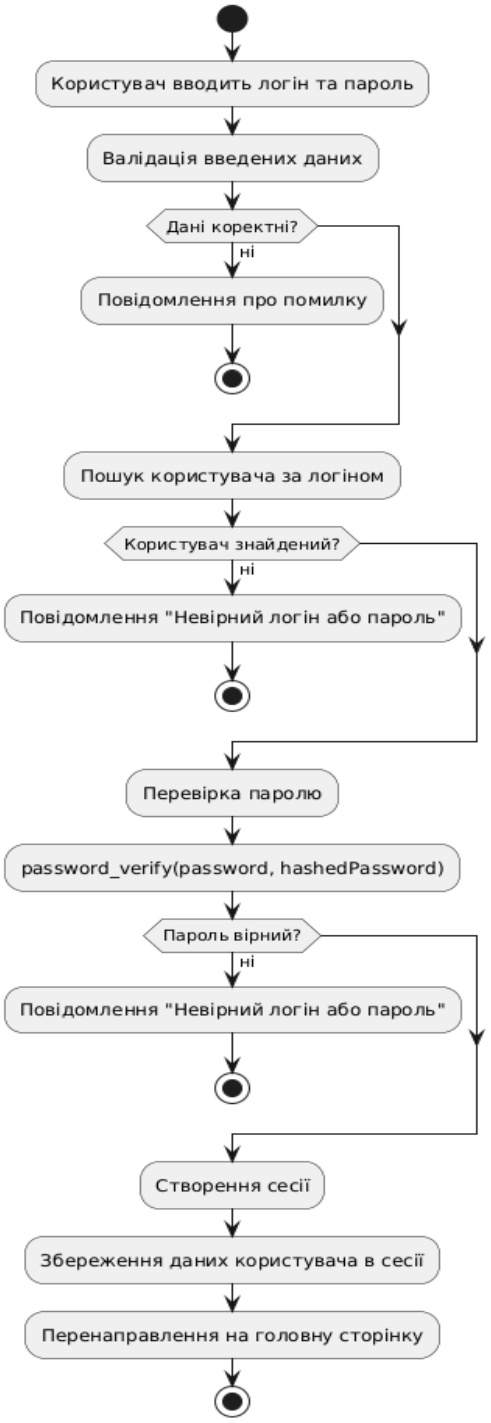


Рисунок 2.10 - Блок схема авторизації

2.5.2.3. Алгоритм бронювання автомобіля

1. Користувач вибирає автомобіль, дати оренди.

2. Система перевіряє доступність автомобіля.

3. Перевіряється відсутність перетину дат з іншими бронюваннями цього користувача для цього авто.

4. Якщо всі умови виконані, створюється новий запис у таблиці бронювань зі статусом "pending" та унікальним ticket_id.

5. Користувач отримує підтвердження про успішне бронювання.

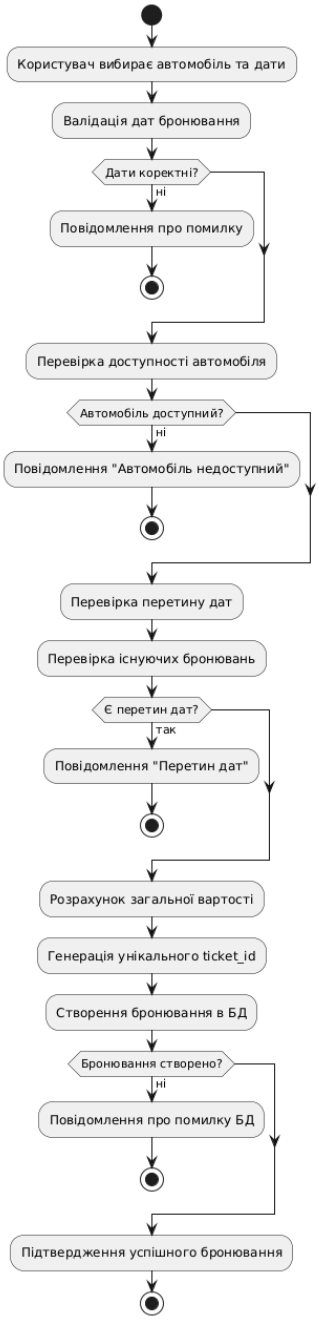


Рисунок 2.11 -Блок схема бронювання

2.5.2.4. Алгоритм додавання відгуку

1. Користувач після завершення бронювання переходить до сторінки відгуків.

2. Система перевіряє, чи користувач має завершене бронювання для цього автомобіля.

3. Перевіряється, чи вже залишав користувач відгук для цього бронювання.

4. Якщо ні, користувач вводить рейтинг та коментар.

5. Відгук зберігається у базі з прив’язкою до бронювання, автомобіля та користувача.

Рисунок 2.12 - Блок схема відгуків

2.5.2.5. Алгоритм скасування бронювання

1. Користувач вибирає бронювання зі статусом "pending".

2. Система перевіряє, чи належить бронювання користувачу.

3. Якщо так, статус бронювання змінюється на "rejected".

4. Користувач отримує підтвердження про скасування.

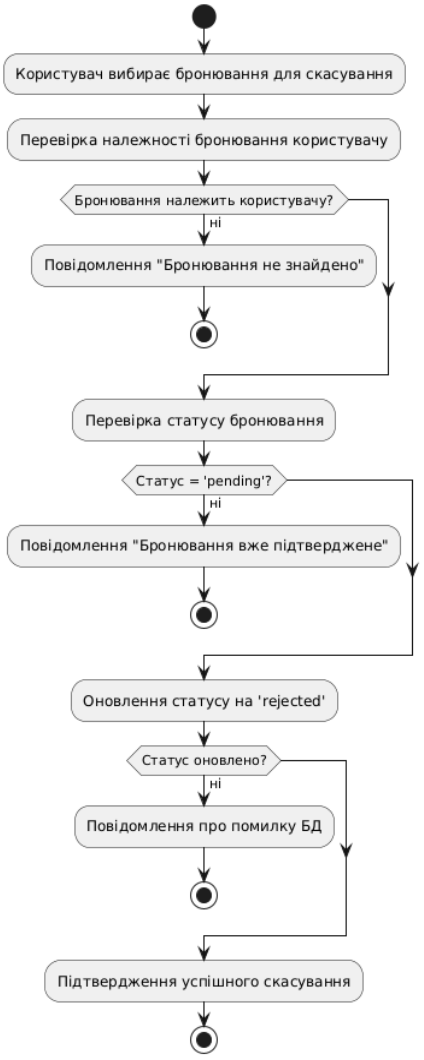


Рисунок 2.13 - Блок схема скасування бронювання

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Огляд, обґрунтування та вибір інструментарію для програмної реалізації

3.1.1. Огляд інструментарію

Для розробки інформаційної системи оренди автомобілів було проведено аналіз сучасних інструментів та технологій, які забезпечують ефективну реалізацію веб-додатків. До основних інструментів, що розглядались, належать:

Серверні технології:

- PHP (Hypertext Preprocessor) - скриптова мова програмування, яка дозволяє динамічно генерувати HTML - сторінки, обробляти форми, взаємодіяти з базами даних та реалізовувати бізнес-логіку [11].
- XAMPP - інтегрований пакет, що включає Apache (веб-сервер), MariaDB(MySQL)/MariaDB(MySQL) (СУБД), PHP та Perl. XAMPP забезпечує локальне середовище для розробки, тестування та налагодження веб-додатків без необхідності встановлення окремих компонентів.
- MariaDB(MySQL)/MariaDB(MySQL) - реляційна система управління базами даних, яка забезпечує надійне зберігання, обробку та запити до даних. MariaDB(MySQL) є однією з найпопулярніших СУБД для веб-додатків завдяки своїй швидкодії, масштабованості та простоті використання.

Клієнтські технології:

- HTML (HyperText Markup Language) - мова розмітки, що використовується для структурування контенту веб-сторінок [3].
- CSS (Cascading Style Sheets) - мова стилів, яка дозволяє форматувати та стилізувати HTML - елементи, забезпечуючи привабливий та адаптивний інтерфейс [3].
- JavaScript - скриптова мова програмування, що використовується для додавання інтерактивності, валідації форм, асинхронних запитів (AJAX) та покращення користувацького досвіду.

Інструменти розробки:

- IDE (Integrated Development Environment) - наприклад, PhpStorm, Visual Studio Code, Sublime Text, які забезпечують зручне редагування коду, підсвічування синтаксису, автодоповнення, налагодження та інтеграцію з системами контролю версій (наприклад, Git).
- Системи контролю версій (Git, GitHub, GitLab) - для відстеження змін у коді, спільної роботи команди, резервного копіювання та управління релізами.

3.1.2. Обґрунтування вибору інструментарію

Вибір інструментарію для реалізації системи оренди автомобілів базувався на наступних критеріях:

Надійність та стабільність:

- PHP, Apache, MariaDB(MySQL) є перевіреними, широко використовуваними технологіями з великою спільнотою розробників, що забезпечує надійність, безпеку та оперативне вирішення проблем.
- XAMPP, як інтегрований пакет, спрощує налаштування локального середовища, що дозволяє швидко розпочати розробку без додаткових конфігурацій.

Масштабованість та продуктивність:

- MariaDB(MySQL) забезпечує ефективне зберігання та обробку даних, підтримує індекси, транзакції, що критично для систем з великою кількістю запитів.
- PHP, завдяки своїй простоті та гнучкості, дозволяє швидко розробляти та впроваджувати нові функції, а також інтегруватися з різними бібліотеками та фреймворками.

Крос-платформність:

XAMPP, PHP, MariaDB(MySQL), HTML, CSS, JavaScript є крос-платформними, що дозволяє розробляти та тестувати додаток на різних операційних системах (Windows, macOS, Linux) без додаткових змін у коді.

Безпека:

- PHP надає вбудовані механізми для захисту від SQL-ін’єкцій (PDO, prepared statements), XSS (фільтрація вхідних даних), CSRF (токени), а також підтримує хешування паролів (password_hash, password_verify).
- MariaDB(MySQL) дозволяє налаштовувати права доступу, шифрування даних, що забезпечує захист конфіденційної інформації.

Спільнота та документація:

Велика кількість навчальних матеріалів, форумів, бібліотек (наприклад, Composer для PHP) спрощує навчання, розробку та підтримку проекту.

3.1.3. Вибір інструментарію

На основі проведеного аналізу та обґрунтування для реалізації системи оренди автомобілів було обрано наступний інструментарій:

Серверна частина:

- PHP (версія 8.2 або вище) – для реалізації бізнес-логіки, обробки форм, взаємодії з базою даних.
- XAMPP (включає Apache, MariaDB(MySQL), PHP) – для локального розгортання, тестування та налагодження.
- MariaDB(MySQL) (версія 10.4 або вище) – для зберігання та управління даними.
- Клієнтська частина:
- HTML5 – для структурування контенту.
- CSS3 – для стилізації та адаптивного дизайну.
- JavaScript (ES6+) – для інтерактивності, валідації форм, асинхронних запитів (AJAX).

Інструменти розробки:

- IDE: PhpStorm або Visual Studio Code – для зручного редагування коду, налагодження, інтеграції з Git.
- Система контролю версій: Git (GitHub/GitLab) – для відстеження змін, спільної роботи, резервного копіювання.

3.2. Опис апаратних засобів

Для розробки, тестування та експлуатації системи оренди автомобілів необхідно забезпечити відповідні апаратні ресурси. Нижче наведено рекомендовані мінімальні вимоги до апаратного забезпечення:

3.2.1. Сервер (для розгортання на хостингу або локально)

Процесор (CPU):

- Мінімум: 2 ядра, 2 ГГц (наприклад, Intel Core i3, AMD Ryzen 3).
- Рекомендовано: 4 ядра, 3 ГГц або вище (Intel Core i5, AMD Ryzen 5) для забезпечення швидкої обробки запитів та підтримки одночасної роботи багатьох користувачів.

Оперативна пам’ять (RAM):

- Мінімум: 4 ГБ.
- Рекомендовано: 8 ГБ або більше для забезпечення стабільної роботи веб-сервера (Apache), СУБД (MariaDB(MySQL)) та PHP-скриптів, особливо при високому навантаженні.

Жорсткий диск (HDD/SSD):

- Мінімум: 20 ГБ вільного місця.
- Рекомендовано: SSD (Solid State Drive) об’ємом 50 ГБ або більше для швидкого доступу до даних, зберігання файлів (фотографій автомобілів, логів, резервних копій) та забезпечення високої продуктивності.

Мережеве з’єднання:

- Швидкість: мінімум 10 Мбіт/с (для локального тестування), рекомендовано 100 Мбіт/с або вище для хостингу.
- Стабільність: низька затримка (latency) для забезпечення швидкого відгуку системи.

3.2.2. Робоча станція розробника (для локальної розробки)

Процесор (CPU):

- Мінімум: 2 ядра, 2 ГГц.

- Рекомендовано: 4 ядра, 3 ГГц або вище для комфортної роботи з IDE, компіляції, налагодження.

Оперативна пам’ять (RAM):

- Мінімум: 4 ГБ.
- Рекомендовано: 8 ГБ або більше для одночасної роботи з IDE, веб-сервером (XAMPP), СУБД, браузером та іншими інструментами.

Жорсткий диск (HDD/SSD):

- Мінімум: 20 ГБ вільного місця.
- Рекомендовано: SSD об’ємом 50 ГБ або більше для швидкого доступу до файлів проекту, зберігання резервних копій, віртуальних машин (якщо використовуються).

Монітор:

- Роздільна здатність: мінімум 1920x1080 (Full HD).
- Рекомендовано: 2K або 4K для зручності роботи з кодом, діаграмами, інтерфейсами.
- Клавіатура, миша:
- Зручні для тривалої роботи, з підтримкою додаткових функцій (наприклад, програмовані клавіші).

Операційна система:

Windows 10/11, macOS, Linux (Ubuntu, Fedora) – залежно від особистих уподобань та сумісності з обраними інструментами (XAMPP, IDE, Git).

3.3. Вимоги до технічного та програмного забезпечення

3.3.1. Вимоги до технічного забезпечення

Сервер (хостинг або локальний):

- Процесор: мінімум 2 ядра, 2 ГГц; рекомендовано 4 ядра, 3 ГГц або вище.
- Оперативна пам’ять: мінімум 4 ГБ; рекомендовано 8 ГБ або більше.
- Жорсткий диск: мінімум 20 ГБ вільного місця; рекомендовано SSD 50 ГБ або більше.

- Мережеве з’єднання: мінімум 10 Мбіт/с; рекомендовано 100 Мбіт/с або вище, низька затримка.

Робоча станція розробника:

- Процесор: мінімум 2 ядра, 2 ГГц; рекомендовано 4 ядра, 3 ГГц або вище.
- Оперативна пам’ять: мінімум 4 ГБ; рекомендовано 8 ГБ або більше.
- Жорсткий диск: мінімум 20 ГБ вільного місця; рекомендовано SSD 50 ГБ або більше.
- Монітор: мінімум 1920x1080 (Full HD); рекомендовано 2K або 4K.
- Клавіатура, миша: зручні для тривалої роботи.
- Операційна система: Windows 10/11, macOS, Linux (Ubuntu, Fedora).

3.3. Вимоги до технічного та програмного забезпечення

3.3.1. Вимоги до технічного забезпечення

Сервер (хостинг або локальний):

- Процесор: мінімум 2 ядра, 2 ГГц; рекомендовано 4 ядра, 3 ГГц або вище.
- Оперативна пам’ять: мінімум 4 ГБ; рекомендовано 8 ГБ або більше.
- Жорсткий диск: мінімум 20 ГБ вільного місця; рекомендовано SSD 50 ГБ або більше.
- Мережеве з’єднання: мінімум 10 Мбіт/с; рекомендовано 100 Мбіт/с або вище, низька затримка.
- Робоча станція розробника:
- Процесор: мінімум 2 ядра, 2 ГГц; рекомендовано 4 ядра, 3 ГГц або вище.
- Оперативна пам’ять: мінімум 4 ГБ; рекомендовано 8 ГБ або більше.
- Жорсткий диск: мінімум 20 ГБ вільного місця; рекомендовано SSD 50 ГБ або більше.
- Монітор: мінімум 1920x1080 (Full HD); рекомендовано 2K або 4K.
- Клавіатура, миша: зручні для тривалої роботи.
- Операційна система: Windows 10/11, macOS, Linux (Ubuntu, Fedora).

3.3.2. Вимоги до програмного забезпечення

Серверна частина:

- XAMPP (включає Apache, MariaDB(MySQL), PHP) – для локального розгортання, тестування.
- PHP (версія 8.2 або вище) – для реалізації бізнес-логіки.
- MariaDB(MySQL) (версія 10.4 або вище) – для зберігання та управління даними.
- Додаткові модулі PHP: PDO, MariaDB(MySQL)i, mbstring, gd (для роботи з зображеннями), json, session.

Клієнтська частина:

- Браузер: сучасні версії Chrome, Firefox, Safari, Edge з підтримкою HTML5, CSS3, JavaScript (ES6+).
- Додаткові бібліотеки/фреймворки (за потреби):
- CSS: Bootstrap, Tailwind CSS – для адаптивного дизайну.
- JavaScript: jQuery, Vue.js, React – для інтерактивності, асинхронних запитів (AJAX), валідації форм.

Інструменти розробки:

- IDE: PhpStorm, Visual Studio Code, Sublime Text – для редагування коду, налагодження, інтеграції з Git.
- Система контролю версій: Git (GitHub, GitLab) – для відстеження змін, спільної роботи, резервного копіювання.
- Додаткові інструменти:
- Composer – менеджер залежностей для PHP.
- npm/yarn – менеджери пакетів для JavaScript (якщо використовуються клієнтські бібліотеки).
- Gulp, Webpack – для автоматизації збірки, мініфікації CSS/JS, оптимізації зображень.

3.4. Опис розробленого програмного забезпечення

3.4.1. Загальна структура проекту

Розроблена система оренди автомобілів має модульну структуру, що дозволяє розділити функціональність на логічні блоки, спростити підтримку та масштабування [5]. Нижче наведено основні модулі та їх призначення:

Рисунок 3.1 - Опис структури проєкту

Модуль авторизації та реєстрації:

Файли: `login.php`, `register.php`, `forgot_password.php`, `reset_password.php`, `logout.php`.

Функціональність: авторизація користувачів, реєстрація нових користувачів, відновлення паролю, вихід із системи.

Модуль керування автомобілями:

Файли: `cars.php`, `car_details.php`, `admin/manage_cars.php`, `admin/add_car.php`, `admin/edit_car.php`, `admin/delete_car.php`.

Функціональність: перегляд списку автомобілів, фільтрація, детальна інформація про автомобіль, додавання, редагування, видалення автомобілів (для адміністраторів).

Модуль бронювань:

Файли: `car_details.php` (частина, що відповідає за бронювання), `admin/manage_bookings.php`,`admin/update_booking_status.php`,`cancel_booking.php`.

Функціональність: створення бронювань, перегляд історії бронювань, підтвердження/відхилення бронювань (для адміністраторів), скасування бронювань.

Модуль відгуків та коментарів:
Файли: `add_review.php`, `edit_review.php`, `delete_review.php`, `comments.php`, `add_comment.php`,`delete_comment.php`,`admin/manage_reviews.php`,`admin/review_action.php`, `admin/comment_action.php`.
Функціональність: додавання, редагування, видалення відгуків та коментарів, модерація (для адміністраторів).

Модуль контактної форми:
Файли: `contact.php`, `admin/manage_messages.php`.
Функціональність: відправка повідомлень, перегляд та обробка повідомлень (для адміністраторів).

Модуль адміністративного інтерфейсу:
Файли: `admin/dashboard.php`, `admin/manage_users.php`, `admin_choice.php`.
Функціональність: перегляд статистики, керування користувачами, перехід між різними адміністративними функціями.

Модуль моделей (бізнес-логіка):
Файли: `models/Car.php`, `models/User.php`.
Функціональність: реалізація основних алгоритмів, взаємодія з базою даних, валідація даних, обчислення (наприклад, розрахунок вартості оренди, перевірка перетину дат бронювань).

Модуль конфігурації:
Файли: `config/database.php`, `config/header.php`, `config/footer.php`.
Функціональність: налаштування підключення до бази даних, загальні шаблони (header, footer) для уніфікації інтерфейсу.

Модуль статичних ресурсів:

Директорія: `assets/` (піддиректорії `css/`, `images/`).

Функціональність: зберігання стилів (CSS), зображень, інших статичних файлів.

3.4.2. Опис основних алгоритмів та функцій

3.4.2.1. Алгоритм авторизації користувача

Файл: `login.php`

Функціональність:

Користувач вводить логін та пароль.

Система перевіряє наявність користувача у базі даних.

Якщо користувач існує та пароль вірний (використовується `password_verify`), створюється сесія, користувач перенаправляється на головну сторінку.

У разі помилки виводиться повідомлення про некоректні дані.

Реалізація в коді:

```
<?php
session_start();
require_once 'config/database.php';
require_once 'models/User.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $username = trim($_POST['username']);
    $password = $_POST['password'];

    if (empty($username) || empty($password)) {
        $error = "Будь ласка, заповніть всі поля";
    } else {
        $user = new User($pdo);
        $userData = $user->login($username, $password);

        if ($userData) {
            $_SESSION['user_id'] = $userData['id'];
            $_SESSION['username'] = $userData['username'];
            $_SESSION['role'] = $userData['role'];
```

```
            header("Location: index.php");
            exit();
        } else {
            $error = "Невірний логін або пароль";
        }
    }
}
?>
```

Метод login в класі User:

```
public function login($username, $password) {
    $stmt = $this->pdo->prepare("SELECT * FROM users WHERE username = ?");
    $stmt->execute([$username]);
    $user = $stmt->fetch();

    if ($user && password_verify($password, $user['password'])) {
        return $user;
    }
    return false;
}
```

3.4.2.2. Алгоритм реєстрації користувача

Файл: `register.php`

Функціональність:

Користувач вводить логін, пароль, email, ім’я, телефон.

Система перевіряє унікальність логіна та email.

Якщо дані унікальні, пароль хешується (`password_hash`), новий користувач додається до бази даних.

Користувач отримує підтвердження про успішну реєстрацію.

Реалізація в коді:

```
<?php
require_once 'config/database.php';
require_once 'models/User.php';
```

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $username = trim($_POST['username']);
    $password = $_POST['password'];
    $email = trim($_POST['email']);
    $phone = trim($_POST['phone']);
    $full_name = trim($_POST['full_name']);

    $user = new User($pdo);

    if ($user->register($username, $password, $email, $phone, $full_name)) {
        $success = "Реєстрація успішна! Тепер ви можете увійти.";
    } else {
        $error = "Користувач з таким логіном або email вже існує";
    }
}
?>
```

Метод register в класі user:

```
public function register($username, $password, $email, $phone, $full_name) {
     $stmt = $this->pdo->prepare("SELECT id FROM users WHERE username = :username OR
email = :email");
    $stmt->execute(['username' => $username, 'email' => $email]);
    if ($stmt->fetch()) {
        return false;
    }

    $hashedPassword = password_hash($password, PASSWORD_DEFAULT);

     $stmt = $this->pdo->prepare("INSERT INTO users (full_name, username, password,
email, phone, role) VALUES (:full_name, :username, :password, :email, :phone,
'client')");
    $stmt->execute([
        'full_name' => $full_name,
        'username' => $username,
        'password' => $hashedPassword,
        'email' => $email,
        'phone' => $phone
    ]);

    return true;
```

```
}
```

3.4.2.3. Алгоритм бронювання автомобіля

Файл: `car_details.php` (частина, що відповідає за бронювання)

Функціональність:

Користувач вибирає автомобіль, вказує дати оренди.

Система перевіряє доступність автомобіля, перетинаються чи дати з іншими бронюваннями.

Якщо всі умови виконані, створюється новий запис у таблиці `bookings` зі статусом "pending" та унікальним `ticket_id`.

Користувач отримує підтвердження про успішне бронювання.

Реалізація в коді:

```
<?php
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_SESSION['user_id'])) {
    $car_id = $_POST['car_id'];
    $start_date = $_POST['start_date'];
    $end_date = $_POST['end_date'];
    $user_id = $_SESSION['user_id'];

    $car = new Car($pdo);
    $carData = $car->getCarById($car_id);

    if ($carData) {
        $days = (strtotime($end_date) - strtotime($start_date)) / (60 * 60 * 24);
        $total_price = $carData['price_per_day'] * $days;
        $ticket_id = 'TICKET-' . date('Ymd') . '-' . strtoupper(substr(md5(uniqid()),
0, 6));

                $result = $car->bookCar($user_id, $car_id, $start_date, $end_date,
$total_price, $ticket_id);

        if ($result['success']) {
```

```
            $success = $result['message'];
        } else {
            $error = $result['message'];
        }
    }
}
?>
```

Метод bookCar в класі Car:

```
public function bookCar($user_id, $car_id, $start_date, $end_date, $total_price, $ticket_id) {
    if (!is_numeric($user_id) || !is_numeric($car_id) || !is_numeric($total_price)) {
        return ['success' => false, 'message' => 'Невірні параметри бронювання'];
    }

    try {
        $this->pdo->beginTransaction();

        $stmt = $this->pdo->prepare("
            SELECT COUNT(*) FROM bookings
            WHERE car_id = ?
              AND user_id = ?
              AND status = 'approved'
              AND start_date <= ?
              AND end_date >= ?
        ");
        $stmt->execute([$car_id, $user_id, $end_date, $start_date]);
        $count = $stmt->fetchColumn();

        if ($count > 0) {
            $this->pdo->rollBack();
                  return ['success' => false, 'message' => 'Ви вже забронювали цей автомобіль на вибрані дати'];
        }

        $stmt = $this->pdo->prepare("
            INSERT INTO bookings (car_id, user_id, start_date, end_date, total_price, ticket_id, status)
```

```
                VALUES (:car_id, :user_id, :start_date, :end_date, :total_price,
:ticket_id, 'pending')
        ");

        $result = $stmt->execute([
            ':car_id' => $car_id,
            ':user_id' => $user_id,
            ':start_date' => $start_date,
            ':end_date' => $end_date,
            ':total_price' => $total_price,
            ':ticket_id' => $ticket_id
        ]);

        $this->pdo->commit();
          return ['success' => true, 'message' => 'Бронювання успішно оформлено! Ваш
номер тікету: ' . $ticket_id];
    } catch (Exception $e) {
        $this->pdo->rollBack();
        error_log("Booking error: " . $e->getMessage());
            return ['success' => false, 'message' => 'Помилка при бронюванні: ' .
$e->getMessage()];
    }
}
```

3.4.2.4. Алгоритм додавання відгуку

Файл: `add_review.php`

Функціональність:

Користувач після завершення бронювання переходить до сторінки відгуків.

Система перевіряє, чи користувач має завершене бронювання для цього автомобіля, чи вже залишав відгук.

Якщо ні, користувач вводить рейтинг (1-5) та коментар.

Відгук зберігається у таблиці `reviews` з прив’язкою до бронювання, автомобіля та користувача.

Реалізація в коді:

```
<?php
session_start();
```

```
require_once 'config/database.php';
require_once 'models/Car.php';

if (!isset($_SESSION['user_id'])) {
    header("Location: login.php");
    exit();
}

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $car_id = $_POST['car_id'];
    $booking_id = $_POST['booking_id'];
    $rating = $_POST['rating'];
    $comment = trim($_POST['comment']);
    $user_id = $_SESSION['user_id'];

    $car = new Car($pdo);

    if ($car->hasUserReviewedBooking($booking_id, $user_id)) {
        $error = "Ви вже залишили відгук для цього бронювання";
    } else {
        if ($car->addReview($car_id, $user_id, $booking_id, $rating, $comment)) {
            $success = "Відгук успішно додано!";
        } else {
            $error = "Помилка при додаванні відгуку";
        }
    }
}
?>
```

Метод addReview в класі Car:

```
public function addReview($car_id, $user_id, $booking_id, $rating, $comment) {
    if (!is_numeric($rating) || $rating < 1 || $rating > 5) {
        throw new InvalidArgumentException('Invalid rating value');
    }

    try {
        $stmt = $this->pdo->prepare("
                INSERT INTO reviews (car_id, user_id, booking_id, rating, comment,
created_at)
            VALUES (?, ?, ?, ?, ?, NOW())
        ");
```

```
                return $stmt->execute([$car_id, $user_id, $booking_id, $rating,
htmlspecialchars($comment)]);
    } catch (PDOException $e) {
        error_log("Error adding review: " . $e->getMessage());
        return false;
    }
}

public function hasUserReviewedBooking($booking_id, $user_id) {
    $stmt = $this->pdo->prepare("
        SELECT COUNT(*) as count
        FROM reviews
        WHERE booking_id = ? AND user_id = ?
    ");
    $stmt->execute([$booking_id, $user_id]);
    $result = $stmt->fetch();
    return $result['count'] > 0;
}
```

3.4.2.5. Алгоритм скасування бронювання

Файл: `cancel_booking.php`

Функціональність:

Користувач вибирає бронювання зі статусом "pending".

Система перевіряє, чи належить бронювання користувачу.

Якщо так, статус бронювання змінюється на "rejected".

Користувач отримує підтвердження про скасування.

Реалізація в коді:

```
<?php
session_start();
require_once 'config/database.php';
require_once 'models/User.php';

if (!isset($_SESSION['user_id'])) {
    header("Location: login.php");
    exit();
}
```

```
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_POST['booking_id'])) {
    $booking_id = $_POST['booking_id'];
    $user_id = $_SESSION['user_id'];

    $user = new User($pdo);
    $result = $user->cancelBooking($booking_id, $user_id);

    if ($result['success']) {
        $success = $result['message'];
    } else {
        $error = $result['message'];
    }
}
?>
```

Метод cancelBooking в класі User:

```
public function cancelBooking($booking_id, $user_id) {
    try {
        $stmt = $this->pdo->prepare("
            SELECT status FROM bookings
            WHERE id = ? AND user_id = ?
        ");
        $stmt->execute([$booking_id, $user_id]);
        $booking = $stmt->fetch();

        if (!$booking) {
             return ['success' => false, 'message' => 'Бронювання не знайдено або вам не належить.'];
        }

        if ($booking['status'] !== 'pending') {
            return ['success' => false, 'message' => 'Бронювання вже підтверджене або скасоване.'];
        }

        $stmt = $this->pdo->prepare("
            UPDATE bookings SET status = 'rejected' WHERE id = ?
        ");
        $stmt->execute([$booking_id]);
```

```
            return ['success' => true, 'message' => 'Бронювання успішно скасовано.'];
        } catch (PDOException $e) {
            error_log("Cancel booking error: " . $e->getMessage());
            return ['success' => false, 'message' => 'Помилка при скасуванні.'];
        }
}
```

3.4.2.6. Алгоритм адміністративного керування

Файли:`admin/dashboard.php`,`admin/manage_cars.php`,`admin/manage_bookings.php`, `admin/manage_reviews.php`, `admin/manage_messages.php`, `admin/manage_users.php`

Функціональність:

Адміністратор переглядає статистику, списки автомобілів, бронювань, відгуків, повідомлень, користувачів.

Адміністратор може додавати, редагувати, видаляти автомобілі, підтверджувати/відхиляти бронювання, модерувати відгуки, обробляти повідомлення, керувати користувачами.

Реалізація в коді:

```
<?php
session_start();
require_once '../config/database.php';

if (!isset($_SESSION['user_id']) || $_SESSION['role'] !== 'admin') {
    header("Location: ../login.php");
    exit();
}

// Отримання статистики
$stmt = $pdo->query("SELECT COUNT(*) as total_cars FROM cars");
$total_cars = $stmt->fetch()['total_cars'];

$stmt = $pdo->query("SELECT COUNT(*) as total_bookings FROM bookings");
$total_bookings = $stmt->fetch()['total_bookings'];

$stmt = $pdo->query("SELECT COUNT(*) as total_users FROM users WHERE role = 'client'");
$total_users = $stmt->fetch()['total_users'];
```

```
$stmt = $pdo->query("SELECT COUNT(*) as total_reviews FROM reviews");
$total_reviews = $stmt->fetch()['total_reviews'];

$stmt = $pdo->query("SELECT COUNT(*) as total_messages FROM messages");
$total_messages = $stmt->fetch()['total_messages'];
?>
```

3.4.3. Опис інтерфейсів користувача

Головна сторінка (`index.php`):

Відображає загальну інформацію про систему, посилання на перегляд автомобілів, авторизацію, реєстрацію, контактну форму.

Містить шапку (header) та підвал (footer), які є спільними для всіх сторінок.

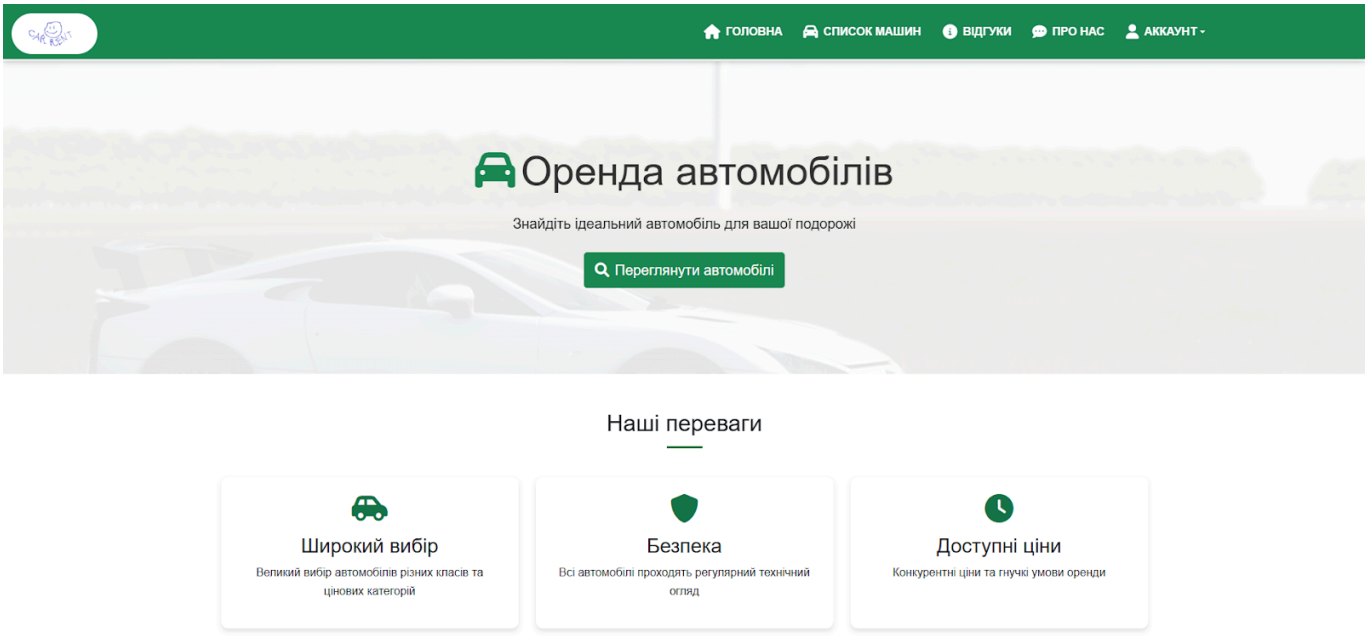


Рисунок 3.2 - Головна сторінка сайту

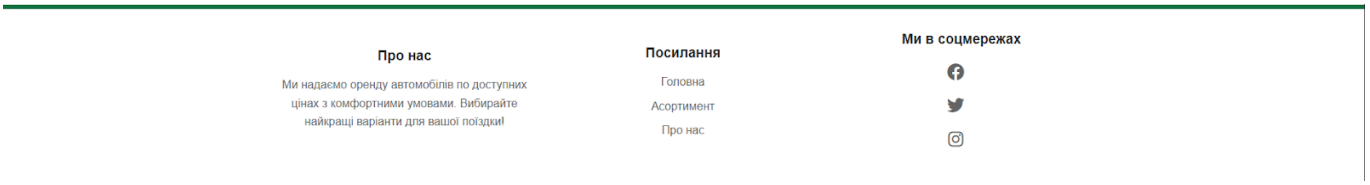


Рисунок 3.3 - Підвал або footer сайту

Сторінка перегляду автомобілів (`cars.php`):

Відображає список доступних автомобілів з можливістю фільтрації за маркою, роком, трансмісією, ціною, кількістю пасажирів, типом пального, кондиціонером.

Кожен автомобіль представлений карткою з фотографією, маркою, моделлю, роком, ціною за добу, посиланням на детальну інформацію.

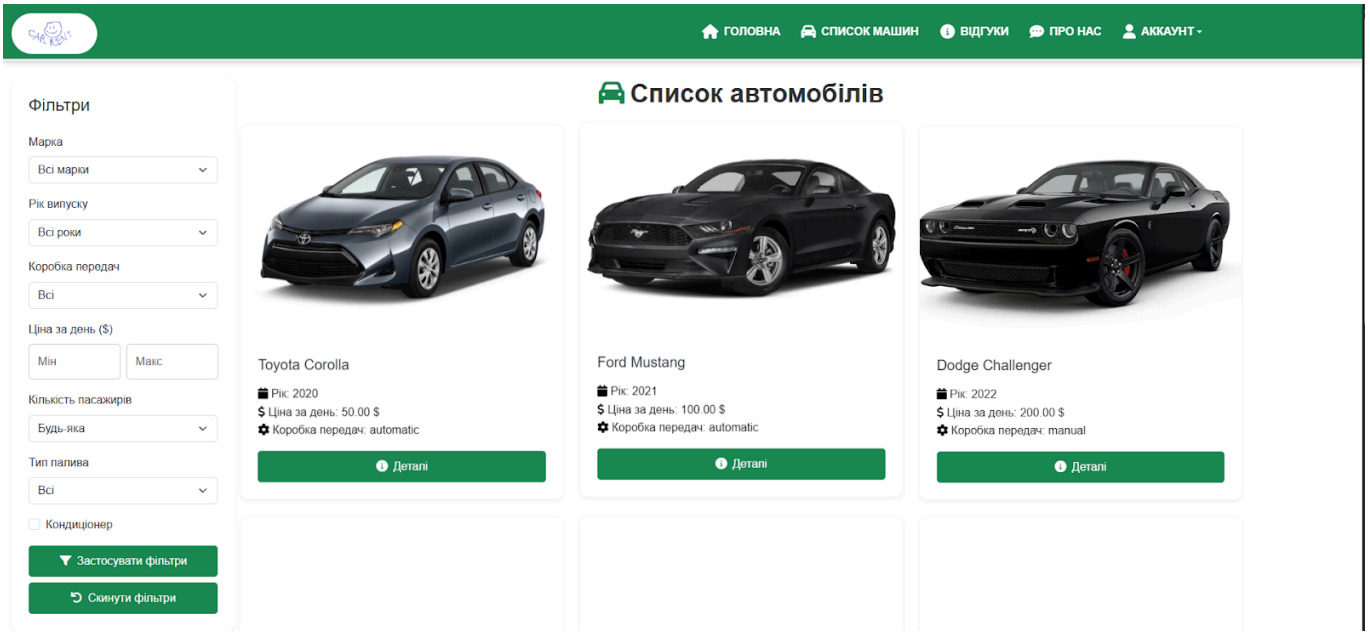


Рисунок 3.4 - Сторінка перегляду автомобілів

Сторінка деталей автомобіля (`car_details.php`):

Відображає повну інформацію про автомобіль: фотографії, марку, модель, рік, ціну, трансмісію, кількість пасажирів, тип пального, кондиціонер, багажник, кількість дверей, об’єм двигуна, пробіг.

Містить форму для бронювання (вибір дат, розрахунок загальної вартості).

Відображає відгуки користувачів, які вже орендували цей автомобіль.

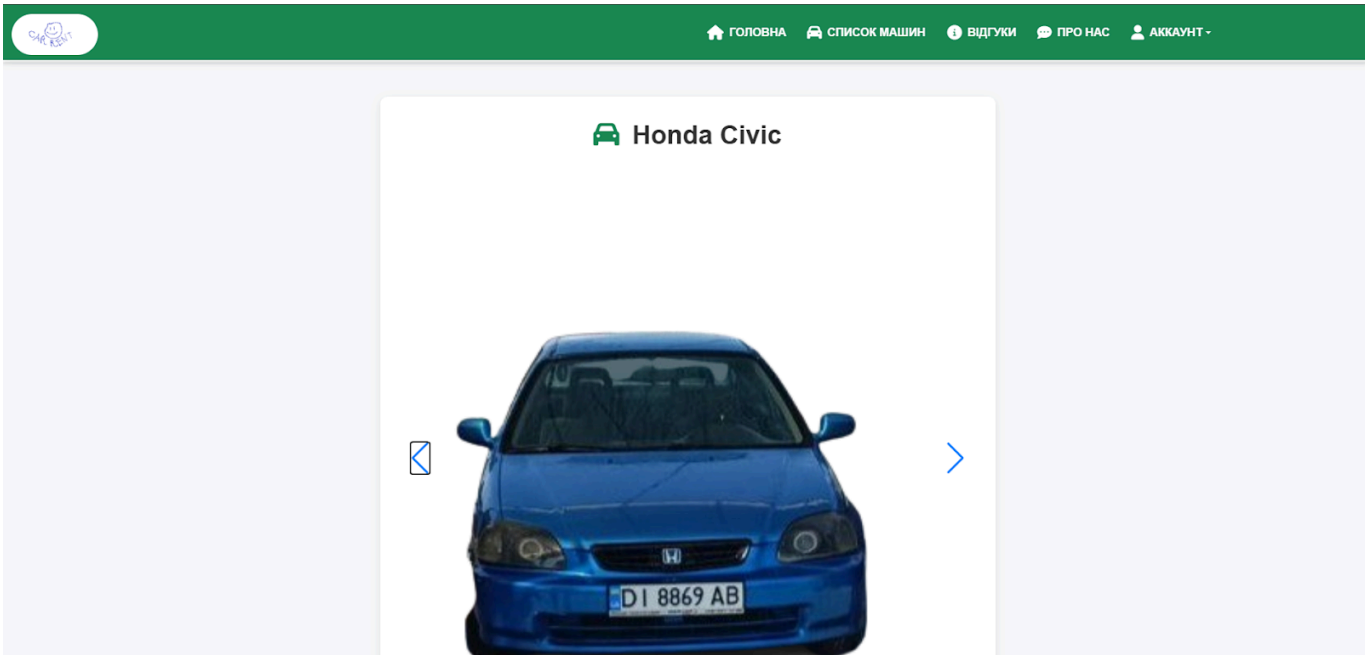


Рисунок 3.5 - Сторінка деталей автомобіля частина 1

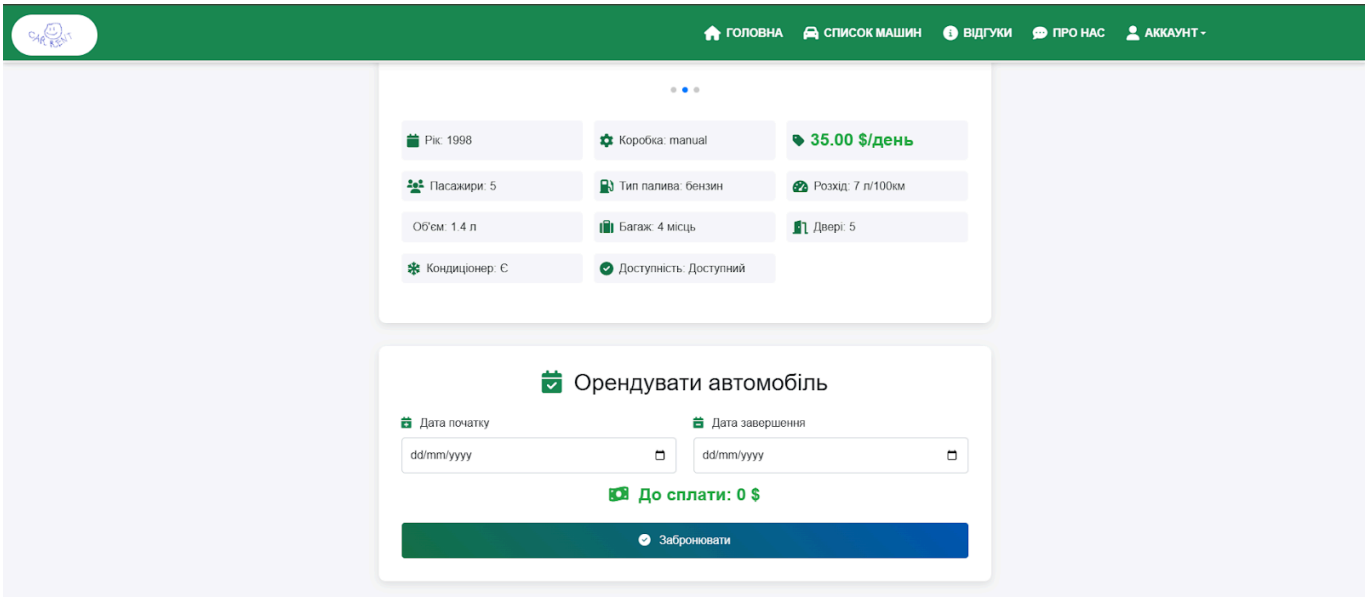


Рисунок 3.6 - Сторінка деталей автомобіля частина 2

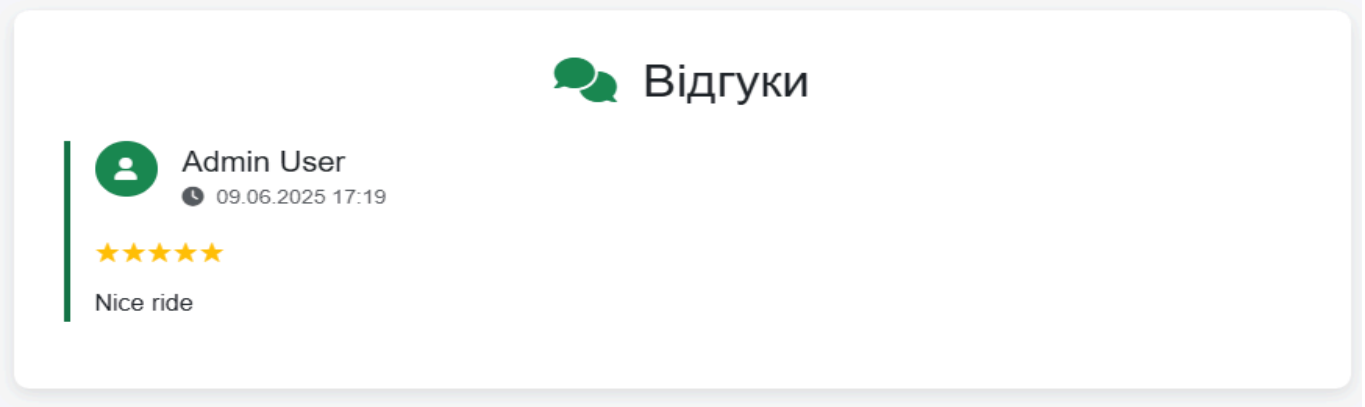


Рисунок 3.7 - Сторінка деталей автомобіля частина 3

Сторінка авторизації (`login.php`):

Форма для введення логіну та паролю.

Посилання на реєстрацію.

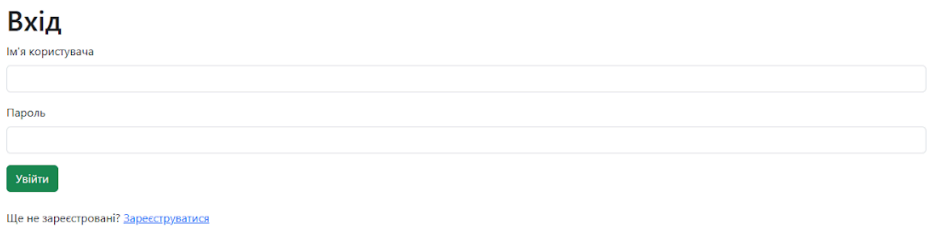


Рисунок 3.8 - Сторінка авторизації

Сторінка реєстрації (`register.php`):

Форма для введення логіну, паролю, email, імені, телефону.

Перевірка унікальності логіну та email, валідація паролю.

Рисунок 3.9 - Сторінка реєстрації

Сторінка контактної форми (`contact.php`):

Форма для введення імені, email, телефону, тексту повідомлення.

Валідація введених даних, відправка повідомлення.

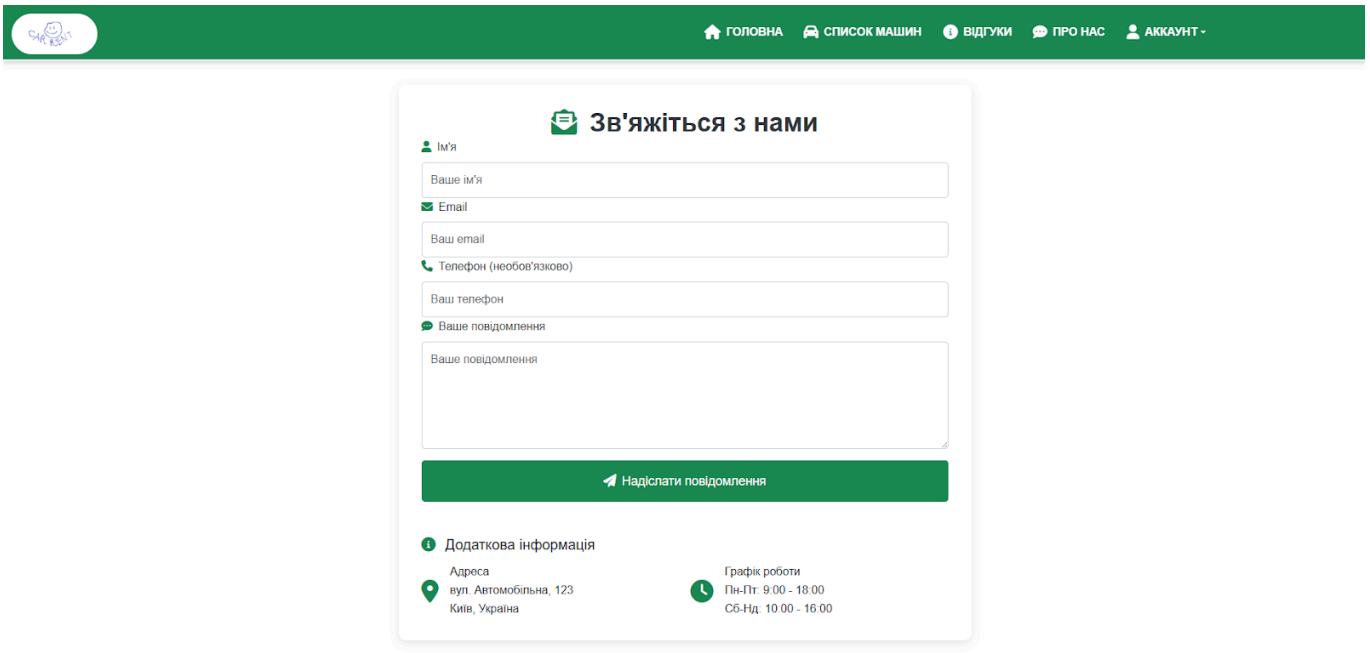


Рисунок 3.10 - Сторінка контактної форми

Сторінка адміністративного інтерфейсу (`admin/dashboard.php`):

Відображає загальну статистику (кількість автомобілів, бронювань, відгуків, повідомлень, користувачів).

Посилання на різні адміністративні модулі (керування автомобілями, бронюваннями, відгуками, повідомленнями, користувачами).

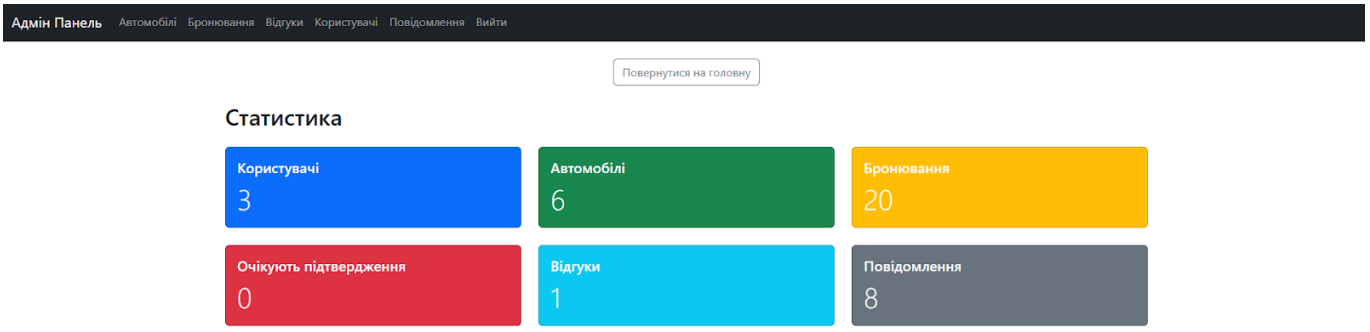


Рисунок 3.11 - Сторінка адміністративного інтерфейсу

Сторінки адміністративного керування (`admin/manage_*.php`):

Відображають списки відповідних сутностей (автомобілі, бронювання, відгуки, повідомлення, користувачі).

Містять форми для додавання, редагування, видалення, підтвердження/відхилення, модерації.

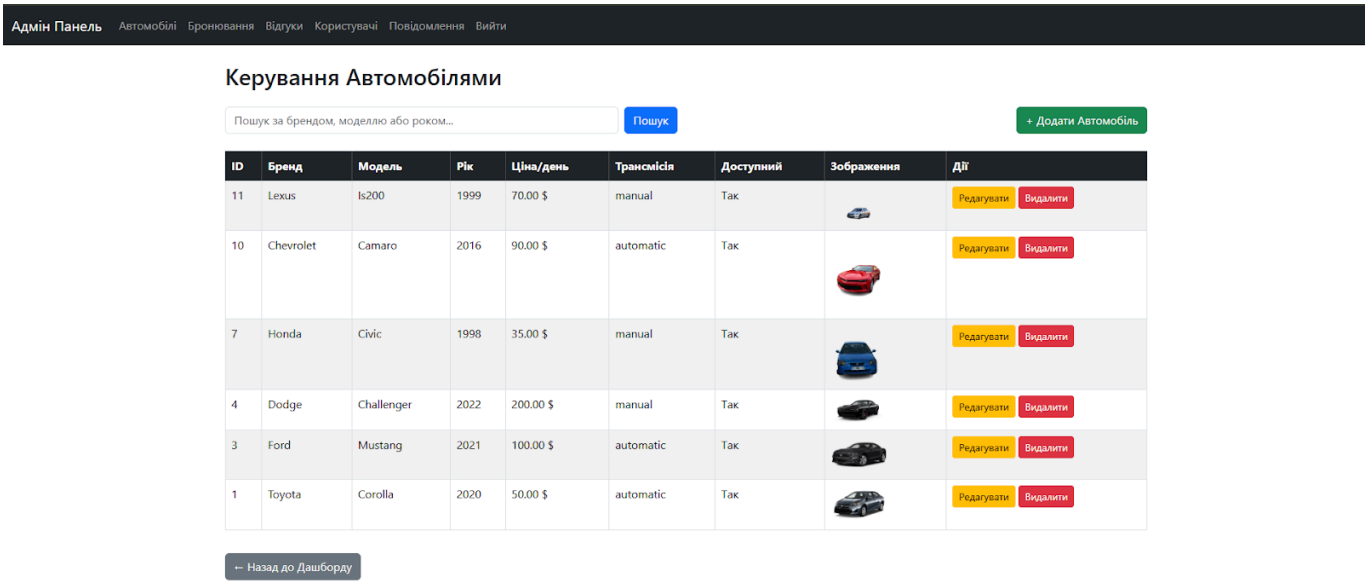


Рисунок 3.12 - Сторінки адміністративного керування автомобілем

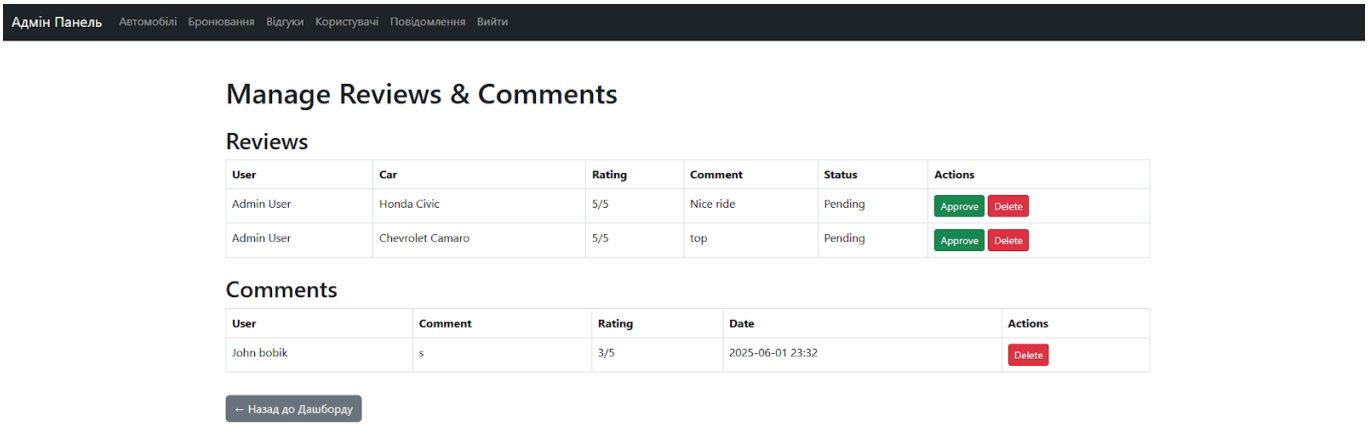


Рисунок 3.13 - Сторінки адміністративного керування відгуками

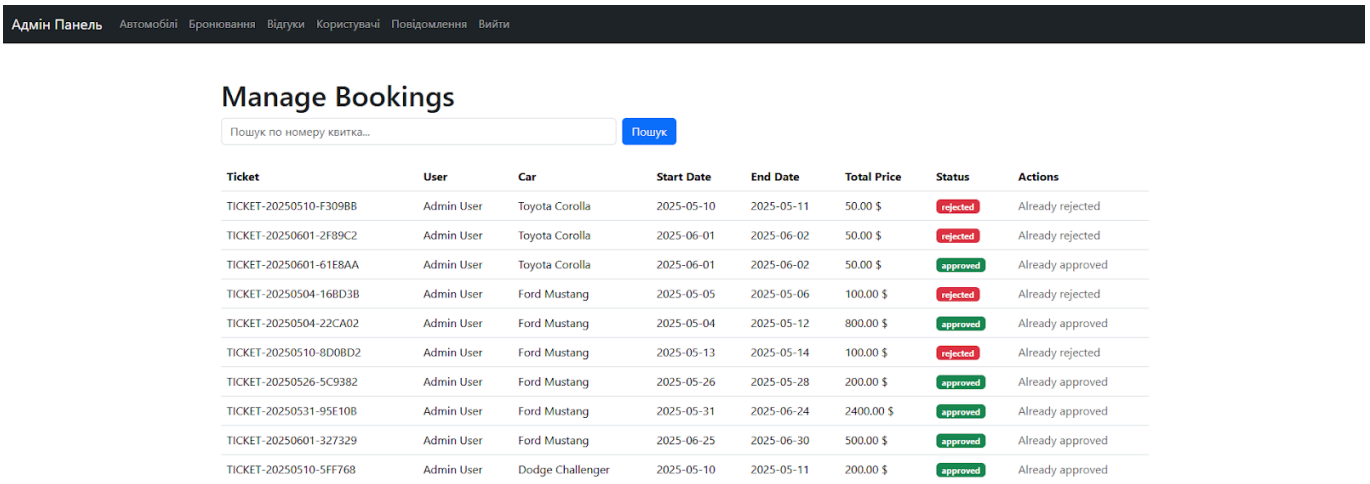


Рисунок 3.14 - Сторінки адміністративного керування бронюваннями

ВИСНОВОК

У процесі виконання бакалаврської дипломної роботи було розроблено та впроваджено сучасну веб-систему оренди автомобілів, яка повністю відповідає поставленим завданням та сучасним вимогам до подібних інформаційних систем [7]. Зокрема було створено оптимальну структуру бази даних, що включає 6 взаємопов'язаних таблиць для ефективного зберігання та обробки даних про автомобілі, користувачів, бронювання, відгуки та повідомлення.А також було розроблено алгоритм генерації унікальних ідентифікаторів бронювань (ticket_id), що забезпечує надійну ідентифікацію транзакцій. Для цього було реалізовано повний функціонал системи оренди автомобілів, який включає модуль управління автопарком з детальним описом характеристик автомобілів.

СПИСОК ДЖЕРЕЛ

1. Bootstrap Documentation. (2023). Bootstrap 5. [Електронний ресурс]. Available at: https://getbootstrap.com/docs/5.3/

2. Crockford, D. (2021). JavaScript: The Good Parts. O'Reilly Media. (170 сторінок)

3. Ducket, J. (2022). HTML & CSS: Design and Build Websites. John Wiley & Sons. (512 сторінок)

4. Evans, E. (2020). Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Professional. (560 сторінок)

5. Fowler, M. (2020). Patterns of Enterprise Application Architecture. Addison-Wesley Professional. (533 сторінки)

6. IEEE Software. (2023). Journal of Software Engineering and Applications. [Електронний ресурс]. Доступно: https://www.scirp.org/journal/jssea/

7. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.

8. Martin, R. (2018). Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall. (464 сторінки)

9. McConnell, S. (2020). Code Complete: A Practical Handbook of Software Construction. Microsoft Press. (914 сторінок)

10. Newman, S. (2021). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media. (616 сторінок)

11. PHP Documentation. (2023). PHP Manual. [Електронний ресурс]. Available at: https://www.php.net/manual/en/

12. W3C Web Content Accessibility Guidelines (WCAG) 2.1. (2023). [Електронний ресурс]. Available at: https://www.w3.org/TR/WCAG21/

ДОДАТКИ

Фрагмент вихідного коду:

index.php

```
<?php
// index.php
session_start();
require_once __DIR__ . '/config/database.php'; // Підключення до бази даних
require_once __DIR__ . '/models/Car.php'; // Підключення моделі

try {
    $carModel = new Car($pdo);
    $cars = $carModel->getAllCars();
} catch (Exception $e) {
    error_log("Error loading cars: " . $e->getMessage());
    $cars = [];
}
?>
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
     <meta name="description" content="Оренда автомобілів - широкий вибір автомобілів для оренди">
    <title>Оренда автомобілів</title>
                                                                    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
                                                                    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css" rel="stylesheet">
                                                                    <link href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css" rel="stylesheet">
                                                                    <link href="https://cdnjs.cloudflare.com/ajax/libs/hover.css/2.3.1/css/hover-min.css" rel="stylesheet">
    <link rel="stylesheet" href="assets/css/styles.css">


</head>
```

```
<body>
<div id="header">
<?php include('config/header.php');?>
</div>

    <!-- Hero Section -->
    <section class="hero-section text-center">
        <div class="container">
            <h1 class="display-4 mb-4 animate__animated animate__fadeInDown">
                <i class="fas fa-car text-success me-2"></i>Оренда автомобілів
            </h1>
            <p class="lead mb-4 animate__animated animate__fadeInUp">
                Знайдіть ідеальний автомобіль для вашої подорожі
            </p>
                 <a href="cars.php" class="btn btn-success btn-lg animate__animated
animate__fadeInUp">
                <i class="fas fa-search me-2"></i>Переглянути автомобілі
            </a>
        </div>
    </section>

    <!-- Features Section -->
    <section class="container mb-5">
        <h2 class="text-center section-title">Наші переваги</h2>
        <div class="row g-4 justify-content-center">
            <div class="col-md-4">
                <div class="feature-card text-center p-4 h-100">
                    <i class="fas fa-car-side feature-icon"></i>
                    <h3>Широкий вибір</h3>
                              <p>Великий вибір автомобілів різних класів та цінових
категорій</p>
                </div>
            </div>
            <div class="col-md-4">
                <div class="feature-card text-center p-4 h-100">
                    <i class="fas fa-shield-alt feature-icon"></i>
                    <h3>Безпека</h3>
                    <p>Всі автомобілі проходять регулярний технічний огляд</p>
                </div>
            </div>
            <div class="col-md-4">
                <div class="feature-card text-center p-4 h-100">
```

```
                    <i class="fas fa-clock feature-icon"></i>
                    <h3>Доступні ціни</h3>
                    <p>Конкурентні ціни та гнучкі умови оренди</p>
                </div>
            </div>
        </div>
    </section>

    <!-- Cars Section -->
    <section id="cars" class="container mb-5">
        <h2 class="text-center section-title">Доступні автомобілі</h2>
        <?php if (empty($cars)): ?>
            <div class="alert alert-info">
                Наразі немає доступних автомобілів для оренди.
            </div>
        <?php else: ?>
            <div class="row justify-content-center">
                <?php foreach ($cars as $car): ?>
                    <div class="col-12 col-sm-6 col-lg-4 mb-4">
                        <div class="card h-100 car-card">
                                                    <img  src="assets/images/<?=
htmlspecialchars($car['image']) ?>"
                                 class="card-img-top"
                                      alt="<?= htmlspecialchars($car['brand']) ?> <?=
htmlspecialchars($car['model']) ?>"
                                 loading="lazy">
                            <div class="card-body">
                                                        <h3  class="card-title  h5"><?=
htmlspecialchars($car['brand']) ?> <?= htmlspecialchars($car['model']) ?></h3>
                                <ul class="list-unstyled">
                                    <li><i class="fas fa-calendar text-success"></i>
Рік: <?= htmlspecialchars($car['year']) ?></li>
                                                    <li><i  class="fas  fa-dollar-sign
text-success"></i>  Ціна  за  день:  <?=  htmlspecialchars($car['price_per_day'])  ?>
$</li>
                                        <li><i class="fas fa-cog text-success"></i>
Коробка передач: <?= htmlspecialchars($car['transmission']) ?></li>
                                </ul>
                                   <a href="car_details.php?id=<?= $car['car_id'] ?>"
class="btn btn-success w-100">
                                    <i class="fas fa-info-circle me-2"></i>Деталі
                                </a>
                            </div>
```

```
                    </div>
                </div>
            <?php endforeach; ?>
        </div>
    <?php endif; ?>
</section>

<!-- Rental Process Section -->
<section class="cta-section">
    <div class="container">
        <h2 class="text-center mb-5">Як орендувати автомобіль?</h2>
        <div class="row g-4 justify-content-center">
            <div class="col-md-3">
                <div class="text-center">
                                <div class="rounded-circle bg-success text-white
d-inline-flex  align-items-center  justify-content-center  mb-3"  style="width:  60px;
height: 60px;">
                        <i class="fas fa-search fa-lg"></i>
                    </div>
                    <h4>1. Виберіть авто</h4>
                       <p class="text-muted">Оберіть автомобіль, який відповідає
вашим потребам</p>
                </div>
            </div>
            <div class="col-md-3">
                <div class="text-center">
                                <div class="rounded-circle bg-success text-white
d-inline-flex  align-items-center  justify-content-center  mb-3"  style="width:  60px;
height: 60px;">
                        <i class="fas fa-calendar-check fa-lg"></i>
                    </div>
                    <h4>2. Забронюйте</h4>
                            <p class="text-muted">Вкажіть дати та час отримання
автомобіля</p>
                </div>
            </div>
            <div class="col-md-3">
                <div class="text-center">
                                <div class="rounded-circle bg-success text-white
d-inline-flex  align-items-center  justify-content-center  mb-3"  style="width:  60px;
height: 60px;">
                        <i class="fas fa-car fa-lg"></i>
                    </div>
```

```
                        <h4>3. Отримайте авто</h4>
                         <p class="text-muted">Заберіть автомобіль у зручному для вас місці</p>
                    </div>
                </div>
                <div class="col-md-3">
                    <div class="text-center">
                                    <div class="rounded-circle bg-success text-white d-inline-flex align-items-center justify-content-center mb-3" style="width: 60px; height: 60px;">
                            <i class="fas fa-road fa-lg"></i>
                        </div>
                        <h4>4. Насолоджуйтесь</h4>
                                    <p class="text-muted">Подорожуйте з комфортом та впевненістю</p>
                    </div>
                </div>
            </div>
            <div class="text-center mt-5">
                <a href="cars.php" class="btn btn-success btn-lg">
                    <i class="fas fa-car me-2"></i>Почати оренду
                </a>
            </div>
        </div>
    </section>

<footer>
    <?php include('config/footer.php');?>
</footer>

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/aos@2.3.4/dist/aos.js"></script>
</body>
</html>
```

model/Car.php

```
<?php
// models/Car.php
```

```
class Car {
    private $pdo;

    public function __construct($pdo) {
        if (!$pdo instanceof PDO) {
            throw new InvalidArgumentException('Invalid PDO instance provided');
        }
        $this->pdo = $pdo;
    }

    // Отримання всіх автомобілів
    public function getAllCars() {
        try {
                $stmt = $this->pdo->query("SELECT * FROM cars WHERE available = TRUE
ORDER BY brand, model");
            return $stmt->fetchAll();
        } catch (PDOException $e) {
            error_log("Error fetching cars: " . $e->getMessage());
            return [];
        }
    }

    // Отримання автомобіля за ID
    public function getCarById($id) {
        if (!is_numeric($id)) {
            throw new InvalidArgumentException('Invalid car ID');
        }
        try {
            $stmt = $this->pdo->prepare("SELECT * FROM cars WHERE car_id = ?");
            $stmt->execute([$id]);
            return $stmt->fetch();
        } catch (PDOException $e) {
            error_log("Error fetching car by ID: " . $e->getMessage());
            return false;
        }
    }

    public function getReviewById($review_id) {
        if (!is_numeric($review_id)) {
            throw new InvalidArgumentException('Invalid review ID');
        }
        try {
            $stmt = $this->pdo->prepare("SELECT * FROM reviews WHERE id = ?");
```

```
            $stmt->execute([$review_id]);
            return $stmt->fetch() ?: false;
        } catch (PDOException $e) {
            error_log("Error fetching review: " . $e->getMessage());
            return false;
        }
    }

public function bookCar($user_id, $car_id, $start_date, $end_date, $total_price, $ticket_id) {
    if (!is_numeric($user_id) || !is_numeric($car_id) || !is_numeric($total_price)) {
        return ['success' => false, 'message' => 'Невірні параметри бронювання'];
    }

    try {
        $this->pdo->beginTransaction();

        // Перевірка, чи цей користувач вже має бронювання на ці дати для цього авто
        $stmt = $this->pdo->prepare("
            SELECT COUNT(*) FROM bookings
            WHERE car_id = ?
              AND user_id = ?
              AND status = 'approved'
              AND start_date <= ?
              AND end_date >= ?
        ");
        $stmt->execute([$car_id, $user_id, $end_date, $start_date]);
        $count = $stmt->fetchColumn();

        if ($count > 0) {
            $this->pdo->rollBack();
                  return ['success' => false, 'message' => 'Ви вже забронювали цей автомобіль на вибрані дати'];
        }

          // Далі просто додаємо бронювання без перевірки на інші броні (дозволяємо іншим користувачам дублювати бронювання)
        $stmt = $this->pdo->prepare("
            INSERT INTO bookings (car_id, user_id, start_date, end_date, total_price, ticket_id, status)
                   VALUES (:car_id, :user_id, :start_date, :end_date, :total_price, :ticket_id, 'pending')
        ");
```

```
        $result = $stmt->execute([
            ':car_id' => $car_id,
            ':user_id' => $user_id,
            ':start_date' => $start_date,
            ':end_date' => $end_date,
            ':total_price' => $total_price,
            ':ticket_id' => $ticket_id
        ]);

        $this->pdo->commit();
          return ['success' => true, 'message' => 'Бронювання успішно оформлено! Ваш номер тікету: ' . $ticket_id];
    } catch (Exception $e) {
        $this->pdo->rollBack();
        error_log("Booking error: " . $e->getMessage());
            return ['success' => false, 'message' => 'Помилка при бронюванні: ' . $e->getMessage()];
    }
}
```

```
// Отримання відфільтрованих машин
public function getFilteredCars($brand = null, $year = null, $transmission = null, $minPrice = null, $maxPrice = null, $passengers = null, $fuelType = null, $airConditioning = null) {
    $sql = "SELECT * FROM cars WHERE available = 1";
    $params = [];

    if ($brand) {
        $sql .= " AND brand = ?";
        $params[] = $brand;
    }
    if ($year) {
        $sql .= " AND year = ?";
        $params[] = $year;
    }
    if ($transmission) {
        $sql .= " AND transmission = ?";
        $params[] = $transmission;
```

```
        }
        if ($minPrice !== null && $minPrice !== '') {
            $sql .= " AND price_per_day >= ?";
            $params[] = $minPrice;
        }
        if ($maxPrice !== null && $maxPrice !== '') {
            $sql .= " AND price_per_day <= ?";
            $params[] = $maxPrice;
        }
        if ($passengers !== null && $passengers !== '') {
            $sql .= " AND passengers = ?";
            $params[] = $passengers;
        }
        if ($fuelType) {
            $sql .= " AND fuel_type = ?";
            $params[] = $fuelType;
        }
        if ($airConditioning) {
            $sql .= " AND air_conditioning = 1";
        }

        $stmt = $this->pdo->prepare($sql);
        $stmt->execute($params);
        return $stmt->fetchAll();
    }

    // Отримання унікальних марок
    public function getUniqueBrands() {
        $stmt = $this->pdo->query("SELECT DISTINCT brand FROM cars");
        return $stmt->fetchAll(PDO::FETCH_COLUMN);
    }

    // Отримання унікальних років випуску
    public function getUniqueYears() {
        $stmt = $this->pdo->query("SELECT DISTINCT year FROM cars ORDER BY year DESC");
        return $stmt->fetchAll(PDO::FETCH_COLUMN);
    }

    // Отримання унікальних типів коробок передач
    public function getUniqueTransmissions() {
        $stmt = $this->pdo->query("SELECT DISTINCT transmission FROM cars");
        return $stmt->fetchAll(PDO::FETCH_COLUMN);
    }
```

```
    public function getCarReviews($car_id) {
        $stmt = $this->pdo->prepare("
             SELECT r.id, r.user_id, u.username, u.full_name, r.rating, r.comment,
r.created_at
        FROM reviews r
        JOIN users u ON r.user_id = u.id
        WHERE r.car_id = ?
    ");
        $stmt->execute([$car_id]);
        return $stmt->fetchAll();
    }

public function addReview($car_id, $user_id, $booking_id, $rating, $comment) {
    if (!is_numeric($rating) || $rating < 1 || $rating > 5) {
        throw new InvalidArgumentException('Invalid rating value');
    }

    try {
        $stmt = $this->pdo->prepare("
                INSERT INTO reviews (car_id, user_id, booking_id, rating, comment,
created_at)
            VALUES (?, ?, ?, ?, ?, NOW())
        ");
                 return $stmt->execute([$car_id, $user_id, $booking_id, $rating,
htmlspecialchars($comment)]);
    } catch (PDOException $e) {
        error_log("Error adding review: " . $e->getMessage());
        return false;
    }
}

public function hasUserReviewedBooking($booking_id, $user_id) {
    $stmt = $this->pdo->prepare("
        SELECT COUNT(*) as count
        FROM reviews
        WHERE booking_id = ? AND user_id = ?
    ");
    $stmt->execute([$booking_id, $user_id]);
    $result = $stmt->fetch();
    return $result['count'] > 0;
}
```

```
public function updateReview($review_id, $rating, $comment) {
    $stmt = $this->pdo->prepare("
        UPDATE reviews
        SET rating = ?, comment = ?, created_at = NOW()
        WHERE id = ?
    ");
    $stmt->execute([$rating, $comment, $review_id]);
    return $stmt->rowCount() > 0;
}

public function deleteReview($review_id) {
    $stmt = $this->pdo->prepare("DELETE FROM reviews WHERE id = ?");
    $stmt->execute([$review_id]);
    return $stmt->rowCount() > 0;
}

// Отримання унікальних типів палива
public function getUniqueFuelTypes() {
    $stmt = $this->pdo->query("SELECT DISTINCT fuel_type FROM cars WHERE fuel_type IS
NOT NULL ORDER BY fuel_type");
    return $stmt->fetchAll(PDO::FETCH_COLUMN);
}

// Отримання унікальних кількостей пасажирів
public function getUniquePassengers() {
     $stmt = $this->pdo->query("SELECT DISTINCT passengers FROM cars WHERE passengers
IS NOT NULL ORDER BY passengers");
    return $stmt->fetchAll(PDO::FETCH_COLUMN);
}
}
?>
model/User.php
<?php
// models/Car.php
class Car {
    private $pdo;

    public function __construct($pdo) {
        if (!$pdo instanceof PDO) {
            throw new InvalidArgumentException('Invalid PDO instance provided');
        }
        $this->pdo = $pdo;
    }
```

```
    // Отримання всіх автомобілів
    public function getAllCars() {
        try {
               $stmt = $this->pdo->query("SELECT * FROM cars WHERE available = TRUE
ORDER BY brand, model");
            return $stmt->fetchAll();
        } catch (PDOException $e) {
            error_log("Error fetching cars: " . $e->getMessage());
            return [];
        }
    }

    // Отримання автомобіля за ID
    public function getCarById($id) {
        if (!is_numeric($id)) {
            throw new InvalidArgumentException('Invalid car ID');
        }
        try {
            $stmt = $this->pdo->prepare("SELECT * FROM cars WHERE car_id = ?");
            $stmt->execute([$id]);
            return $stmt->fetch();
        } catch (PDOException $e) {
            error_log("Error fetching car by ID: " . $e->getMessage());
            return false;
        }
    }

    public function getReviewById($review_id) {
        if (!is_numeric($review_id)) {
            throw new InvalidArgumentException('Invalid review ID');
        }
        try {
            $stmt = $this->pdo->prepare("SELECT * FROM reviews WHERE id = ?");
            $stmt->execute([$review_id]);
            return $stmt->fetch() ?: false;
        } catch (PDOException $e) {
            error_log("Error fetching review: " . $e->getMessage());
            return false;
        }
    }
```

```
public function bookCar($user_id, $car_id, $start_date, $end_date, $total_price, $ticket_id) {
    if (!is_numeric($user_id) || !is_numeric($car_id) || !is_numeric($total_price)) {
        return ['success' => false, 'message' => 'Невірні параметри бронювання'];
    }

    try {
        $this->pdo->beginTransaction();

        // Перевірка, чи цей користувач вже має бронювання на ці дати для цього авто
        $stmt = $this->pdo->prepare("
            SELECT COUNT(*) FROM bookings
            WHERE car_id = ?
              AND user_id = ?
              AND status = 'approved'
              AND start_date <= ?
              AND end_date >= ?
        ");
        $stmt->execute([$car_id, $user_id, $end_date, $start_date]);
        $count = $stmt->fetchColumn();

        if ($count > 0) {
            $this->pdo->rollBack();
                  return ['success' => false, 'message' => 'Ви вже забронювали цей автомобіль на вибрані дати'];
        }

          // Далі просто додаємо бронювання без перевірки на інші броні (дозволяємо іншим користувачам дублювати бронювання)
        $stmt = $this->pdo->prepare("
            INSERT INTO bookings (car_id, user_id, start_date, end_date, total_price, ticket_id, status)
                   VALUES (:car_id, :user_id, :start_date, :end_date, :total_price, :ticket_id, 'pending')
        ");

        $result = $stmt->execute([
            ':car_id' => $car_id,
            ':user_id' => $user_id,
            ':start_date' => $start_date,
            ':end_date' => $end_date,
            ':total_price' => $total_price,
            ':ticket_id' => $ticket_id
```

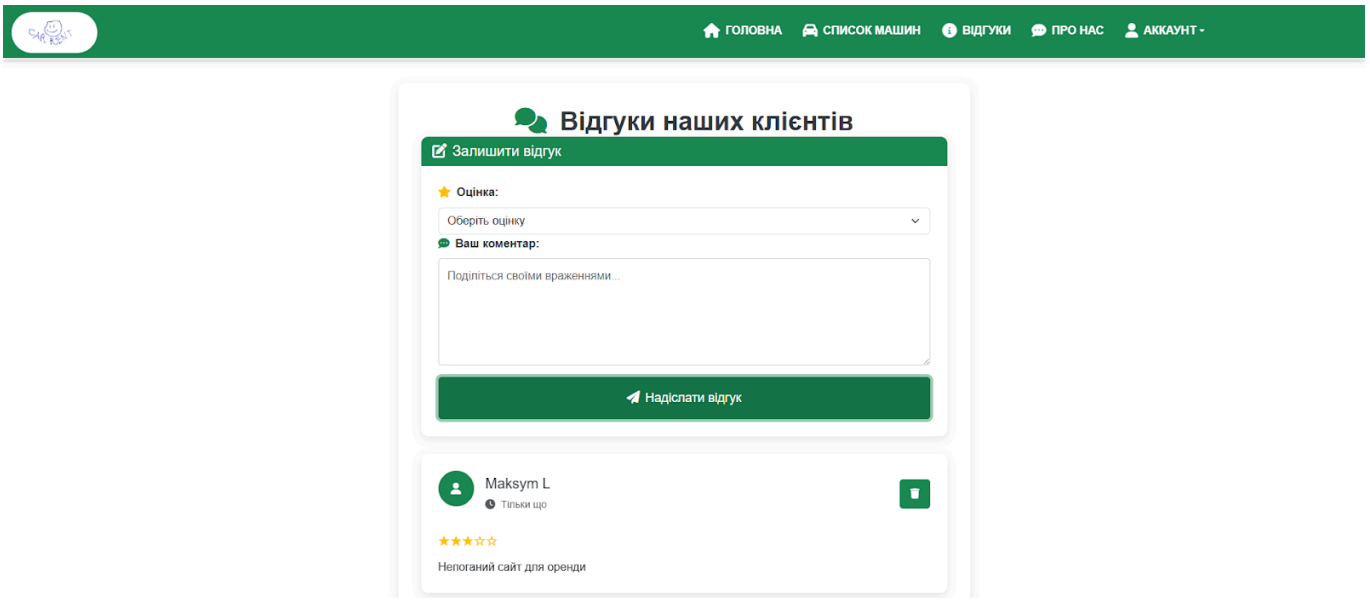
```
        ]);

        $this->pdo->commit();
          return ['success' => true, 'message' => 'Бронювання успішно оформлено! Ваш номер тікету: ' . $ticket_id];
    } catch (Exception $e) {
        $this->pdo->rollBack();
        error_log("Booking error: " . $e->getMessage());
            return ['success' => false, 'message' => 'Помилка при бронюванні: ' . $e->getMessage()];
    }
}
```

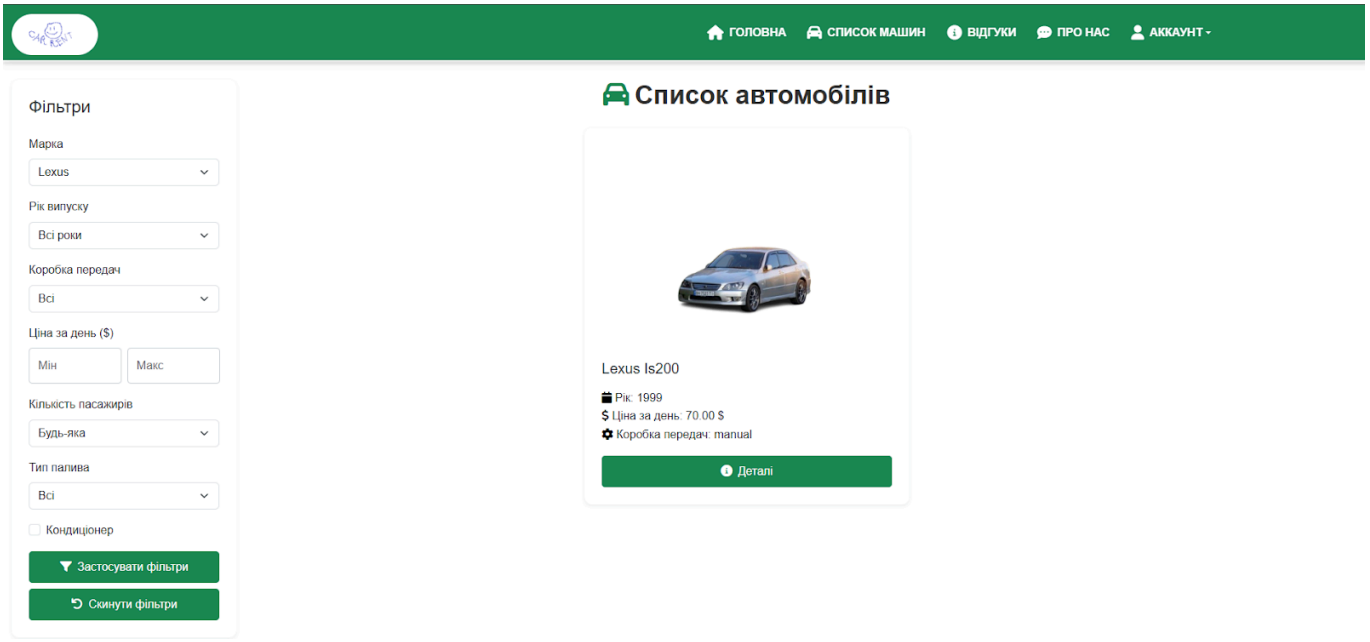
```
// Отримання відфільтрованих машин
public function getFilteredCars($brand = null, $year = null, $transmission = null, $minPrice = null, $maxPrice = null, $passengers = null, $fuelType = null, $airConditioning = null) {
    $sql = "SELECT * FROM cars WHERE available = 1";
    $params = [];

    if ($brand) {
        $sql .= " AND brand = ?";
        $params[] = $brand;
    }
    if ($year) {
        $sql .= " AND year = ?";
        $params[] = $year;
    }
    if ($transmission) {
        $sql .= " AND transmission = ?";
        $params[] = $transmission;
    }
    if ($minPrice !== null && $minPrice !== '') {
        $sql .= " AND price_per_day >= ?";
        $params[] = $minPrice;
    }
    if ($maxPrice !== null && $maxPrice !== '') {
        $sql .= " AND price_per_day <= ?";
        $params[] = $maxPrice;
```

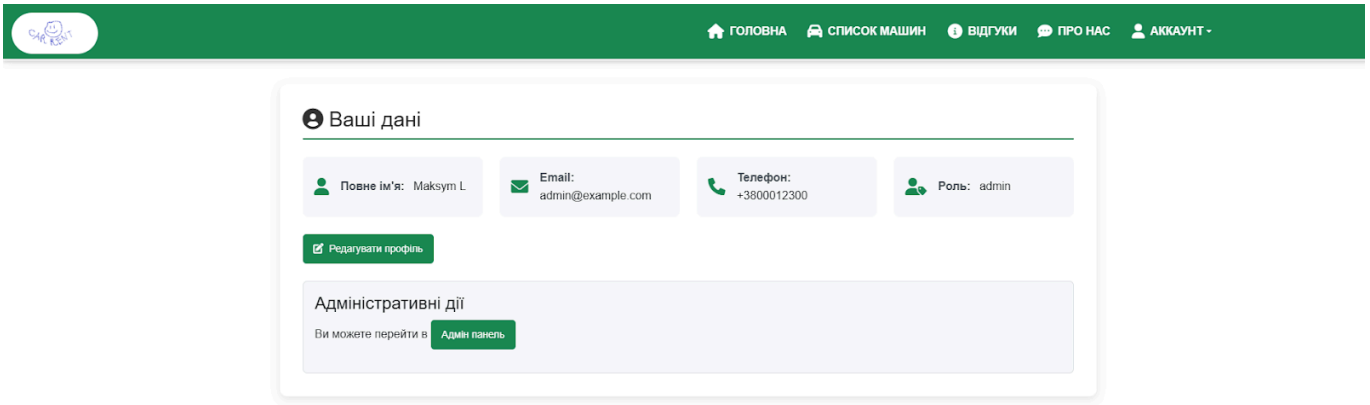
```
    }
    if ($passengers !== null && $passengers !== '') {
        $sql .= " AND passengers = ?";
        $params[] = $passengers;
    }
    if ($fuelType) {
        $sql .= " AND fuel_type = ?";
        $params[] = $fuelType;
    }
    if ($airConditioning) {
        $sql .= " AND air_conditioning = 1";
    }

    $stmt = $this->pdo->prepare($sql);
    $stmt->execute($params);
    return $stmt->fetchAll();
}

// Отримання унікальних марок
public function getUniqueBrands() {
    $stmt = $this->pdo->query("SELECT DISTINCT brand FROM cars");
    return $stmt->fetchAll(PDO::FETCH_COLUMN);
}

// Отримання унікальних років випуску
public function getUniqueYears() {
    $stmt = $this->pdo->query("SELECT DISTINCT year FROM cars ORDER BY year DESC");
    return $stmt->fetchAll(PDO::FETCH_COLUMN);
}

// Отримання унікальних типів коробок передач
public function getUniqueTransmissions() {
    $stmt = $this->pdo->query("SELECT DISTINCT transmission FROM cars");
    return $stmt->fetchAll(PDO::FETCH_COLUMN);
}

    public function getCarReviews($car_id) {
        $stmt = $this->pdo->prepare("
            SELECT r.id, r.user_id, u.username, u.full_name, r.rating, r.comment,
r.created_at
        FROM reviews r
        JOIN users u ON r.user_id = u.id
        WHERE r.car_id = ?
```

```
    ");
        $stmt->execute([$car_id]);
        return $stmt->fetchAll();
    }

public function addReview($car_id, $user_id, $booking_id, $rating, $comment) {
    if (!is_numeric($rating) || $rating < 1 || $rating > 5) {
        throw new InvalidArgumentException('Invalid rating value');
    }

    try {
        $stmt = $this->pdo->prepare("
                INSERT INTO reviews (car_id, user_id, booking_id, rating, comment,
created_at)
            VALUES (?, ?, ?, ?, ?, NOW())
        ");
                 return  $stmt->execute([$car_id,  $user_id,  $booking_id,  $rating,
htmlspecialchars($comment)]);
    } catch (PDOException $e) {
        error_log("Error adding review: " . $e->getMessage());
        return false;
    }
}

public function hasUserReviewedBooking($booking_id, $user_id) {
    $stmt = $this->pdo->prepare("
        SELECT COUNT(*) as count
        FROM reviews
        WHERE booking_id = ? AND user_id = ?
    ");
    $stmt->execute([$booking_id, $user_id]);
    $result = $stmt->fetch();
    return $result['count'] > 0;
}

public function updateReview($review_id, $rating, $comment) {
    $stmt = $this->pdo->prepare("
        UPDATE reviews
        SET rating = ?, comment = ?, created_at = NOW()
        WHERE id = ?
    ");
    $stmt->execute([$rating, $comment, $review_id]);
    return $stmt->rowCount() > 0;
```

```
}

public function deleteReview($review_id) {
    $stmt = $this->pdo->prepare("DELETE FROM reviews WHERE id = ?");
    $stmt->execute([$review_id]);
    return $stmt->rowCount() > 0;
}

// Отримання унікальних типів палива
public function getUniqueFuelTypes() {
    $stmt = $this->pdo->query("SELECT DISTINCT fuel_type FROM cars WHERE fuel_type IS
NOT NULL ORDER BY fuel_type");
    return $stmt->fetchAll(PDO::FETCH_COLUMN);
}

// Отримання унікальних кількостей пасажирів
public function getUniquePassengers() {
     $stmt = $this->pdo->query("SELECT DISTINCT passengers FROM cars WHERE passengers
IS NOT NULL ORDER BY passengers");
    return $stmt->fetchAll(PDO::FETCH_COLUMN);
}
}
?>
```


ГОЛОВНА
СПИСОК МАШИН
ВІДГУКИ
ПРО НАС
АККАУНТ
Відгуки наших клієнтів
Залишити відгук
Оцінка:
Оберіть оцінку
Ваш коментар:
Поділіться своїми враженнями...
Надіслати відгук
Maksym L
Тільки що
Непоганий сайт для оренди


ГОЛОВНА
СПИСОК МАШИН
ВІДГУКИ
ПРО НАС
АККАУНТ
Фільтри
Марка
Lexus
Рік випуску
Всі роки
Коробка передач
Всі
Ціна за день ($)
Мін
Макс
Кількість пасажирів
Будь-яка
Тип палива
Всі
Кондиціонер
Застосувати фільтри
Скинути фільтри
Список автомобілів
Lexus Is200
Рік: 1999
Ціна за день: 70.00 $
Коробка передач: manual
Деталі

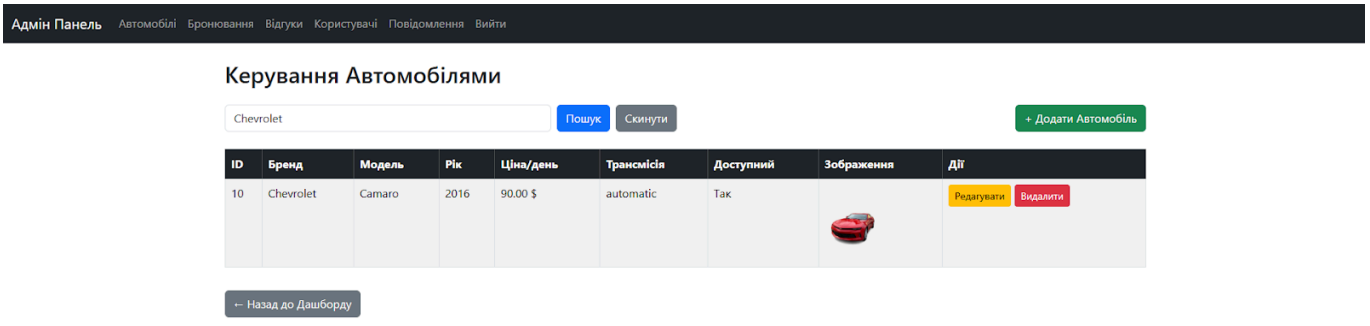

ГОЛОВНА
СПИСОК МАШИН
ВІДГУКИ
ПРО НАС
АККАУНТ
Ваші дані
Повне ім'я: Maksym L
Email:
admin@example.com
Телефон:
+3800012300
Роль: admin
Редагувати профіль
Адміністративні дії
Ви можете перейти в
Адмін панель

Ваші бронювання

Тікет	Автомобіль	Дата початку	Дата завершення	Ціна	Статус	Дії
TICKET-20250510-F309BB	Toyota Corolla	2025-05-10	2025-05-11	50.00 $	Скасовано	Деталі Недоступно
TICKET-20250601-2F89C2	Toyota Corolla	2025-06-01	2025-06-02	50.00 $	Скасовано	Деталі Недоступно
TICKET-20250601-61E8AA	Toyota Corolla	2025-06-01	2025-06-02	50.00 $	Підтверджено	Деталі Відгук
TICKET-20250504-16BD3B	Ford Mustang	2025-05-05	2025-05-06	100.00 $	Скасовано	Деталі Недоступно
TICKET-20250504-22CA02	Ford Mustang	2025-05-04	2025-05-12	800.00 $	Підтверджено	Деталі Відгук
TICKET-20250510-8D0BD2	Ford Mustang	2025-05-13	2025-05-14	100.00 $	Скасовано	Деталі Недоступно
TICKET-20250526-5C9382	Ford Mustang	2025-05-26	2025-05-28	200.00 $	Підтверджено	Деталі Відгук

Ваші відгуки

Автомобіль	Рейтинг	Коментар	Дата	Дії
Chevrolet Camaro	★★★★★	top	2025-06-01 23:39:00	Редагувати Видалити
Honda Civic	★★★★★	Nice ride	2025-06-09 17:19:50	Редагувати Видалити



Додати Новий Автомобіль

Бренд

Модель

Рік випуску

Ціна за день ($)

Трансмісія

Механічна

Доступний

Кількість пасажирів

Тип двигуна

Оберіть тип двигуна

Розхід палива (л/100км)

Кондиціонер

Кількість багажу (місць)

Кількість дверей

Об'єм двигуна (л)

Зображення 1

Choose file | No file chosen

Зображення 2

Choose file | No file chosen

Зображення 3

Choose file | No file chosen

Зображення 4

Choose file | No file chosen

Додати Автомобіль

Назад до керування авто

Редагувати Автомобіль

Бренд

Honda

Модель

Civic

Рік випуску

1998

Ціна за день ($)

35.00

Трансмісія

Механічна

Доступний

Кількість пасажирів

5

Тип двигуна

Бензин

Розхід палива (л/100км)

7

Кондиціонер

Кількість багажу (місць)

4

Кількість дверей

5

Об'єм двигуна (л)

1.4

Зображення 1

Choose file | No file chosen

Поточне зображення: 

Зображення 2

Choose file | No file chosen

Поточне зображення: 

Зображення 3

Choose file | No file chosen

Поточне зображення: 

Зображення 4

Choose file | No file chosen

Оновити Автомобіль

Назад до керування авто