Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Центр післядипломної освіти

Кафедра Програмної інженерії

(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для підтримки оренди автомобілів

(тема)

Виконав:
студент 2 курсу, групи ПЗПІп-22-2
Слободяник О.Ю.
(прізвище, ініціали)
Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)
Керівник доц. кафедри ПІ Русакова Н. Є.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри, проф. _____ З.В. Дудар
(підпис) (посада, прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Центр післядипломної освіти

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програми Програмна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«___»_____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Слободянику Олегу Юрійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмна система для підтримки оренди автомобілів*
затверджена наказом університету від 17.06.2024 р. № 588Ст

2. Термін подання студентом роботи до екзаменаційної комісії 22.07.2024 р.

3. Вихідні данні до роботи *вимоги до розроблюваної програми, вимоги до архітектури системи, електронні ресурси за обраною темою, мови програмування PHP та Javascript, СКБД MySQL, середовища розробки PhpStorm. Сервер OpenServer.*

4. Перелік питань, що потрібно опрацювати в роботі *вступ, аналіз предметної області, формування вимог до програмного забезпечення, архітектура та проектування розробленого програмного забезпечення, опис прийнятих програмних рішень, тестування програмного забезпечення, висновки, додатки.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Позначка про виконання
1	Аналіз проблемної області	6.05.2024 – 8.05.2024	Виконано
2	Розробка постановки задачі	8.05.2024 – 11.05.2024	Виконано
3	Аналіз та моделювання предметної області	12.05.2024 – 15.05.2024	Виконано
4	Проектування БД	15.05.2024 – 19.05.2024	Виконано
5	Розробка алгоритмів	20.05.2024 – 27.05.2024	Виконано
6	Проектування архітектури програмної системи	29.05.2024 – 10.06.2024	Виконано
7	Програмна реалізація системи	01.06.2024 – 05.07.2024	Виконано
8	Підготовка пояснювальної записки	05.06.2024 – 11.07.2024	Виконано
9	Підготовка презентації та доповіді	13.07.2024	Виконано
10	Нормоконтроль	15.07.2024	Виконано
11	Рецензування	17.07.2024	Виконано
12	Занесення записки в електронний архів	18.07.2024	Виконано
13	Попередній захист	19.07.2024	Виконано
14	Допуск до захисту у зав. кафедри	22.07.2024	Виконано

Дата видачі завдання 6 травня 2024р.

Студент _____ Слободяник О.

(підпис) (прізвище, ініціали)

Керівник роботи _____ доц.кафедри ПІ Русакова Н.Є.

(підпис) (прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 68 стор., 24 рис., 3 табл., 10 джерел.

РЕЛЯЦІЙНА БАЗА ДАНИХ, СКБД, CSS, FRAMEWORK BOOTSTRAP, HTML, JAVASCRIPT, MYSQL, NGINX+APACH, PHP

Метою роботи є розробка програмної системи для підтримки оренди автомобілів з використанням реляційних баз даних та логіки об'єктно-орієнтованого програмування.

В ході створення системи використано середу розробки IDE PhpStorm 2022.1, локальний веб-сервер для Windows (Open Server) з Apach+Nginx, мову програмування PHP, веб-технології HTML, CSS, JAVASCRIPT, Framework Bootstrap та систему керування базами даних MySQL.

В результаті роботи створено програмну систему, яка дозволяє взаємодіяти та зберігати дані про автомобілі, менеджерів, клієнтів, договори між менеджерами та клієнтами та ДТП автомобілів. Також програма дозволяє шукати, фільтрувати інформацію, отримувати статистики та звіти, підтримує процес оренди автомобіля на основних етапах.

BOOTSTRAP FRAMEWORK, CSS, DBD, HTML, JAVASCRIPT, MYSQL, NGINX+APACH, PHP, RELATIONAL DATABASE.

The aim of the work is to develop a software system to support car rental using relational databases and object-oriented programming logic.

The system was developed using the PhpStorm 2022.1 IDE, a local web server for Windows (Open Server) with Apach+Nginx, the PHP programming language,

HTML, CSS, JAVASCRIPT web technologies, the Bootstrap Framework, and the MySQL database management system.

As a result of the work, a software system was created that allows you to interact and store data about cars, managers, clients, contracts between managers and clients, and car accidents. The program also allows you to search, filter information, receive statistics and reports, and supports the car rental process at the main stages.

Я, Слободяник Олег Юрійович, студент гр. ПЗПІп-22-2, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для підтримки оренди автомобілів», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

Перелік скорочень 7
Вступ 8
1 Аналіз проблемної галузі 9
1.1 Аналіз проблемної галузі 9
1.2 Аналіз існуючих аналогів 13
1.3 Постановка задачі 16
2 Формування вимог до програмної системи 18
3 Архітектура та проєктування програмного забезпечення 20
3.1 Аналіз та моделювання предметної області 20
3.2 Проектування бази даних 25
3.3 Розробка алгоритмів підтримки оренди автомобіля 26
3.4 Розробка архітектури системи 30
3.5 Створення UI / UX або іншого дизайну системи 31
4 Опис програмної реалізації 34
4.1 Вибір засобів програмної реалізації 34
4.2 Описання фізичної моделі бази даних 34
4.3 Опис серверної частини системи 36
4.4 Опис інтерфейсу користувача 37
5 Тестування розробленого програмного забезпечення 45
5.1 Обґрунтування вибору виду тестування 45
5.2 Опис тестування 45
Висновки 48
Перелік джерел посилання 49
Додаток А 50
Додаток Б 51
Додаток В 59

ПЕРЕЛІК СКОРОЧЕНЬ

DBMS – Database Management System;

CSS – Cascading Style Sheets;

FTP – File Transfer Protocol;

HTML – HyperText Markup Language;

HTTP – Hypertext Transfer Protocol;

MYSQL – My Structured Query Language;

PHP – Hypertext Preprocessor (Personal Home Page Tools).

ВСТУП

Оренда автомобілів стає все більш популярною через зростаючу потребу в мобільності, особливо в умовах урбанізації та змін у транспортних вподобаннях. Актуальність проблеми оренди автомобілів обумовлена необхідністю забезпечення гнучкого та економічно ефективного транспорту, який може задовольнити потреби різних користувачів, від туристів до бізнесменів. Зручність, економія коштів на обслуговування власного автомобіля та можливість вибору транспортного засобу під конкретні потреби роблять оренду автомобілів вигідним рішенням для багатьох.

Даний бізнес-процес може бути підтриманий за рахунок розробки програмної системи, яка дозволяє взаємодіяти та зберігати дані про автомобілі, менеджерів, клієнтів, договори між менеджерами та клієнтами та ДТП автомобілів. Також програма дозволяє шукати, фільтрувати інформацію, отримувати статистики продажів, підтримує задачу автоматизації керування контрактом.

Під час дипломування було проведено аналіз та моделювання предментої області, проектування БД, розробку архітектури системи та її проектування.

Було спроектовано та створено програмну систему. Вона відрізняється простотою у використанні, розумінні та швидкому освоєнні функціоналу, тому навіть людина з мінімальним досвідом зможе її опанувати після ознайомлення з відповідною документацією.

Під час розробки системи використовувалися середовище розробки IDE PhpStorm 2022.1, локальний веб-сервер для Windows (OpenServer) з Nginx+Apach, мова програмування PHP, а також веб-технології HTML, CSS, JavaScript, фреймворк Bootstrap і система керування базами даних MySQL.

Робота пройшла перевірку на унікальність тексту в мережі Інтернет та базі ХНУРЕ (див. додаток А). За результатами кваліфікаційної роботи було підготовлено презентацію (див. додаток Б). Лістинг програмного коду наведено в додатку В.

1 АНАЛІЗ ПРОБЛЕМНОЇ ГАЛУЗІ

1.1 Аналіз проблемної галузі

Дослідження сфери, як важливий етап, дозволяє збагатити розуміння процесів та взаємодій у певній галузі. Особливу увагу слід звернути на розвиток та застосування інформаційних технологій у цій області, оскільки саме вони допомагають вирішувати виникаючі проблеми та вдосконалювати процеси за допомогою програмних систем.

Користь від оренди автомобілів полягає в доступності транспортного засобу для подорожей та вирішення різних завдань без необхідності власності на авто. Вона дозволяє ефективно використовувати час та зручно переміщатися, особливо у випадках, коли власний автомобіль відсутній або не може бути використаний. Крім того, оренда автомобіля дозволяє економити на витратах на обслуговування, страхування та ремонт транспортного засобу, що може бути вигідним для користувачів, які рідко використовують автомобіль.

Створення програмної системи для підтримки оренди автомобілів є надзвичайно важливим у сучасному житті з ряду причин, які охоплюють зручність, доступність, ефективність та безпеку.

Сьогоднішні люди цінують зручність і доступність у всіх аспектах їхнього життя, включаючи транспортування. Програмна система для оренди автомобілів надає можливість швидко та легко знайти та забронювати автомобіль за допомогою мобільного пристрою або комп'ютера. Користувачі можуть вибирати автомобіль з різних категорій, переглядати детальну інформацію та характеристики, а також здійснювати оплату онлайн, що забезпечує максимальну зручність та ефективність у використанні послуги.

Програмна система для оренди автомобілів дозволяє оптимізувати процес оренди та управління автопарком для як клієнтів, так і компаній, які надають ці послуги. Зокрема, вона дозволяє автоматизувати бронювання, обробку платежів, управління запасами автомобілів та маршрутами, що дозволяє підвищити продуктивність та знизити витрати для обох сторін. Крім того, система може

надати аналітичні звіти та дані, які допомагають у прийнятті рішень для подальшого розвитку бізнесу.

Однією з головних переваг програмної системи для оренди автомобілів є забезпечення безпеки та захисту як для клієнтів, так і для компаній. За допомогою цифрового підтвердження та автоматизованої системи адміністрування, ризик помилок та шахрайства значно знижується. Крім того, інтеграція геолокаційних систем та інших технологій дозволяє відстежувати місцезнаходження автомобіля, що забезпечує безпеку та контроль за ним.

Отже, створення програмної системи для оренди автомобілів дозволяє підвищити зручність, ефективність та безпеку використання послуг оренди автомобілів, що робить його незамінним і важливим компонентом сучасного життя.

А щоб програмна система була ефективною та корисною для користувачів, розробникові потрібно вирішити кілька ключових питань:

- вибір оптимальної архітектури системи, яка забезпечить швидку та ефективну роботу;
- вибір підходящої бази даних для забезпечення надійного зберігання та доступу до інформації;
- розробка алгоритму для ефективного пошуку та підбору книг за допомогою набору ключових слів;
- створення зручного та інтуїтивно зрозумілого інтерфейсу користувача;
- забезпечення безпеки та конфіденційності даних, що зберігаються в системі.

Архітектура програмного забезпечення визначає структуру програми чи обчислювальної системи, включаючи компоненти, їх властивості та взаємозв'язки. Основною метою архітектури є спрощення системи шляхом розділення функцій на відокремлені модулі. Існує різноманітні архітектурні стилі, які пропонують різні підходи до побудови системи в залежності від її концепції. При виборі архітектурного стилю важливо звернути увагу на масштабованість та гнучкість системи, щоб зміни можна було внести без великих труднощів та затрат

часу. З цієї точки зору, архітектура "клієнт-сервер" є оптимальним варіантом, де логіка системи розташована на сервері, а клієнтська частина відповідає за інтерфейс користувача.

Сервер отримуватиме запити від клієнтської частини, взаємодіятиме з базою даних та виконуватиме всі алгоритмічні операції. Зазвичай, веб-інтерфейс буде надсилати HTTP-запити на сервер і відображати отримані відповіді за допомогою UI/UX елементів. При виборі методу збереження даних важливо врахувати всі потреби системи та обсяг інформації, який вона буде зберігати. Ураховуючи переваги та недоліки різновидів баз даних, найбільш оптимальним вибором є реляційна база даних. Оскільки цей тип бази даних побудований на основі реляційної моделі, тобто дані зберігаються у вигляді таблиць. Проектування такої бази даних включає три етапи: концептуальний, логічний та фізичний.

Концептуальне проектування - це перший етап процесу проектування бази даних, під час якого визначаються загальні вимоги до системи та описуються її концептуальні засади без уточнення технічних деталей. Основна мета концептуального проектування - визначення основних сутностей, атрибутів та взаємозв'язків між ними на високому рівні абстракції. Цей етап допомагає зрозуміти бізнес-потреби та логіку взаємодії з даними без врахування технічних деталей реалізації. В результаті концептуального проектування може бути створена модель сутностей-зв'язків (ER-модель), яка відображає основні концепції та взаємозв'язки в інформаційній системі.

Логічне проектування - це другий етап процесу проектування бази даних, під час якого концептуальна модель перетворюється в структуровану форму, що враховує специфічні вимоги до системи. На цьому етапі розробляються структури даних, визначаються типи даних для атрибутів, ідентифікатори та зовнішні ключі, а також встановлюються зв'язки між таблицями. Логічне проектування враховує технічні обмеження та можливості конкретної системи керування базами даних (СКБД), але ще не залежить від конкретної реалізації СКБД. Результатом логічного проектування є набір таблиць з відповідними атрибутами та зв'язками

між ними, а також визначення правил цілісності даних. Обов'язковою умовою є нормалізація реляційної моделі даних. Нормалізація полягає у видаленні дублювання даних та встановленні функціональних зв'язків між ними. Кожен етап нормалізації спрямований на перетворення схеми відношень у нормальні форми, які визначають вимоги до відношень. Зазвичай процес нормалізації включає досягнення третьої нормальної форми, кожна з яких має свої правила та обмеження.

Перша нормальна форма (1NF) вимагає, щоб кожне значення у відношенні було атомарним, тобто на перетині кожного стовпця та запису був лише один елемент даних.

Друга нормальна форма (2NF) передбачає, що відношення повинно бути у першій нормальній формі і кожен неключовий атрибут повністю залежить від первинного ключа відношення.

Третя нормальна форма (3NF) вимагає, щоб відношення було у другій нормальній формі, а також щоб між неключовими атрибутами не було транзитивних залежностей.

Фізичне проектування - це останній етап процесу проектування бази даних, під час якого логічна модель перетворюється в конкретну структуру бази даних, яка враховує технічні особливості обраної СКБД. На цьому етапі визначається фізичне розміщення даних на диску, вибираються оптимальні типи даних та індексація, розробляються стратегії забезпечення ефективності та надійності бази даних. Фізичне проектування включає в себе також розробку скриптів для створення таблиць, індексів, переглядів та інших об'єктів бази даних у СКБД. Результатом фізичного проектування є готова до реалізації база даних, яка може бути розгорнута та використана у відповідності з вимогами та обмеженнями проекту.

Для створення інтерфейсу користувача важливо керуватися рекомендаціями з щодо комфорту та зручності використання веб-системи, дотримуватися принципів розробки інтерфейсів та дизайну UI/UX. Оскільки архітектура системи передбачає взаємодію через "клієнт-сервер", інтерфейс буде реалізований у складі

"клієнтської" частини, що функціонує як окремий додаток з власною логікою. Крім того, важливо враховувати можливості фреймворків та бібліотек, які будуть використовуватися для реалізації "клієнтської" частини, оскільки кожен з них має власні обмеження та правила використання.

Забезпечення безпеки даних є критичним аспектом розробки системи. Існує багато способів забезпечення безпечного обміну та збереження інформації. Для максимально надійного захисту рекомендується використовувати протокол HTTPS, який гарантує безпечну передачу даних. Крім того, всі паролі користувачів повинні бути збережені у базі даних у захешованому вигляді, що значно підвищить безпеку системи.

1.2 Аналіз існуючих аналогів

На сьогоднішній час стає все більше і більше можливості щодо придбання автомобіля в оренду, а з тим і конкуренція на ринку також зростає. Користувач, шукаючи вигідну ціну, вже не потребує великих зусиль для цього.

Тож для створення додатку було проведено дослідження декількох веб-додатків по оренді автомобілів в Україні:

- Укр-Прокат [1];
- RENTAL.UA [2].

«Укр-Прокат» [1] на Головній сторінці сайту показує велику кількість автомобілів різних марок, що підлягають оренді та експлуатації. Автомобілі можна фільтрувати за категоріями (див. рис. 1).

Головним недоліком є оптимізація. При натисканні кнопки «Бронюй», досить довго завантажується інформація про бронювання.

Також на сайті представлений, власне, сам список автомобілів із зображенням їх зовнішнього вигляду, що є досить зручним. Переходячи на сторінку з описом будь-якого автомобілю можна побачити детальну інформацію про обрану машину (див. рис.2).

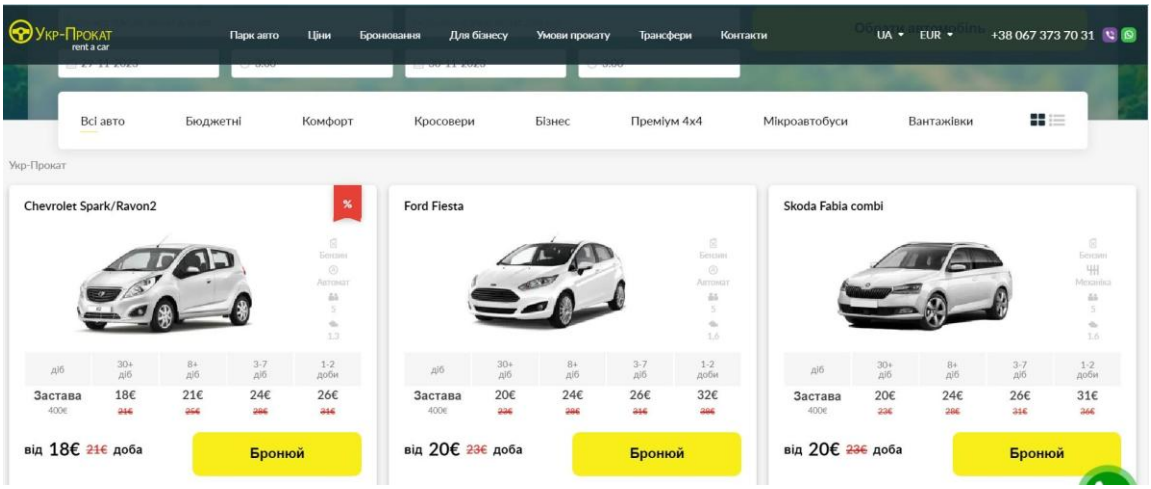


Рисунок 1 – Головна сторінка сайту «Укр-Прокат» (за даними [1])

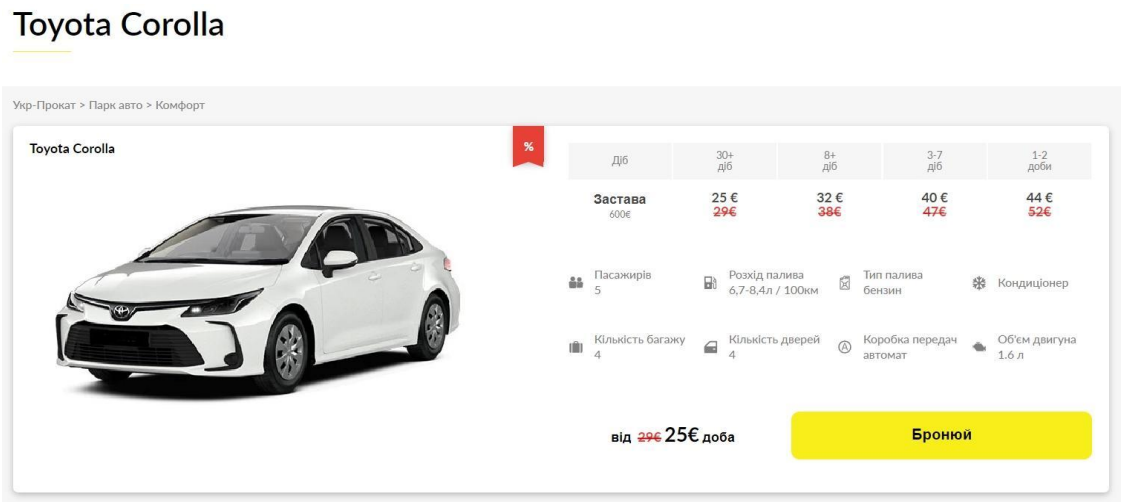


Рисунок 2 – Детальна інформація про автомобіль на сайті «Укр-Прокат» (за даними [1])

Розглянемо веб-додаток «RENTAL.UA» [2].

Представлення автомобілів в дуже якісних зображеннях та розташовані в слайдерах, що дуже зручно для їх перегляду. Що до детальної інформації, то тут вже досить багато опису про автомобіль та додано значну кількість фото даного автомобіля, це дає клієнту більш розгорнуту інформацію про автомобіль (див. рис.3).

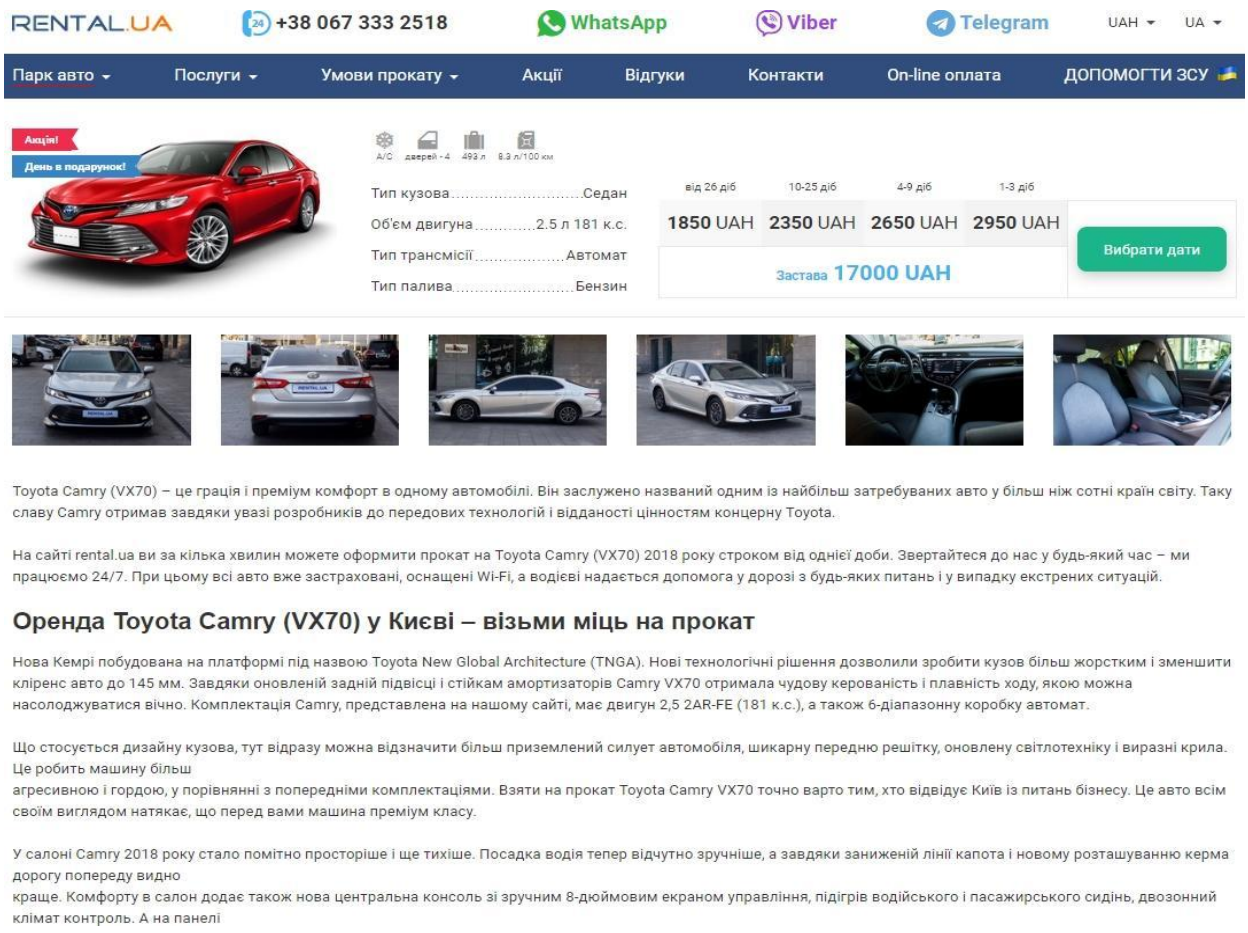


Рисунок 3 – Детальна інформація автомобіля (за даними [2])

Сайт просто всипаний соц. мережами, месенджерами, телефонами, навіть чат бот присутній. Для клієнтів - це дуже зручний спосіб, чим швидше замовити автомобіль.

Тож, після перегляду й аналізу сайтів-конкурентів по оренді автомобілів можна зробити висновок, що більшість додатків не реалізовують функціонал для безпосередньої оренди, а використовуються, як інформативні сторінки.

Головні функції, що повинні бути реалізовані при розробці проекту:

- оренда автомобіля на сайті;
- реалізація фільтру для пошуку;
- можливість реєстрації користувачів;
- динамічна зміна інформації на сайті.

1.3 Постановка задачі

Метою роботи є розробка програмної системи для оренди автомобілів. Така система допоможе швидко та зручно орендувати автомобіль в будь якому місці (де є наявність інтернету).

Необхідно спроектувати базу даних для збереження інформацію стосовно:

- укладання договорів на здачу автомобілів в оренду;
- обслуговування пошукових запитів (пошук автомобіля за базою даних (марка, державний номер, рік випуску, ціна за день), пошук орендарів (прізвище, номер тел, email, номер паспорту);
- система повинна відображати данні безпосередньо про основні поняття ПЗ: автомобілі, менеджерів, клієнтів, договори між менеджерами та клієнтами та ДТП автомобілів;
- система повинна підтримувати сортування, пошук та фільтрацію даних:
    1) сортувати клієнтів за прізвищем, автомобілі за маркою, договори за датою укладання;
    2) здійснювати пошук інформації про клієнта за його повним ПІБ, телефоном; про автомобіль за його державним номером, маркою, роком випуску;
    3) здійснювати фільтрацію інформації за марками авто;
- система повинна підтримувати додавання нових даних про автомобілі, менеджерів, клієнтів, договорів та дтп;
- система повинна підтримувати можливості редагування інформації про автомобілі, менеджерів, клієнтів, договорів та дтп;
- система повинна підтримувати можливості вилучення інформації про автомобілі, менеджерів, клієнтів, договорів та дтп з підтримкою режиму підтвердження користувачем видалення інформації про поточний об’єкт;
- система повинна підтримувати виконання наступних часто виникаючих запити до БД:

1) отримати список клієнтів, які користувались сервісом N або більше ніж N разів (ПІБ, Місто, Телефон, Кількість договорів);
2) статистика кількості разів орендованих авто за певний період часу (Автомобіль, Кількість разів);
3) отримати статистику з кількості ДТП для кожного автомобіля (назва авто, кількість ДТП);
4) отримати статистику про кількість клієнтів в розрізі категорії авто (дата, категорія, кількість клієнтів);

- система повинна підтримувати підготовку та друк наступних звітів:
  1) прайс-лист автомобілів та їх ціна оренди на день для клієнтів (марка, рік випуску, вартість на день);
  2) список автомобілів, які користуються найбільшим попитом протягом останнього місяця (марка, кількість контрактів);
- система повинна реалізовувати наступні задачі автоматизації:
  1) імпорт даних про автомобілі з файлу .xls;
  2) автоматична зміна статусу автомобіля в залежності від стану договору.

Система створюється для обслуговування наступних груп користувачів:

- співробітники, які обслуговують замовлення та клієнтів;
- клієнти (для оренди автомобілів).

Використовуючи вищезгадану інформацію, постає такий список завдань, які слід виконати:

- провести аналіз та моделювання предметної області програмної системи;
- спроектувати базу даних для збереження інформації з предметної області;
- розробити алгоритми підтримки оренди автомобіля на основних її етапах;
- спроектувати архітектуру програмної системи;
- виконати програмну реалізацію серверної та клієнтської частин системи та провести тестування створенного програмного продукту.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Формування вимог до програмної системи є критично важливим процесом, який визначає успіх проекту. Чітке визначення, аналіз, документування та управління вимогами дозволяє створити якісний продукт, який задовольняє потреби користувачів та відповідає бізнес-цілям. Використання методів збору вимог, таких як інтерв'ю, спостереження та аналіз документів, забезпечує повноту та узгодженість вимог, що є основою для подальшої розробки та впровадження системи.

Вимоги до програмної системи для підтримки оренди автомобілів:

- функціональні вимоги:
    1) система повинна дозволяти користувачам реєструватися та входити до особистого кабінету;
    2) користувачі повинні мати можливість переглядати доступні автомобілі та здійснювати пошук за різними критеріями;
    3) система повинна забезпечувати можливість бронювання автомобіля;
    4) система повинна забезпечувати можливість створення статистик;
    5) система повинна забезпечувати можливість створення звітів та їх імпорту;
    6) система повинна забезпечувати автоматичну зміну статусу автомобіля в залежності від стану договору;
    7) адміністратори або менеджери повинні мати доступ до управління інформацією про автомобілі та бронювання;
- нефункціональні вимоги:
    1) час відгуку на будь-яку операцію не повинен перевищувати 2 секунд;
    2) усі паролі користувачів повинні зберігатися у хешованому вигляді;
    3) інтерфейс системи повинен бути інтуїтивно зрозумілим та зручним для використання на різних пристроях.

Створення програмної системи для оренди автомобілів включає в себе налаштування середовища розробки, створення бази даних, розробку серверної та клієнтської частин, а також налаштування веб-сервера. Використання HTML, CSS, JavaScript, Framework Bootstrap, PHP, Apach+Nginx та MySQL забезпечить гнучкість та ефективність роботи додатку, що дозволить користувачам зручно орендувати автомобілі.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз та моделювання предметної області

Предметна область «Оренда автомобілів» обґрунтована розвитком комерційних організацій з прокату автомобілів, у яких необхідний комп'ютеризований облік машин. Ця інформація велика і розрізнена. Щоб вести облік всіх автомобілів, які можна взяти в прокат, в організації є потреба у структуруванні даних про автомобілі. Відсутність такої можливості призводить до проблеми втрати даних та великих часових витрат на вибірку даних.

Предметною областю є підприємство, спеціалізація якого – прокат автомобілів (або оренда автомобілів). Процедура оренди автомобіля формалізована. Клієнт повинен бути дорослiше 21 року. Він повинен пред'явити паспорт і міжнародне водійське посвідчення працівнику сервісу прокату. До моменту укладення договору посвідчення має бути не менше 2 років.

У вартість орендної плати має входити:

- необмежений пробіг автомобіля;
- доставка клієнту автомобіля в межах міста (послуга надається в унікальних випадках);
- ремонт або заміна автомобіля у разі технічної несправності, крім пошкодження покришок та вітрового скла; повне страхування на випадок ДТП, що сталося не з вини клієнта; страховка, що покриває збитки, завдані автомобілю в ДТП з вини клієнта, понад певну суму (але якщо на момент ДТП водій перебував у стані алкогольного сп'яніння, страховка не виплачується); страховка пасажирів (крім водія) від нещасних випадків (себе водій може застрахувати за додаткову оплату);
- податки.

Зазвичай машину доставляють з повним баком, але й повернути до офісу орендодавця її потрібно також з повним баком. Оренду автомобіля у місці відпочинку можна замовити ще під час купівлі туру у своєму агентстві, включивши її до пакету послуг. Багато агентств пропонують це своїм клієнтам,

оскільки замовлення оренди машини у своїй агенції полегшує туристу проблеми з прочитання договору на оренду іноземною мовою, гарантує надання якісних та зрозумілих послуг договором на купівлю туру загалом. Автомобілі застраховані від усіх ризиків на умовах КАСКО та ОСАГО. У разі ДТП відповідальність Клієнта становить величину застави – 200$, всі інші збитки, завдані автомобілю, покриває страхова компанія.

Процес оформлення оренди автомобіля простий та зрозумілий. Від Вас вимагається лише мінімальний комплект документів (паспорт та посвідчення водія) для оформлення документів оренди.

Між підприємством та клієнтом складається договір. У договорі обов'язково мають бути вказані такі дані:

- інформація про марку автомобіля;
- державний реєстраційний номер;
- номер двигуна;
- умови, терміни і порядок внесення плати за оренду;
- права орендодавця і орендаря;
- інформація про страхування;
- територія використання автомобіля;
- інформація про обмеження пробігу;
- наявність додаткових послуг;
- відповідальність за неналежне виконання умов оренди;
- умови продовження оренди;
- інша інформація.

Відповідно до предметної області система будується з урахуванням таких особливостей:

- кожен автомобіль здається у межах контракту;
- автомобіль може бути орендований одним клієнтом;
- один договір створюється на кожну угоду про аренду;
- договір оформляється на одного клієнта;
- кількість замовлень, які зробив клієнт, впливає на ціну угоди;

- кожен автомобіль випущений певною фірмою;
- автомобіль кожної фірми має певну ціну протягом дня оренди;

Додатково до договору оформляється акт прийому-передачі. Там вказуються характеристики переданого в оренду автомобіля, його пробіг, вартість, а також технічний стан. У цьому акті фіксуються недоліки, наявні в машині на момент її здачі в оренду.

Орендодавець перед укладенням договору повинен ознайомити орендаря з правилами використання автомобіля. Крім того, він зобов'язаний у присутності замовника послуги перевірити справність машини.

Капітальний і поточний ремонт автомобіля повинен здійснюватися орендодавцем. Це повинно бути відображено в договорі.

Є декілька сервісів підприємства, в яких можна подивитися та забрати автомобілі. Також є гаряча лінія, за допомогою якої можна отримати консультацію щодо автомобілів, особливостей договору та інше.

Існує межа кількості автомобілів, якими може оперувати авто-прокат, приблизно 50-100.

Першим етапом є розробка сценаріїв використання системи різними категоріями користувачів. Для моделювання цих сценаріїв зазвичай використовується UML, що є стандартом для створення моделей, що описують об'єкти. У якості мови моделювання, UML має свої правила форматування та синтаксис. Використовуючи графічну нотацію UML, можна візуалізувати структуру системи, об'єднати всі її компоненти в єдину модель та уточнювати її у процесі розробки. Отже, діаграма UML, що відображає дії та сценарії взаємодії користувача з додатком або програмою з метою виконання певних дій та досягнення конкретних цілей, є діаграмою Use Case.

В програмній системі існують три ролі: незареєстрований користувач (майбутній клієнт), адміністратор та менеджер.

Незареєстрований в додатку користувач, зможе переглядати тільки інформацію про автомобілі, а після обрання якогось із них, він також зможе

заповнити необхідною інформацією форму, яка відправить його дані на email компанії або зателефонувати менеджеру за телефоном указаному в додатку.

Відмінність між адміністратором та менеджером полягає втому, що Адміністратор має можливість додавати/редагувати/видаляти інформацію про менеджерів, а Менеджер такої можливості не має.

На рисунку (див. рис. 4) наведено опис функціонального призначення ІС, що створюється.

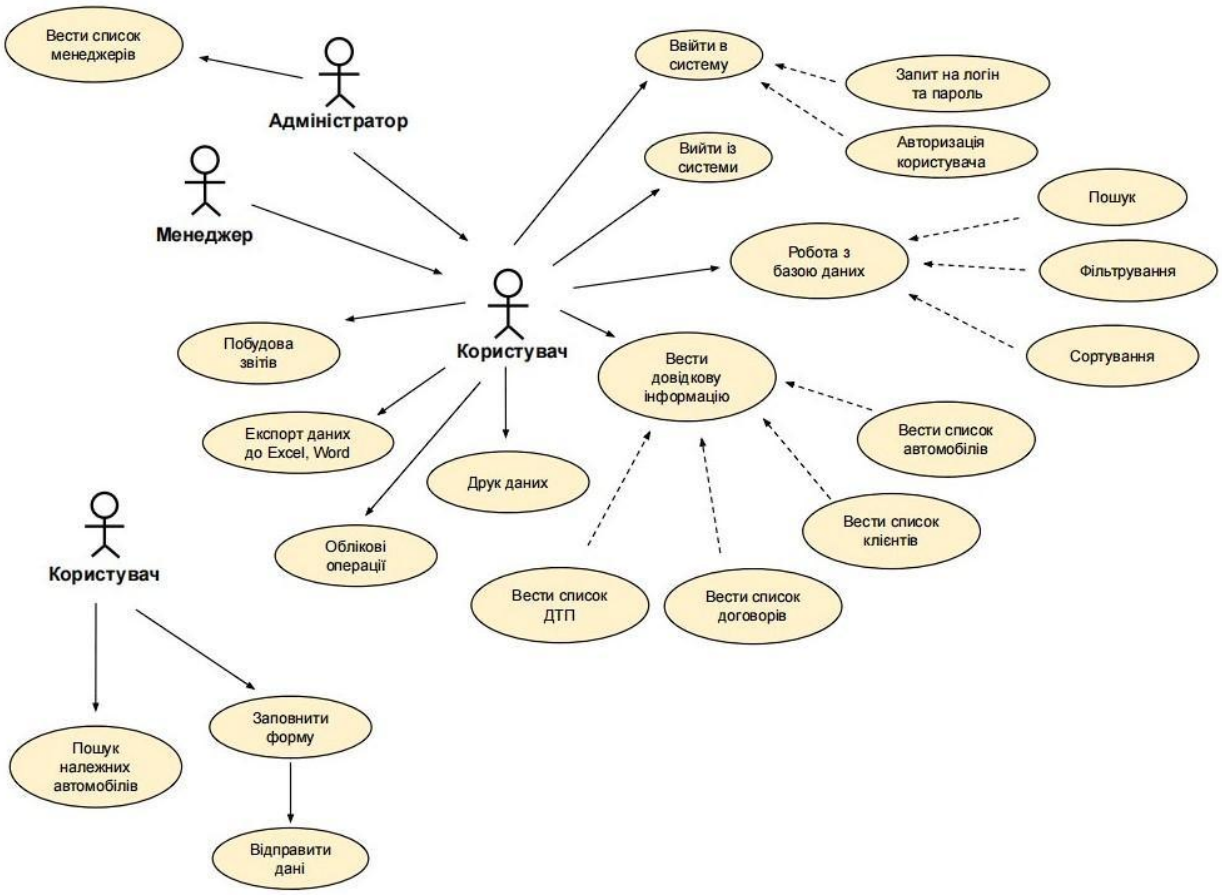


Рисунок 4 – USE-CASE діаграма (рисунок виконано самостійно)

В предметній області ми можемо виділити два користувача (Менеджер/Адміністаратор та Клієнт), відповідно матимемо 5 сутностей, такі як:

- сутність Client (Клієнт) – містить в собі інформацію про клієнта, який бере автомобіль в оренду;

- сутність Manager (Менеджер) – містить в собі інформацію про менеджера, який буде заключати договір з клієнтом по даному автомобілю;
- сутність Car (Автомобіль) – містить в собі інформацію про автомобіль;
- сутність Contract (Договір) – містить в собі інформацію про договір, укладеним між клієнтом та менеджером по даному автомобілю;
- сутність Accident (Штрафи) – містить в собі штрафи по даному автомобілю.

Опис концептів програмної області, їх властивостей та зв'язків між ними зобразимо у вигляді загальної діаграми класів представлено на рисунку (див. рис. 5).

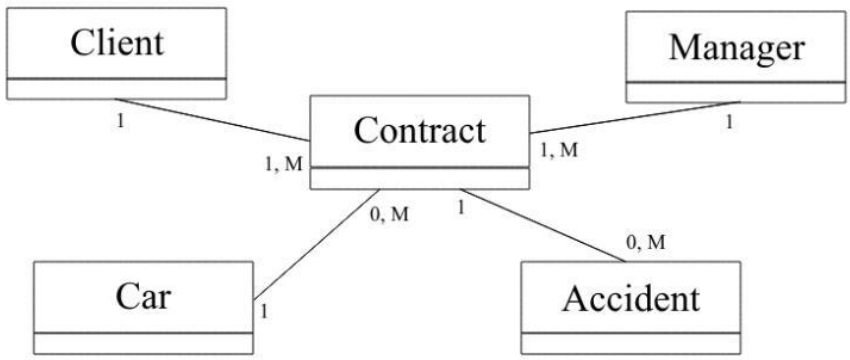


Рисунок 5 – Загальна діаграма класів (рисунок виконано самостійно)

Отже з діаграми класів ми можемо побачити 5 сутностей, які взаємодіють між собою різними зв’язками. Також ці сутності мають перелік властивостей:

- сутність Client (Клієнт) – client_id, surname, name, patronymic, phone, email, day, month, year, city, address, passport, driver_license, experience;
- сутність Manager (Менеджер) – manager_id, login, password, surname, name, patronymic, phone;
- сутність Car (Автомобіль) – car_id, state_number, brand, year_of_manufacture, body_number, cost_per_day, photo;
- сутність Contract (Договір) – contract_id, start_date, end_date, sum, manager_id, client_id, car_id;

- сутність Accident (Штрафи) – accident_id, date, damage_%, description, contract_id.

Далі визначимо інформаційні потреби користувача. Тож, Клієнт може мати змогу переглядати автомобілі, сортувати, фільтрувати, скористатися пошуком та залишати заявку на оренду автомобіля, якого обрав. В свою чергу, Менеджер має змогу приймати заявки, які надходять з веб-додатку від клієнтів на пошту, додавати, редагувати та видаляти дані, як клієнтів так і автомобілів, договорів та штрафів.

Кожен клієнт, менеджер, автомобіль, договір та штраф ідентифікується відповідно за його унікальним ключем.

Користувач може використовувати також ще й додаткові функції, такі як:

- сортування інформації;
- пошук інформації;
- фільтрація по деяким опціям (наприклад автомобілі за ціною, за роком випуску);
- виведення статистик;
- завантаження звітів в файл формату .xls.

Після укладання договору, клієнту автоматично буде відправлятися лист на пошту про успішне укладання договору та оренду автомобіля на визначений строк, який буде указаний в договорі.

3.2 Проектування бази даних

Було розроблена ER-діаграма за нотацією Баркера (див. рис. 6).

Реляційна база даних – це база даних, заснована на реляційній моделі даних. Реляційна база даних є сукупністю елементів даних, організованих у вигляді набору формально описаних таблиць, з яких дані можуть бути доступними або повторно зібрані багатьма різними способами без необхідності реорганізації таблиць бази даних.

Метою нормалізації є усунення недоліків структури БД, які призводять до шкідливої надмірності в даних, яка в свою чергу потенційно призводить до різних

аномалій і порушень цілісності даних. Кожен рядок у таблиці такої бази даних являє собою запис унікальним ідентифікатором – ключем. Стовпці таблиць мають атрибути даних, що дає змогу встановлювати зв’язки між елементами даних.

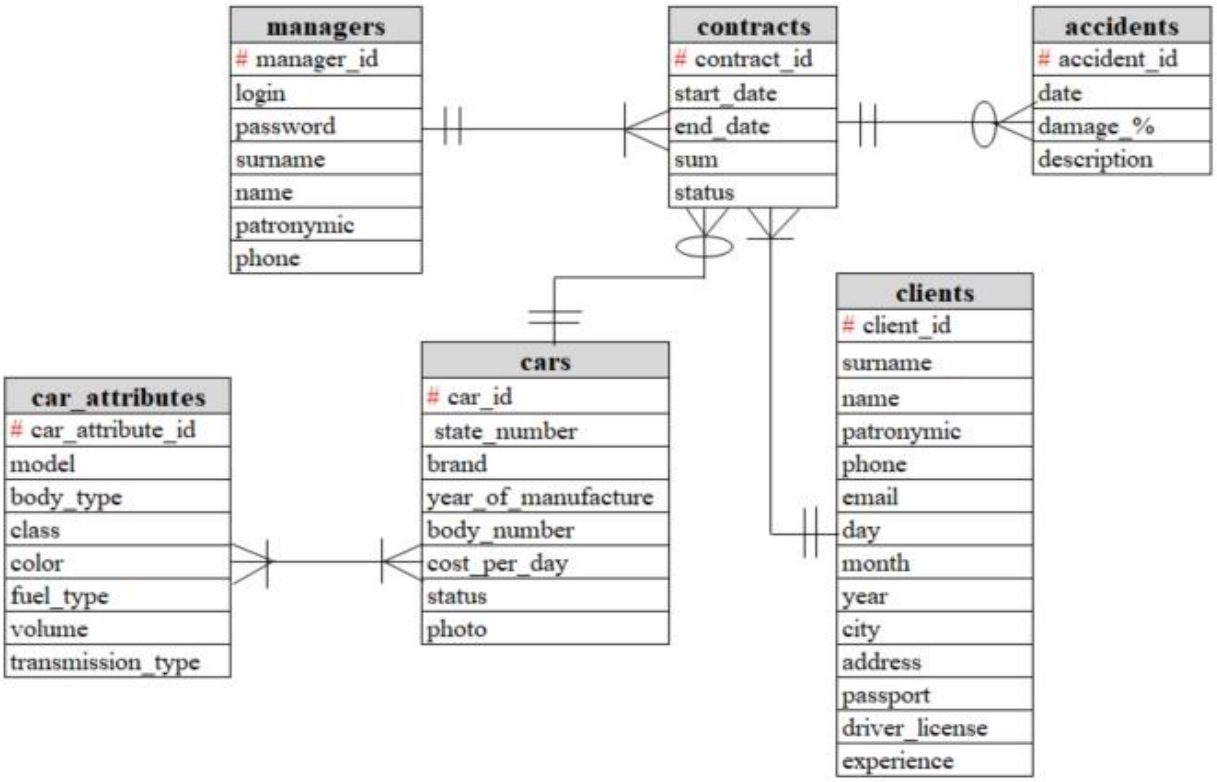


Рисунок 6 – ER-діаграма за нотацією Баркера (рисунок виконано самостійно)

Саме цьому, реляційна модель бази даних є найбільш ефективною для вирішення задачі даного курсового проекту (див. рис. 7).

3.3 Розробка алгоритмів підтримки оренди автомобіля

Розглянемо основні етапи керування контрактом оренди автомобіля:

- реєстрація клієнта: Клієнт надає особисті дані та проходить процес реєстрації;
- бронювання автомобіля: Клієнт вибирає автомобіль, дати оренди та місце отримання;
- підписання контракту: Клієнт підписує договір оренди, де вказуються всі умови оренди;

– отримання автомобіля: Клієнт отримує автомобіль у зазначеному місці;
– використання автомобіля: Клієнт використовує автомобіль протягом зазначеного терміну;
– повернення автомобіля: Клієнт повертає автомобіль у зазначене місце;
– завершення контракту: Підрахунок загальної суми, оплата оренди та закриття контракту.

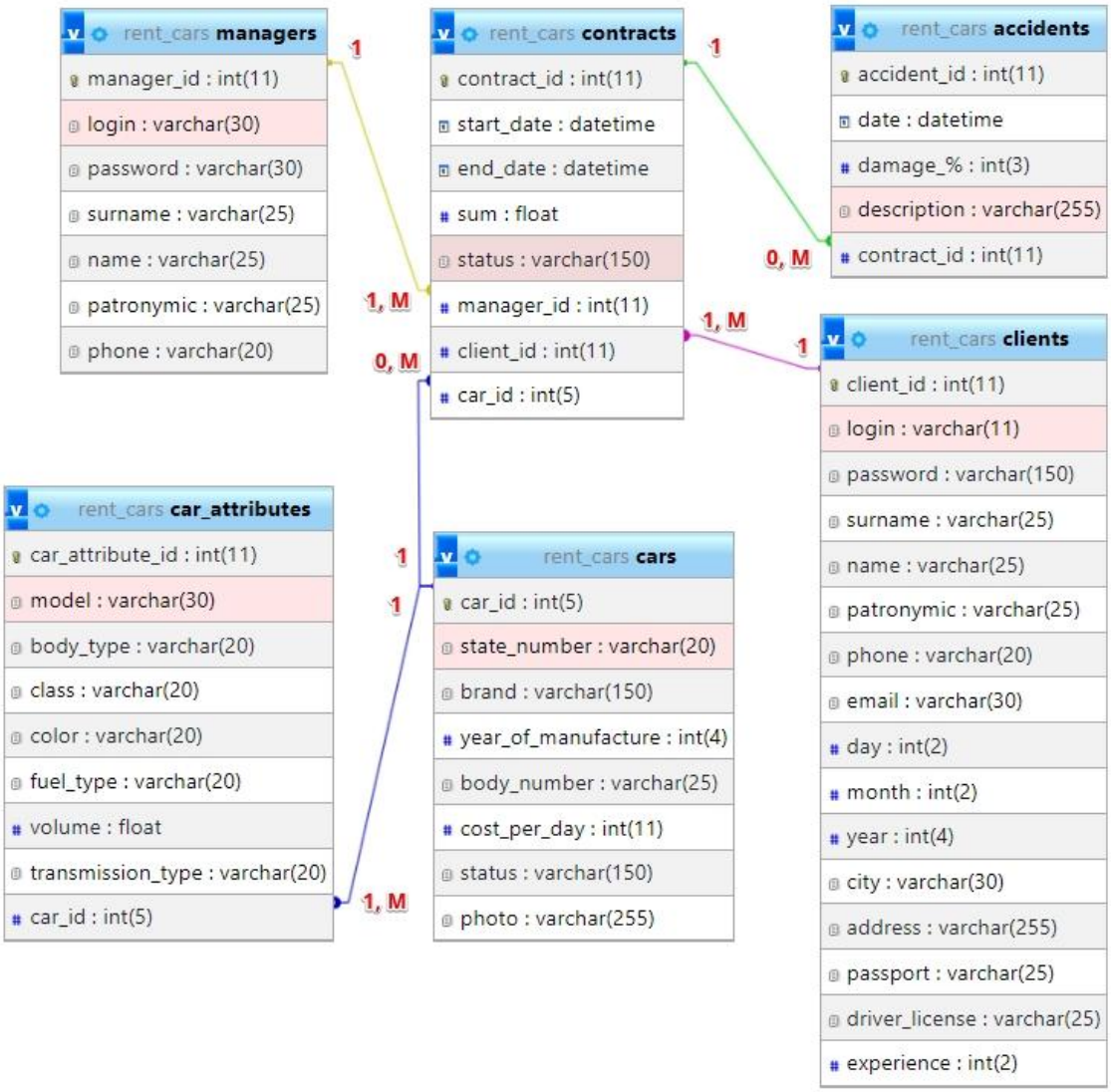


Рисунок 7 – Схема реляційної бази даних (рисунок виконано самостійно)

Зображення нижче показує основні етапи процесу оренди автомобіля (див. рис. 8).

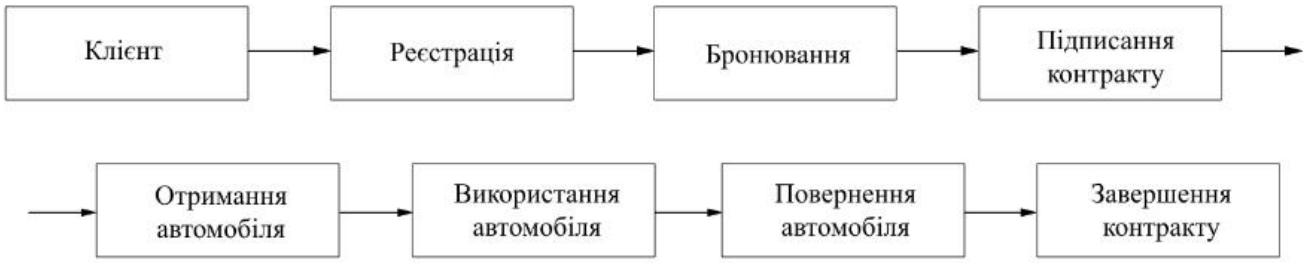


Рисунок 8 – Основні етапи процесу оренди автомобіля (рисунок виконано самостійно)

Тепер опишемо потоки даних для кожного етапу:

- клієнт – початковий етап, де клієнт ініціює процес оренди;
- реєстрація – клієнт реєструється у системі;
- бронювання – після реєстрації клієнт бронює автомобіль;
- підписання контракту – після бронювання клієнт підписує контракт;
- отримання автомобіля – клієнт отримує автомобіль;
- використання автомобіля – клієнт використовує автомобіль протягом орендного періоду;
- повернення автомобіля – клієнт повертає автомобіль;
- завершення контракту – оренда завершується, і контракт закривається.

Діаграма потоків даних (Data Flow Diagram) (див. рис. 9):

- реєстрація:
    1) вхід: Дані клієнта;
    2) вихід: Підтвердження реєстрації;
- бронювання:
    1) вхід: Дані автомобіля, дані клієнта;
    2) вихід: Підтвердження бронювання, деталі бронювання.
- підписання контракту:
    1) вхід: Дані клієнта, дані автомобіля, деталі бронювання;
    2) вихід: Підписаний контракт, дані про оплату.

– отримання автомобіля:
    1) вхід: Підписаний контракт;
    2) вихід: Передача автомобіля, звіт про стан автомобіля.
– використання автомобіля:
    1) вхід: Автомобіль, дані про оренду;
    2) вихід: Дані про використання (пробіг, стан автомобіля).
– повернення автомобіля:
    1) вхід: Автомобіль, дані про використання;
    2) вихід: Звіт про стан автомобіля, дані про завершення оренди.
– завершення контракту:
    1) вхід: Звіт про стан автомобіля, дані про завершення оренди;
    2) вихід: Завершений контракт, фінальні розрахунки.

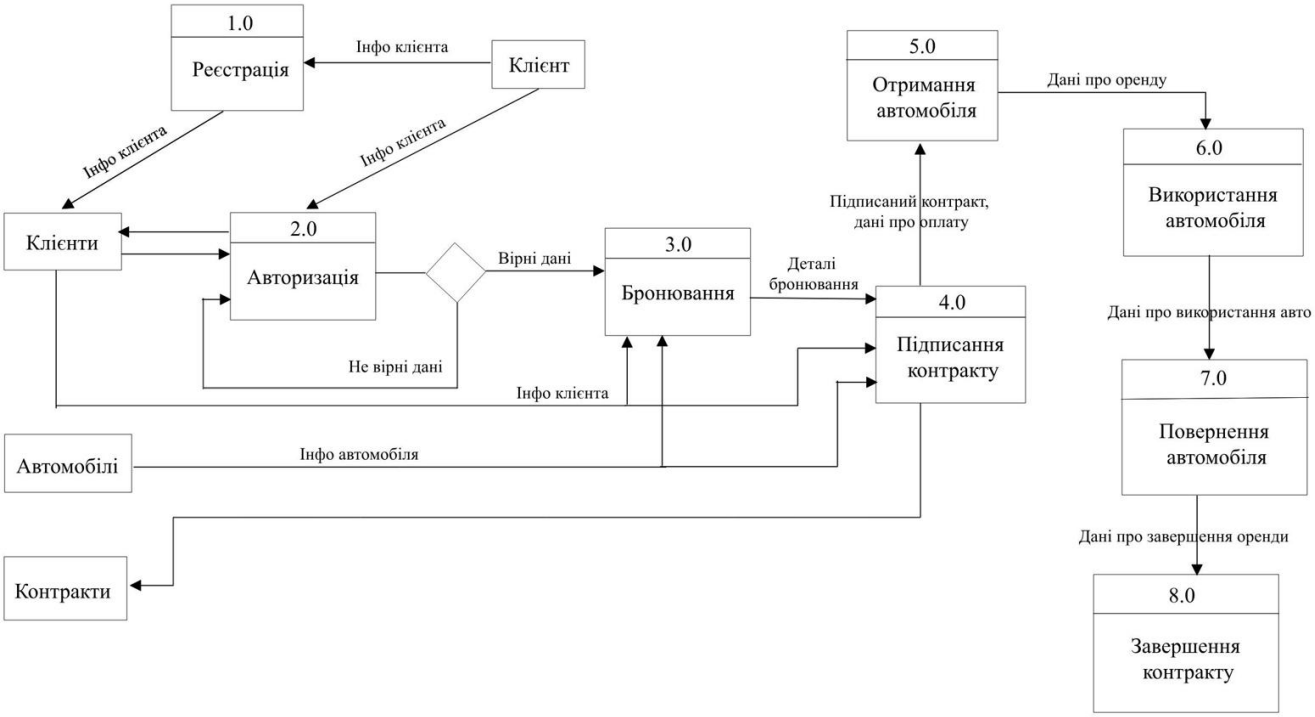


Рисунок 9 – Діаграма потоків даних (DFD) (рисунок виконано самостійно)

В даній програмній системі є реактивні об'єкти предметної області, для яких можна промоделювати зміну їх станів:

– створений (початковий стан);

– в очікуванні оплати;
– активний / скасований;
– завершений.

Отже "Створений (початковий стан)" є початковим станом договору, якщо він ще не був створений. При створенні нового договору, він може переходити у стан "В очікуванні оплати" або, наприклад, "Активний", залежно від дій користувача або інших факторів. Договір може перейти в стан "Завершений" лише після того, як він буде в статусі "Активний". Статус "Скасований" може бути застосований, якщо клієнт не підписав та/або не оплатив протягом зазначеного терміну.

Відповідно статусів ми також маємо діаграму станів договору (див. рис. 10)

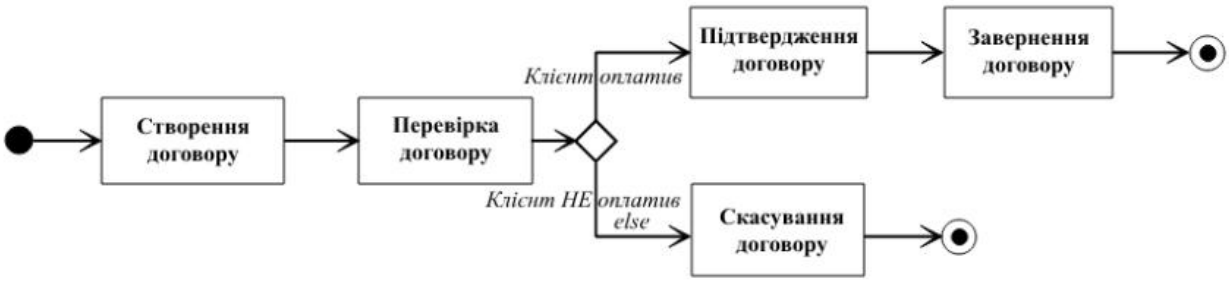


Рисунок 10 – Діаграма станів договору (рисунок виконано самостійно)

3.4 Розробка архітектури системи

Архітектура програмного забезпечення – це метод структуризації системи та її компонентів на конкретному етапі функціонування. У межах цієї програмної системи було обрано клієнт-серверну архітектуру (див. рис. 11), де клієнтом виступає веб-застосунок, доступ до якого можна отримати за допомогою будь-якого сучасного браузера.

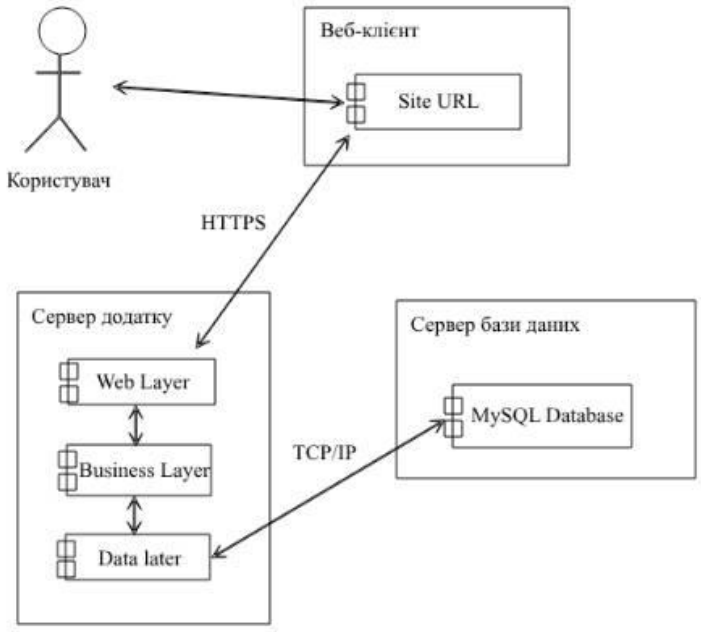


Рисунок 11 – Загальна архітектура системи (рисунок виконано самостійно)

Треба відмітити, що на сервері БД будуть зберігатися таблиці та буде реалізована певна логіка у вигляді тригерів.

3.5 Створення UI / UX або іншого дизайну системи

UI/UX дизайн відіграє ключову роль у розробці успішного програмної системи для оренди автомобілів. Від ефективності та зручності інтерфейсу користувача залежить загальне враження користувачів від сервісу, їхня задоволеність та, як наслідок, лояльність до продукту. У цьому розділі розглянемо етапи створення UI/UX дизайну, що сприяють забезпеченню інтуїтивно зрозумілого та привабливого користувацького досвіду.

Нижче зображена схема навігації програмної системи (див. рис. 12).

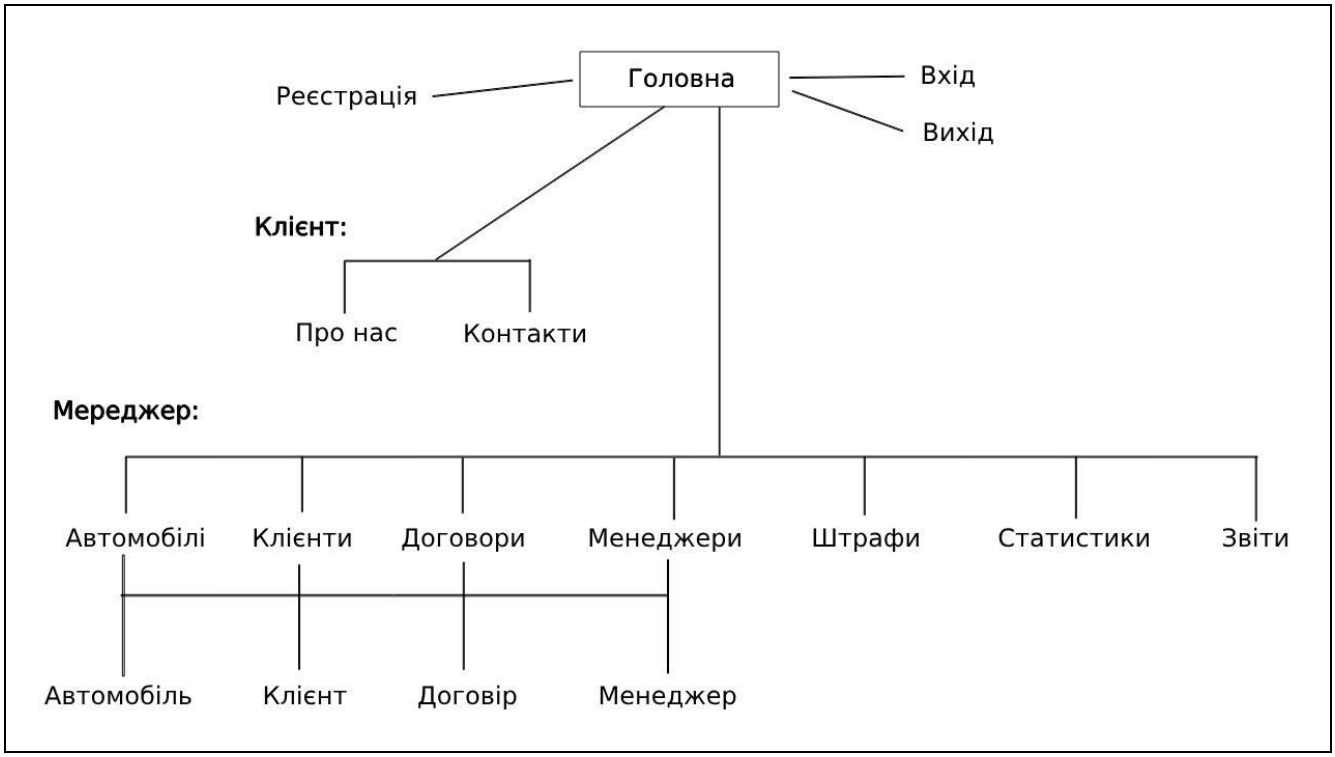


Рисунок 12 – Схема навігації програмної системи (рисунок виконано самостійно)

Головна сторінка веб-застосунку для оренди автомобілів повинна бути зрозумілою, інтуїтивно простою та привабливою. Основні елементи макету включають:

- шапка (Header);
- навігаційне меню (Menu);
- основний контент (Content);
- футер (Footer).

На рисунку нижче наведено макет головної сторінки (див. рис. 13).

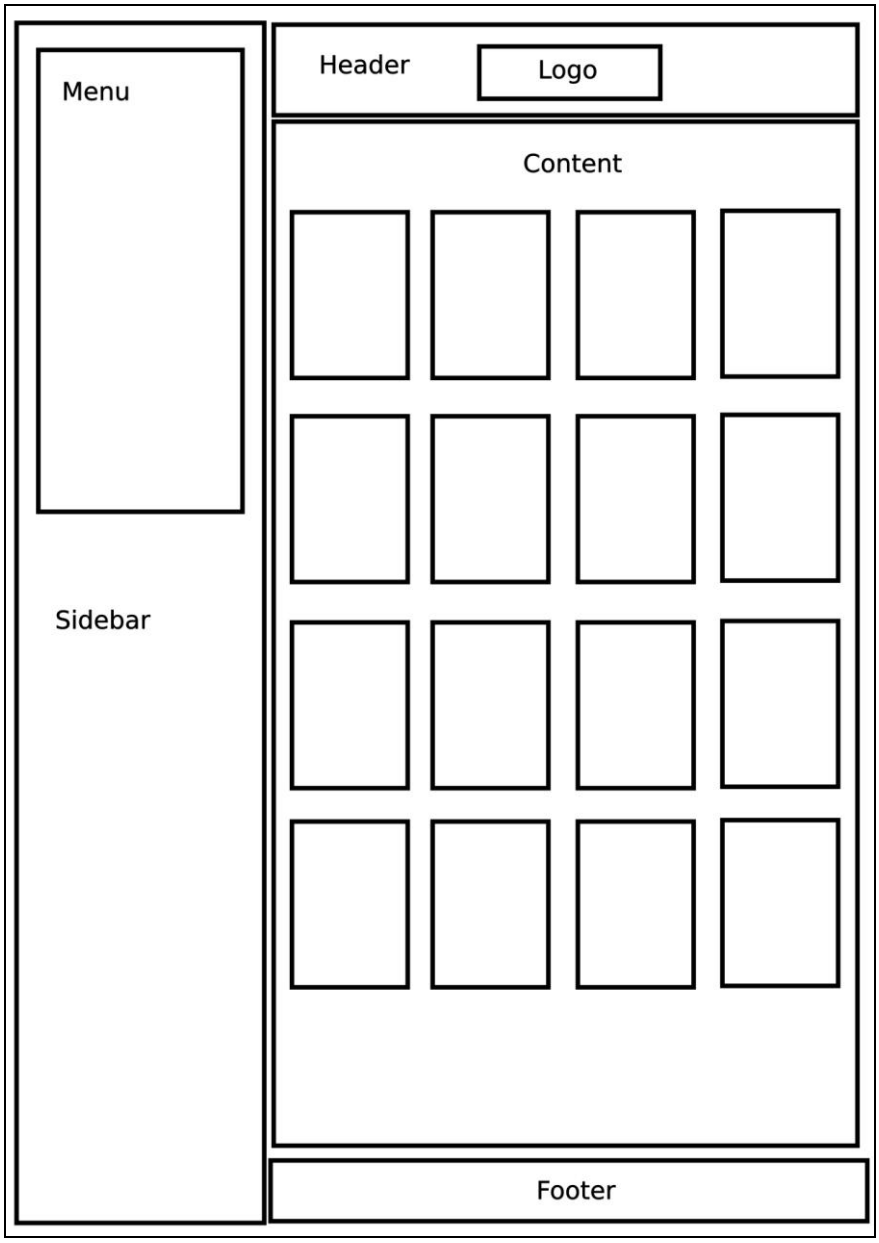


Рисунок 13 – Макет Головної сторінки (рисунок виконано самостійно)

На основі прийнятих проектних рішень можна переходити до програмної реалізації.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Вибір засобів програмної реалізації

Для розробки було використано середу розробки IDE PhpStorm 2022.1 та мову програмування PHP. Для створення візуального представлення програми було обрано веб технології HTML, CSS, JavaScript та Framework Bootstrap. Даний інтерфейс дозволяє швидко створити зручний та інтуїтивно зрозумілий додаток.

Додаток націлено на взаємодію із базами даних, отже для їх створення було обрано систему управління реляційними базами даних (MySQL). Саме ця СКБД була використана через її здатність швидкої обробки інформації, легкість використання та інтегрованість з іншими продуктами.

В розробці даного додатку я використовував саме OpenServer - це портативне програмне середовище, створене спеціально для веб-розробників з урахуванням їх рекомендацій та побажань.

Даний програмний комплекс включає ретельно підібраний набір серверного програмного забезпечення, а також неймовірно зручну і продуману керуючу утиліту, яка володіє потужними можливостями з адміністрування і налаштування всіх доступних компонентів.

OSPanel широко використовується з метою розробки, налагодження та тестування веб-проектів, а також надання веб-сервісів у локальних мережах.

4.2 Описання фізичної моделі бази даних

При реалізації фізичної моделі бази даних було створено шість таблиць. Нижче наведено команди на їх створення.

Розглянемо деякі приклади створення таблиць:

```
--
-- Структура таблиці `cars`
--

CREATE TABLE `cars` (
  `car_id` int(5) NOT NULL,
  `state_number` varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL,
  `brand` varchar(150) COLLATE utf8mb4_unicode_ci NOT NULL,
```

```
  `year_of_manufacture` int(4) NOT NULL,
  `body_number` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `cost_per_day` int(11) NOT NULL,
  `status` varchar(150) COLLATE utf8mb4_unicode_ci NOT NULL,
  `photo` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;



--
-- Структура таблиці `clients`
--


CREATE TABLE `clients` (
  `client_id` int(11) NOT NULL,
  `surname` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `name` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `patronymic` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `phone` varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL,
  `email` varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL,
  `day` int(2) NOT NULL,
  `month` int(2) NOT NULL,
  `year` year(4) NOT NULL,
  `city` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `address` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `passport` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `driver_license` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `experience` int(2) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;


-- --------------------------------------------------------


--
-- Структура таблиці `managers`
--


CREATE TABLE `managers` (
  `manager_id` int(11) NOT NULL,
  `login` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `password` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `surname` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `name` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `patronymic` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `phone` varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

4.3 Опис серверної частини системи

У даному розділі описано серверну частину розробленої програмної системи. Серверна частина виконує ключові функції з обробки даних, забезпечення безпеки та ефективної взаємодії з клієнтською частиною. Її правильне проектування та реалізація є критичними для загальної продуктивності та надійності системи.

Нижче наведено приклад запиту на створення таблиці contracts з атрибутом status, з яким ми далі будемо працювати:

```
CREATE TABLE `contracts` (
  `contract_id` int(11) NOT NULL,
  `start_date` datetime NOT NULL,
  `end_date` datetime NOT NULL,
  `sum` float NOT NULL,
  `status` varchar(150) COLLATE utf8mb4_unicode_ci NOT NULL,
  `manager_id` int(11) NOT NULL,
  `client_id` int(11) NOT NULL,
  `car_id` int(5) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Розглянемо створення тригерів, що складають серверну частину системи.

Створено тригер на зміну статусу автомобіля в залежності від зміни статусу договору. В цьому тригері ми створюємо обробник після оновлення таблиці contracts. При кожному оновленні договору тригер перевіряє новий статус договору і встановлює відповідний статус автомобіля на основі цього статусу. Потім він оновлює статус автомобіля в таблиці cars.

Створено тригер на запис початкового стану Створено в статус контракту, при додаванні контракту. Цей тригер активується перед вставкою нового рядка у таблицю contracts. При цьому він встановлює значення стовпця status для нового рядка на "created', що відповідає початковому статусу "Створений (початковий стан) ". Приклад реалізації даного тригеру:

```
DELIMITER $$
CREATE TRIGGER set_initial_contract_status
BEFORE INSERT ON contracts
```

```
FOR EACH ROW
BEGIN
    SET NEW.status = 'created';
END;
$$
DELIMITER ;
```

Наступний тринер створено для перевірки зміни статусу при зміні контракту. Даний тригер активується перед оновленням запису у таблиці contracts. При цьому він перевіряє, чи новий статус договору відповідає правильній послідовності зміни статусів. Якщо послідовність не є вірною, тригер спрацьовує та генерує помилку.

В Додатку В.2 можна детальніше ознайомитись з прикладами реалізації тригерів.

4.4 Опис інтерфейсу користувача

Неавторизований в додатку користувач може переглядати тільки інформацію про автомобілі (див. рис. 14), а після обрання якогось із них, він також зможе заповнити необхідною інформацією форму, яка відправить його дані на email компанії або зателефонувати менеджеру за телефоном указанному в додатку.

Переглянемо інтерфейс авторизованого користувача в ролі Менеджера. Він має можливість додавати/редагувати/видаляти інформацію.

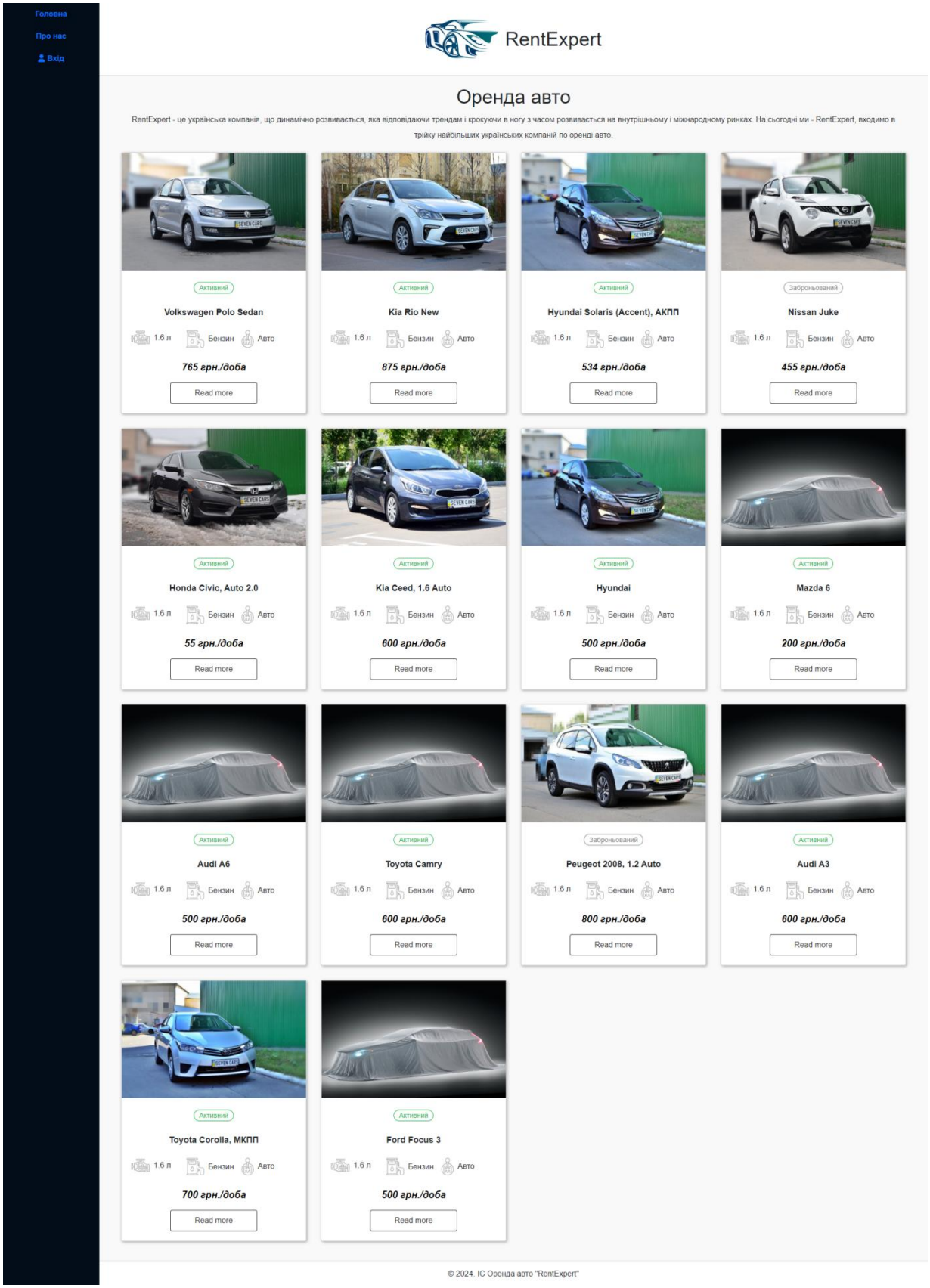


Рисунок 14 – Початковий інтерфейс неавторизованого користувача (рисунок виконано самостійно)

Відображення інформація про всі автомобілі (див. рис. 15)

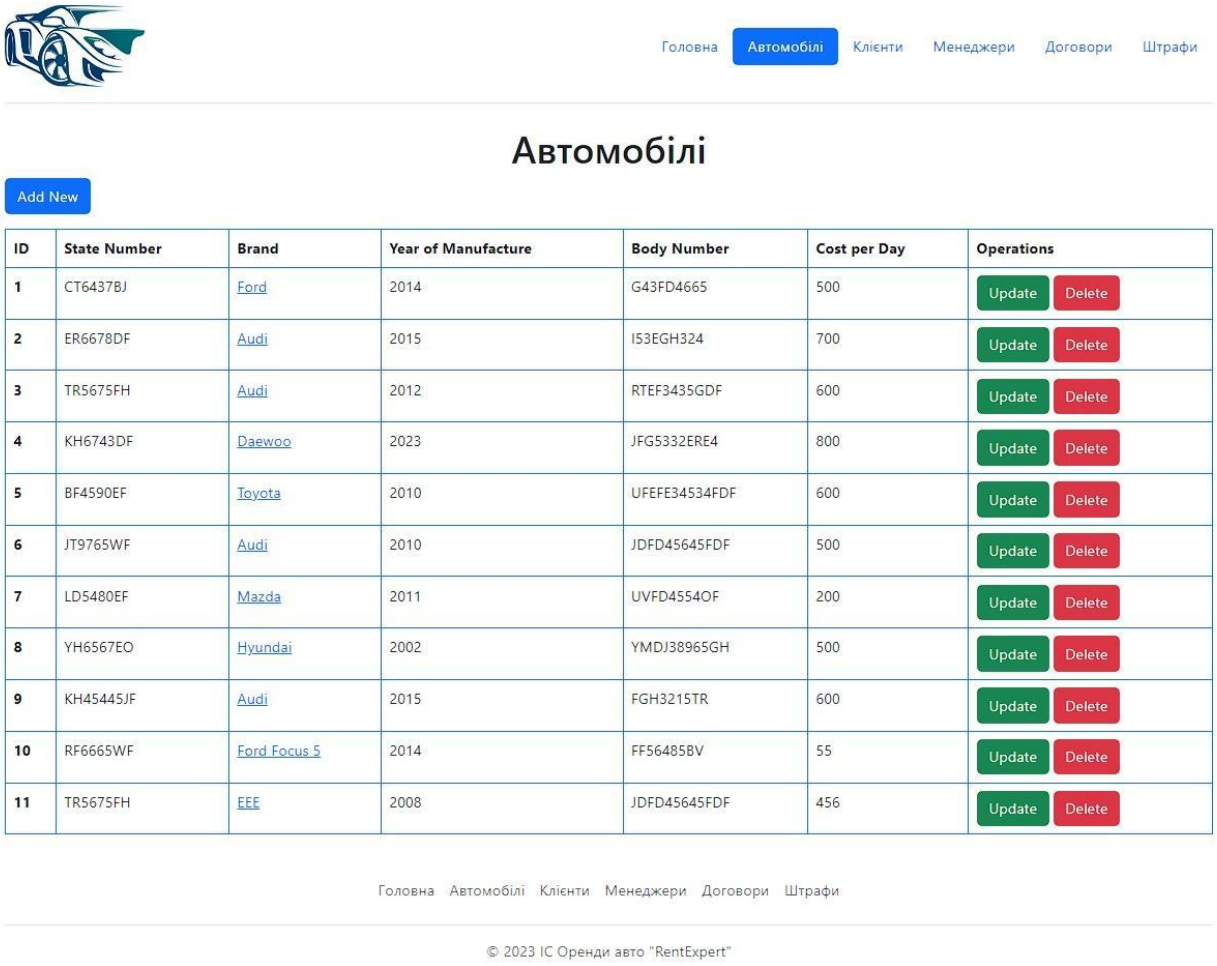


Рисунок 15 – Інтерфейс списку автомобілів для авторизованого користувача (рисунок виконано самостійно)

За допомогою кнопки Add New –> Add Car користувач зможе додати інформацію про новий автомобіль, заповнивши всі поля в формі (див. рис. 16)

На екрані також ми побачимо кнопки Update та Delete. Відповідно кнопка Update відкриє форму для редагування інформації (див. рис. 17), а кнопка Delete видалить конкретний запис із списку автомобілів без перезавантаження сторінки.

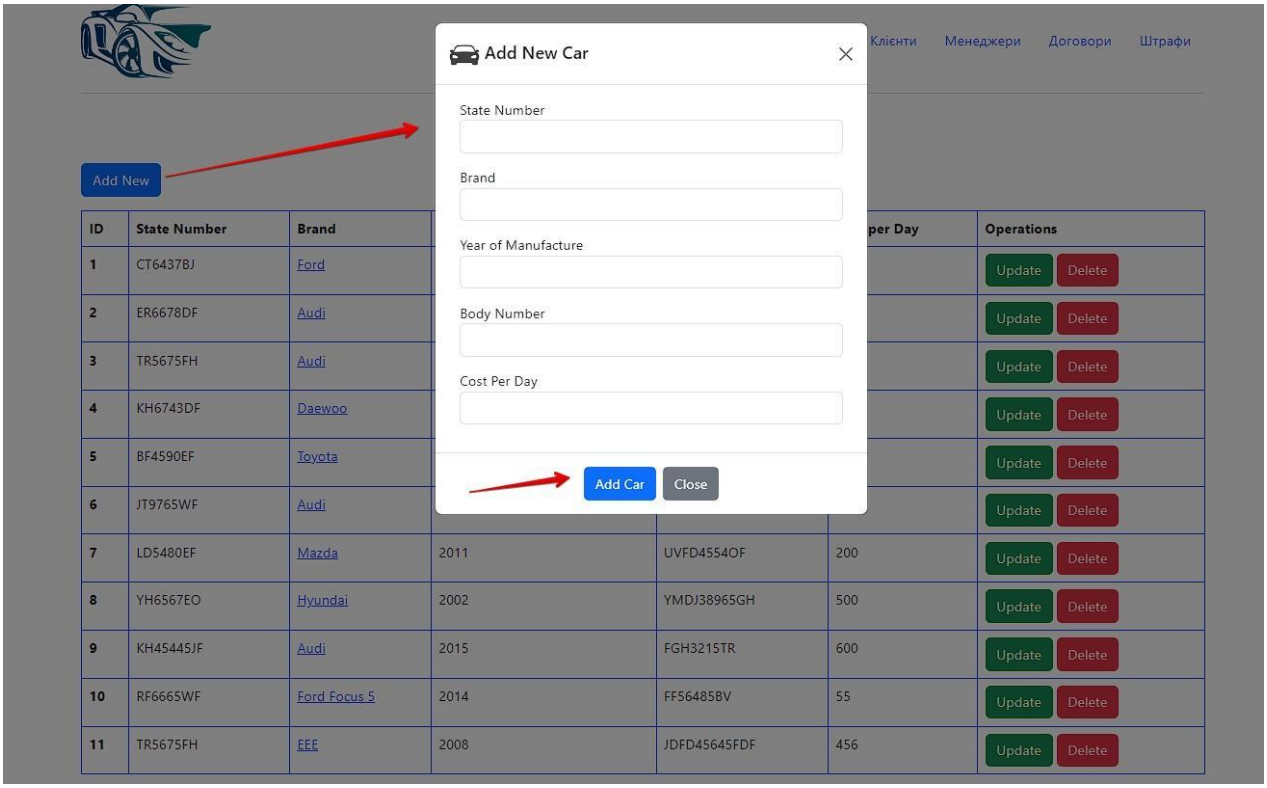


Рисунок 16 – Форма додавання інформації про автомобілі (рисунок виконано самостійно)

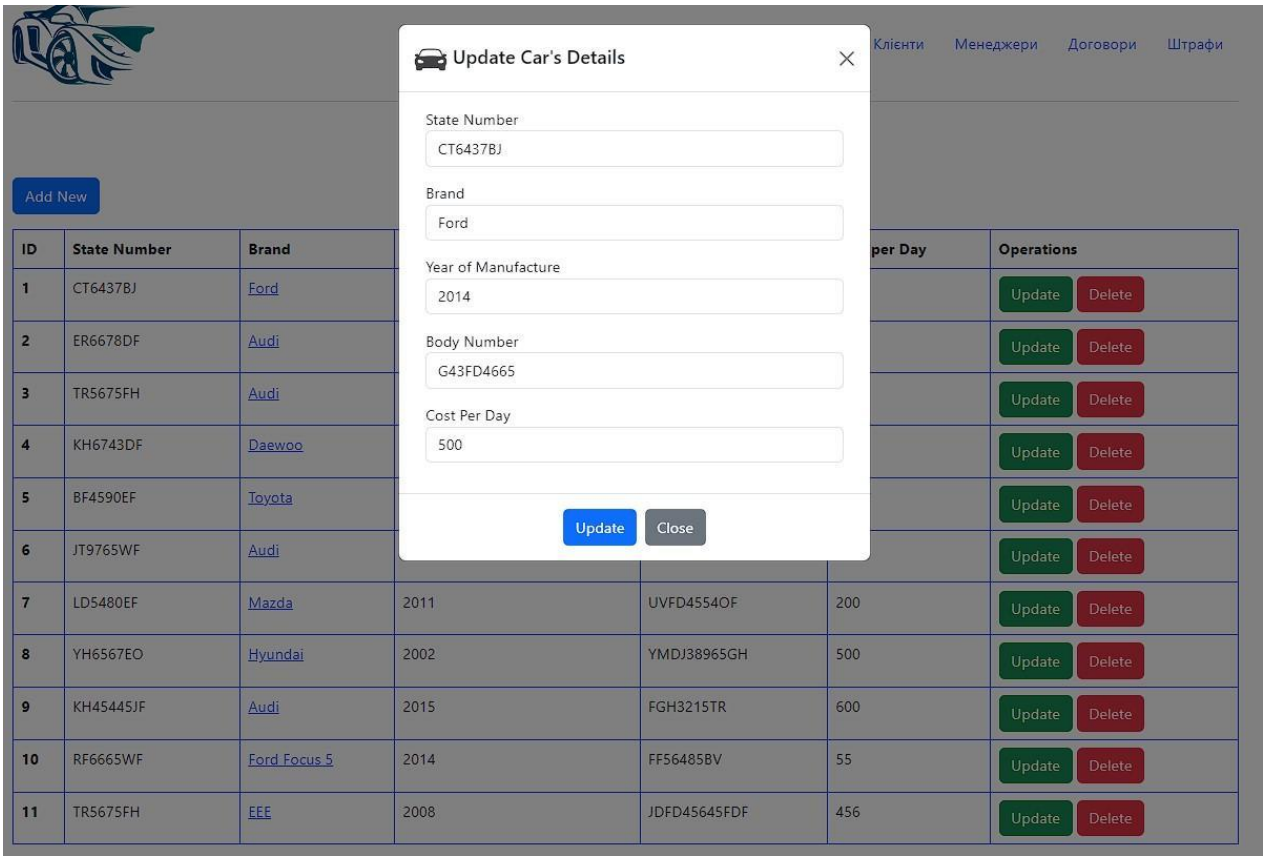


Рисунок 17 – Форма редагування інформації про автомобіль (рисунок виконано самостійно)

В додатку також є сторінка Договори, в виведеній таблиці посиланнями будуть відображатися також Менеджер, Клієнт та Автомобіль. Для перегляду більш детальної інформації можна буде перейти натиснувши на посилання в таблиці (див. рис. 18).

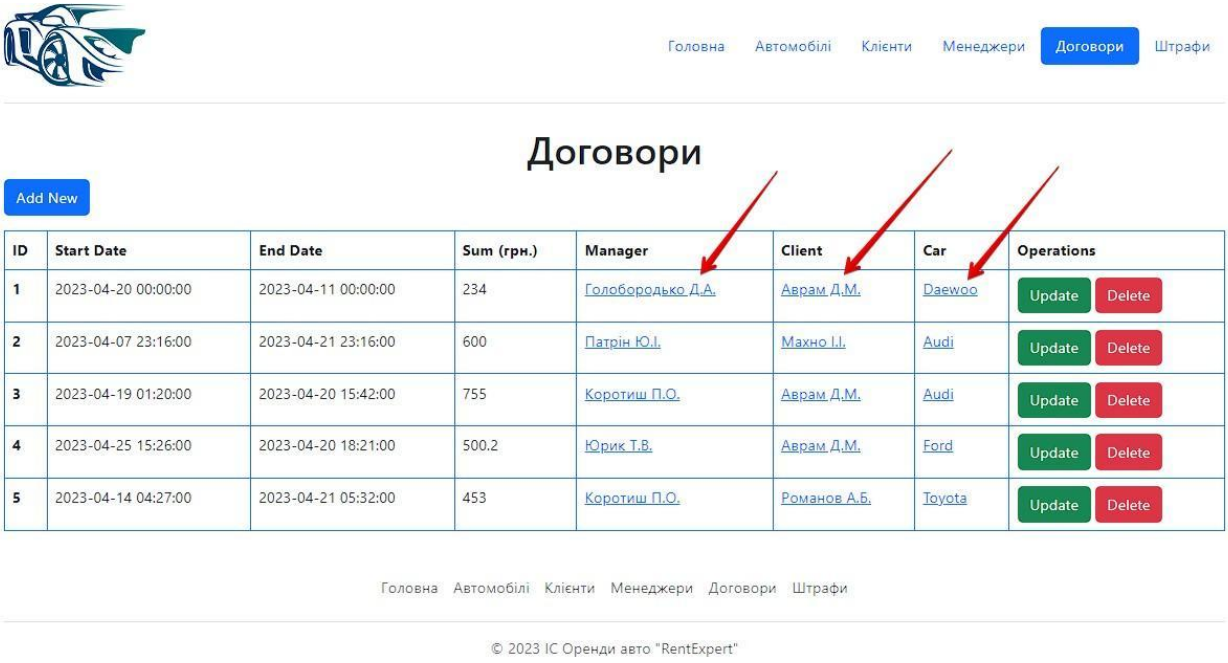


Рисунок 18 – Сорінка Договори з посиланнями (рисунок виконано самостійно)

В додатку ми маємо ще й інтерфейс для пошуку, фільтрації та сортування даних. Реалізацію всіх методів буду проводити з використанням таблиці *Cars* (див. рис. 19).

Рисунок 19 – Пошук, фільтрація та сортування (рисунок виконано самостійно)

Для реалізації фільтрації даних використовуємо оператор *BETWEEN* та *AND*. Візьмемо поля *Year of Manufacture* (від - до) та *Cost per Day* (від - до) (див. рис. 20).

```
SELECT * FROM cars WHERE year_of_manufacture BETWEEN 2008 AND 2014 AND
cost_per_day BETWEEN 400 AND 600 ORDER BY year_of_manufacture DESC
```

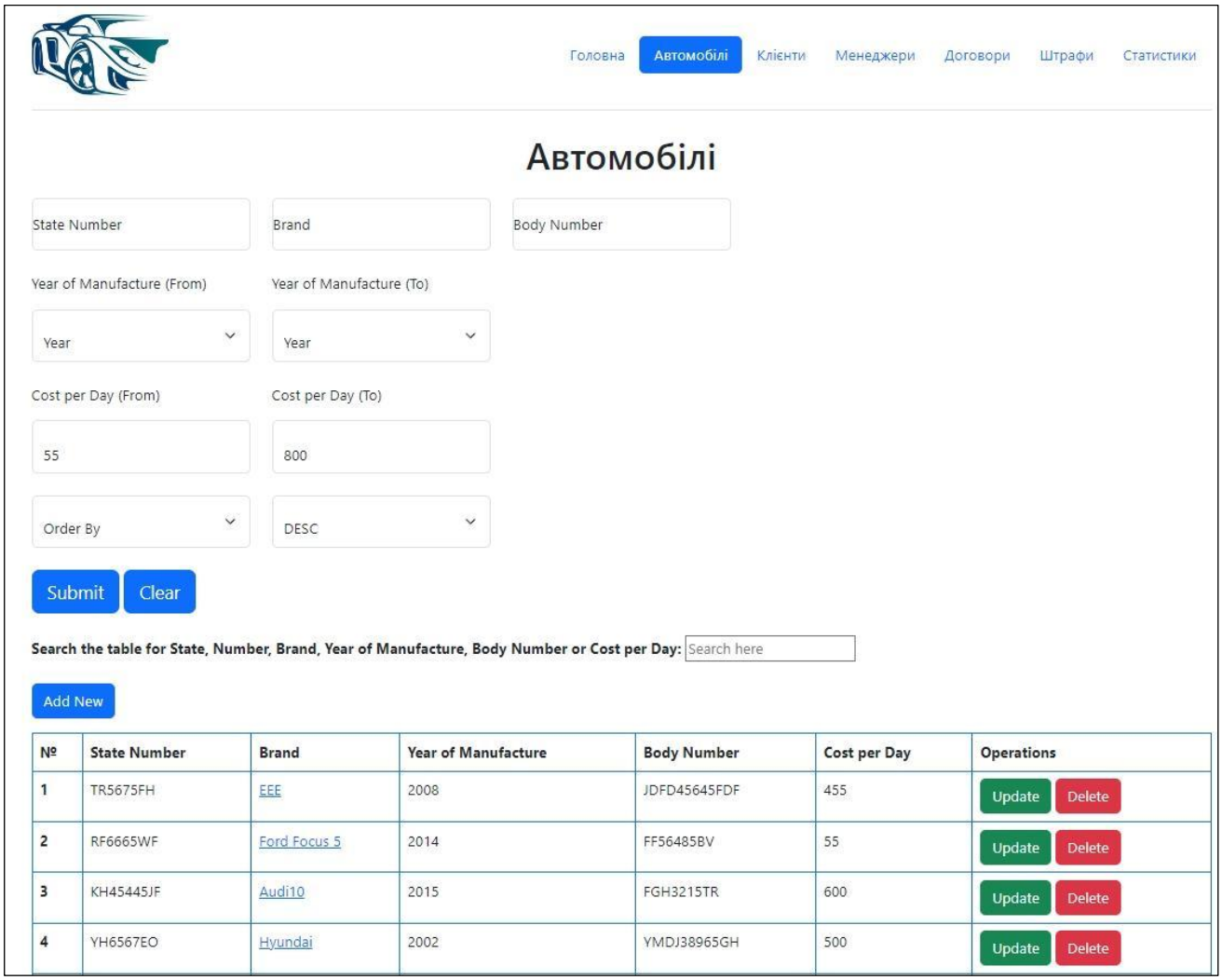


Рисунок 20 – Загальний вигляд пошуку, фільтрації та сортування (рисунок виконано самостійно)

Результат фільтрації наведено на рисунку 21.

№	State Number	Brand	Year of Manufacture	Body Number	Cost per Day	Operations
1	CT6437BJ	Ford	2014	G43FD4665	500	Update Delete
2	TR5675FH	Audi3	2012	RTEF3435GDF	600	Update Delete
3	BF4590EF	Toyota	2010	UFEFE34534FDF	600	Update Delete
4	JT9765WF	Audi6	2010	JDFD45645FDF	500	Update Delete
5	TR5675FH	EEE	2008	JDFD45645FDF	455	Update Delete

Рисунок 21 – Результат фільтрації даних (рисунок виконано самостійно)

Код реалізації цих методів описаний в додатоку В.

На сторінці *Статистики* ми бачимо різні види статистик (див. рис. 22).

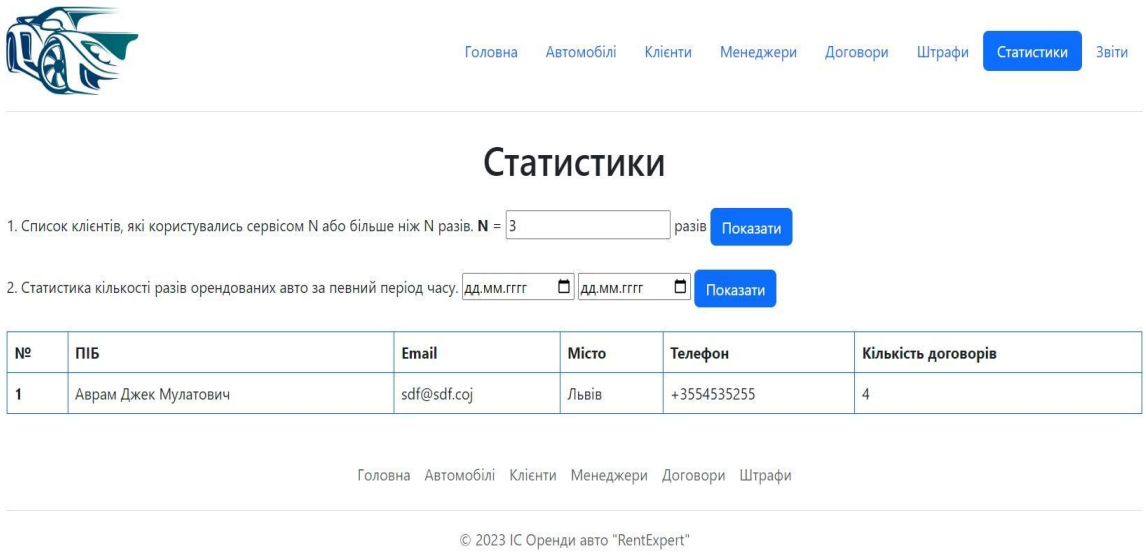


Рисунок 22 – Сторінка Статистики (рисунок виконано самостійно)

На сторінці Звіти ми бачимо різні види звітів, які можна завантажити в форматі .xls. Для реалізації функціоналу (Імпорт даних про автомобілі з файлу .xlsx**)** застосовано PHP бібліотеку PHPOffice/PhpSpreadsheet [8].

Припустимо, що в нас є файл з даними про автомобілі, які потрібно занести в нашу таблицю *cars* бази даних. Погодьтесь, що додавати в таблицю по одному запису із файлу це не дуже зручно, тож я вирішив зробити імпорт всіх даних з файлу в таблицю в одне натискання кнопки (див. рис. 23).

Рисунок 23 – Інтерфейс імпорту даних (рисунок виконано самостійно)

Структура даних в файлі повинна бути наступною (див. рис. 24).

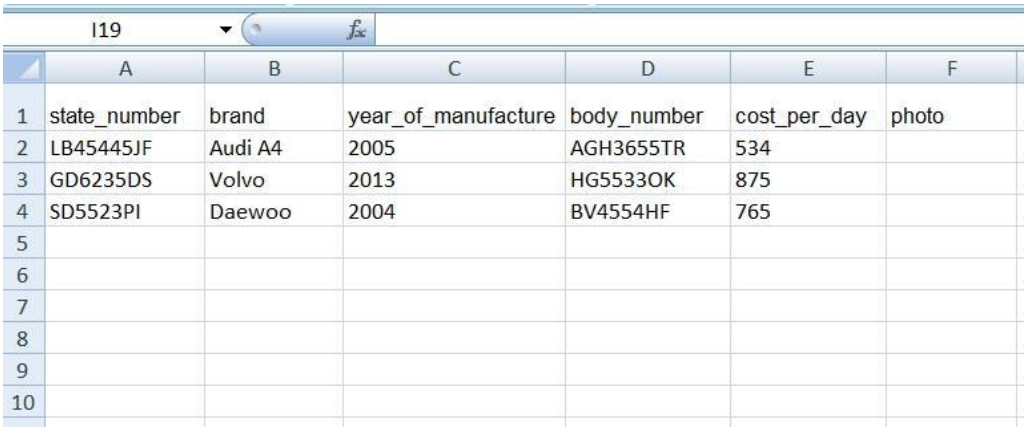


Рисунок 24 – Структура файлу .xlsx (рисунок виконано самостійно)

Якщо в таблиці вже є дані, які співпадають по якимось значенням, то вони просто будуть оновлюватися, в іншому випадку – додаватися в таблицю.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Обґрунтування вибору виду тестування

Завершуючи розробку програмної системи, важливо звернути увагу на критичний етап роботи — тестування. Існує багато різних підходів до тестування, кожен з яких належить до певної категорії. Наприклад, за рівнем тестування можна виділити такі підходи, як модульне, інтеграційне та системне тестування.

Враховуючи архітектуру та особливості системи, тестування слід поділити на дві частини: тестування серверної частини та тестування клієнтської частини. Обидві ці частини будуть протестовані за допомогою модульного тестування.

Модульне тестування — це метод, який полягає в окремому тестуванні кожного модуля програми. Модуль є найменшою частиною програми, яку можна протестувати. Кожен метод тестування розглядається як модуль, який необхідно перевірити.

Після визначення частин системи для тестування та вибору підходу необхідно обрати ступінь тестування. Існують два види тестування: ручне та автоматизоване. Ручне тестування передбачає перевірку тест-кейсів вручну без використання автоматизованих інструментів. Автоматизоване тестування, навпаки, здійснюється за допомогою набору інструментів та засобів автоматизації. Оскільки метою тестування на цьому етапі є виявлення помилок та неточностей у системі, автоматизоване тестування є найкращим варіантом.

5.2 Опис тестування

В Таблиці 1, Таблиці 2 та Таблиці 3 наведено приклад ручного тестування одного з модулів програмної системи.

Таблиця 1. – Тест-кейс для авторизації користувача в ПС.

ID	Назва	Опис	Попередні умови	Кроки для виконання	Очікуваний результат
ТС-001	Авторизація користувача з валідними даними	Перевірка успішної авторизації користувача в системі за допомогою валідних даних (email та пароль).	1. Користувач має обліковий запис на сайті. 2. Користувач знає свій email та пароль.	1. Відкрити веб-браузер і перейти на сторінку авторизації системи http://rent-cars.loc/login.php . 2. Ввести валідний email. 3. Ввести валідний пароль. 4. Натиснути кнопку "Вхід".	1. Користувач успішно авторизується на сайті. 2. Відображається головна сторінка користувача http://rent-cars.loc/ . 3. В лівому сайдбарі сторінки відображається ім'я користувача або його профільна інформація.

Таблиця 2. – Тест-кейс авторизації користувача з невалідною електронною адресою

ID	Назва	Опис	Попередні умови	Кроки для виконання	Очікуваний результат
ТС-002	Авторизація з невалідною електронною адресою	Перевірка спроби авторизації з невалідною електронною адресою.	1. Користувач має обліковий запис на сайті. 2. Користувач знає свій email та пароль.	1. Відкрити веб-браузер і перейти на сторінку авторизації сайту. 2. Ввести невалідну електронну адресу в поле "Email". 3. Ввести валідний пароль. 4. Натиснути кнопку "Вхід".	1. Відображається повідомлення про помилку " Email або пароль не вірні!". 2. Користувач не може авторизуватися на сайті.

Таблиця 3. – Тест-кейс авторизації користувача з невалідним паролем

ID	Назва	Опис	Попередні умови	Кроки для виконання	Очікуваний результат
TC-003	Авторизація з невалідним паролем	Перевірка спроби авторизації з невалідним паролем.	1. Користувач має обліковий запис на сайті. 2. Користувач знає свій email та пароль.	1. Відкрити веб-браузер і перейти на сторінку авторизації сайту. 2. Ввести валідну електронну адресу в поле "Email". 3. Ввести невалідний пароль у поле "Пароль". 4. Натиснути кнопку "Вхід".	1. Відображається повідомлення про помилку " Email або пароль не вірні!". 2. Користувач не може авторизуватися на сайті.

Проведене тестування дозволило виявити та виправити всі виявлені помилки, а також оптимізувати роботу програмної системи. У ході тестування було використано ручнее тестування, що забезпечило високу якість перевірки.

Результати тестування підтвердили відповідність програмної системи заявленим вимогам, забезпечили стабільну роботу та гарантували безпечність і надійність функціонування. Оптимізація коду та підвищення продуктивності дозволили досягти покращення швидкості виконання основних операцій, що значно підвищило загальну ефективність системи.

ВИСНОВКИ

Під час дипломування було проведено аналіз та моделювання предметної області, проектування БД, розробку архітектури системи, її проектування та реалізацію.

В результаті було розроблено програмну систему оренди автомобілів, яка дозволяє взаємодіяти та зберігати дані про автомобілі, менеджерів, клієнтів, договори між менеджерами та клієнтами та ДТП автомобілів. Також програма дозволяє шукати, фільтрувати інформацію, отримувати статистики продажів, підтримує задачу автоматизації керування контрактом.

Застосунок є легким для розуміння та швидкого пристосування, отже людина з будь-яким досвідом зможе з легкістю його використовувати.

В ході створення додатку були використані середа розробки IDE PhpStorm 2022.1, локальний веб-сервер для Windows (OpenServer) з Nginx+Apach, мова програмування PHP, обрано веб-технології HTML, CSS, JAVASCRIPT, Framework Bootstrap та система керування базами даних (СКБД) MySQL.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Укр-Прокат - оренда автомобілів в Україні [Електронний ресурс]. – 2013-2023. – Режим доступу до ресурсу: https://ukr-prokat.com/orenda-avto/kyiv (дата звернення: 7.05.2024).

2. RENTAL.UA – Оренда нових автомобілів з доставкою за адресою за 1 годину [Електронний ресурс]. – Rental 2023. – Режим доступу до ресурсу: https://rental.ua/ua (дата звернення: 7.05.2024).

3. Bootstrap 5.3 (Front-end frameworks) [Електронний ресурс] – https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення 10.05.2024)

4. PHP документація [Електронний ресурс] – https://www.php.net/ (дата звернення 18.05.2024)

5. PHP 7 в оригіналі – Дмитро Котеров, Ігор Симдянов (дата звернення 18.05.2024).

6. Open Server документація URL https://ospanel.io/ (дата звернення 23.05.2024)

7. phpMyAdmin документація URL: https://www.phpmyadmin.net/ (дата звернення 23.05.2024).

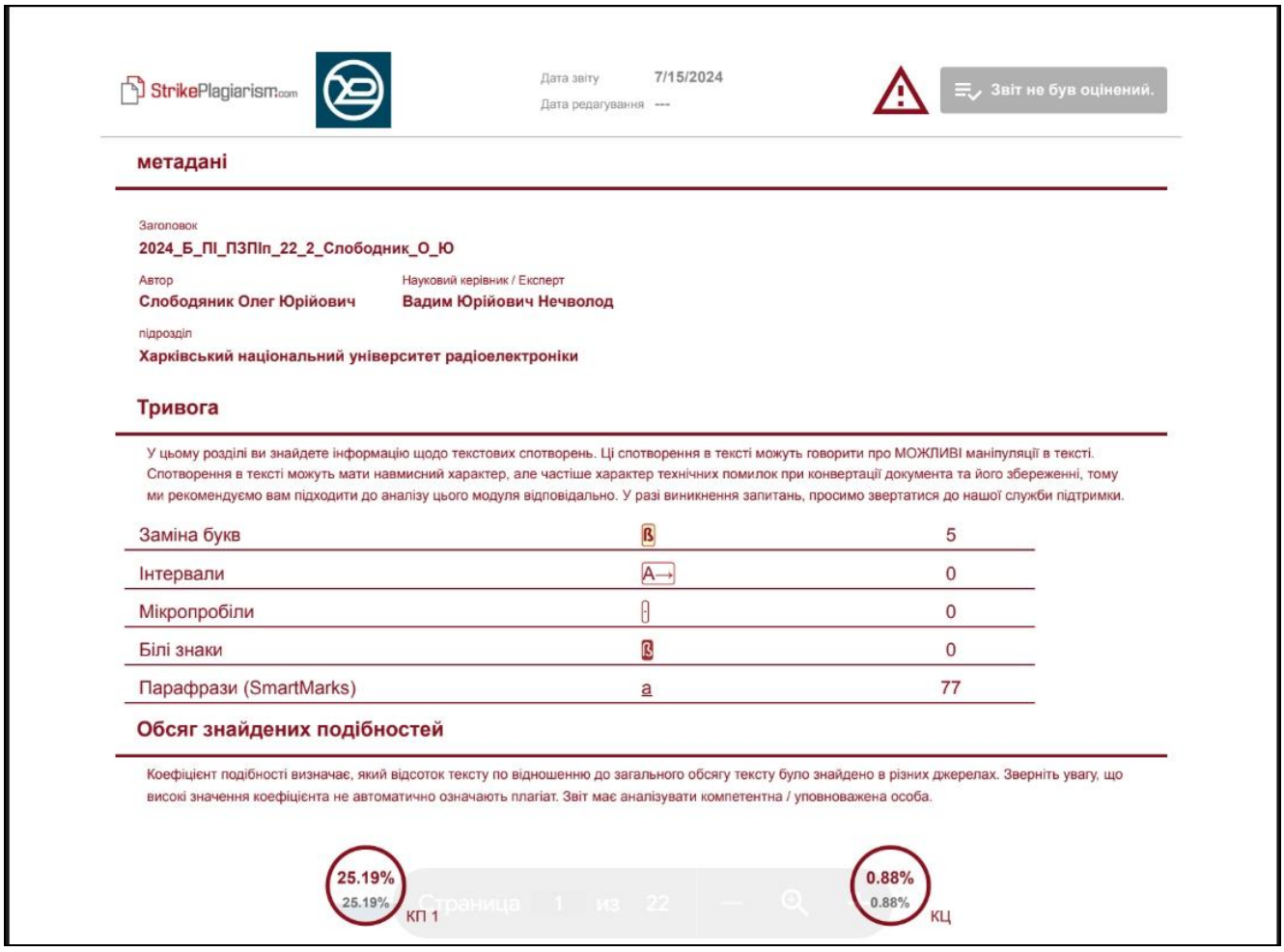
8. Бібліотека PhpSpreadsheet [Електронний ресурс]. - 2023 GitHub, Inc. – Режим доступу до ресурсу: https://github.com/PHPOffice/PhpSpreadsheet (дата звернення 3.06.2024).

9. Business Automation Software (BAS), програмні рішення для створення комплексних систем автоматизації бізнесу [Електронний ресурс] – https://www.bas-soft.eu/ (дата звернення 9.06.2024).

10. Щоденні задачі UX/UI дизайнера. Програми для дизайну [Електронний ресурс] – https://frusia.pro/p/9 (дата звернення 11.06.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 7/15/2024
Дата редагування ---

Звіт не був оцінений.

метадані

Заголовок
2024_Б_ПІ_ПЗПІп_22_2_Слободник_О_Ю

Автор
Слободяник Олег Юрійович

Науковий керівник / Експерт
Вадим Юрійович Нечволод

підрозділ
Харківський національний університет радіоелектроніки

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	5
Інтервали	A→	0
Мікропробіли		0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	77

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

25.19% 25.19% КП 1

Страница 1 из 22

0.88% 0.88% КЦ

ДОДАТОК Б

Слайди презентації

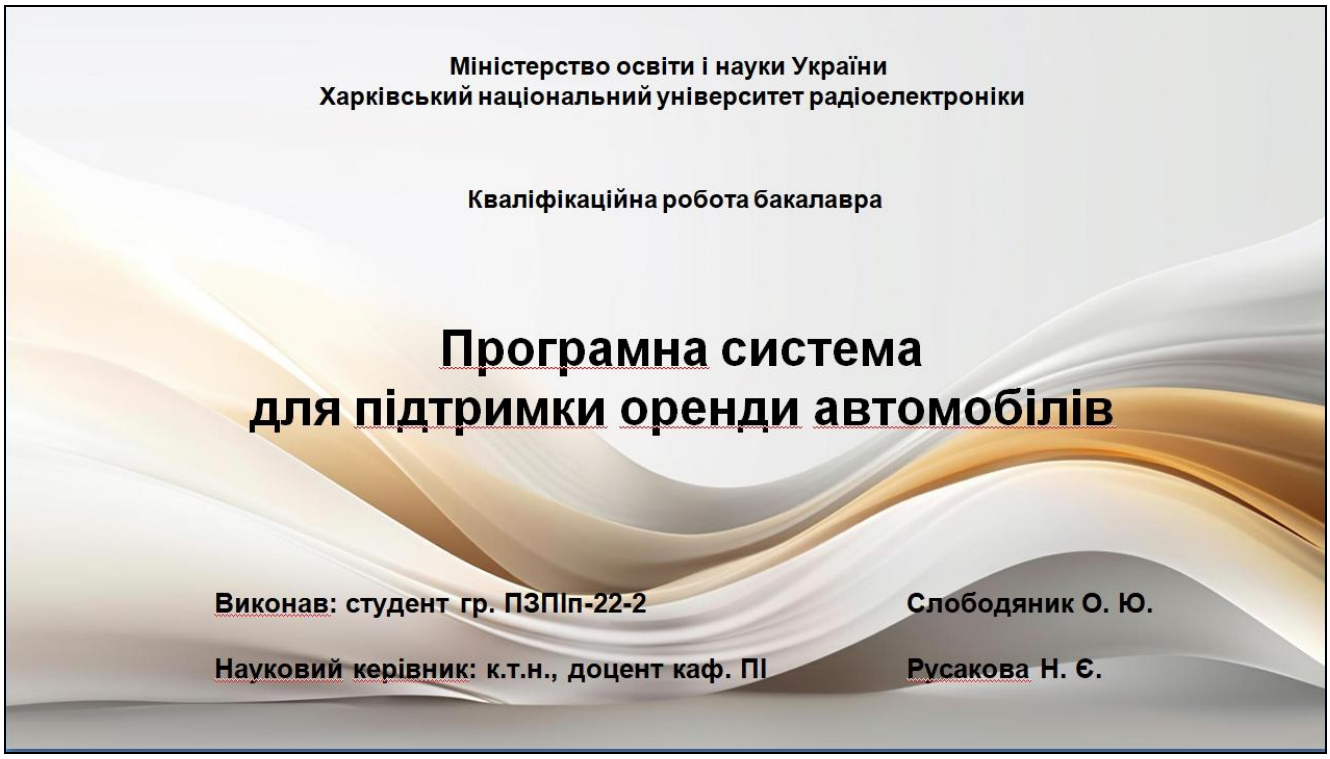


Рисунок Б.1 – Титульний слайд

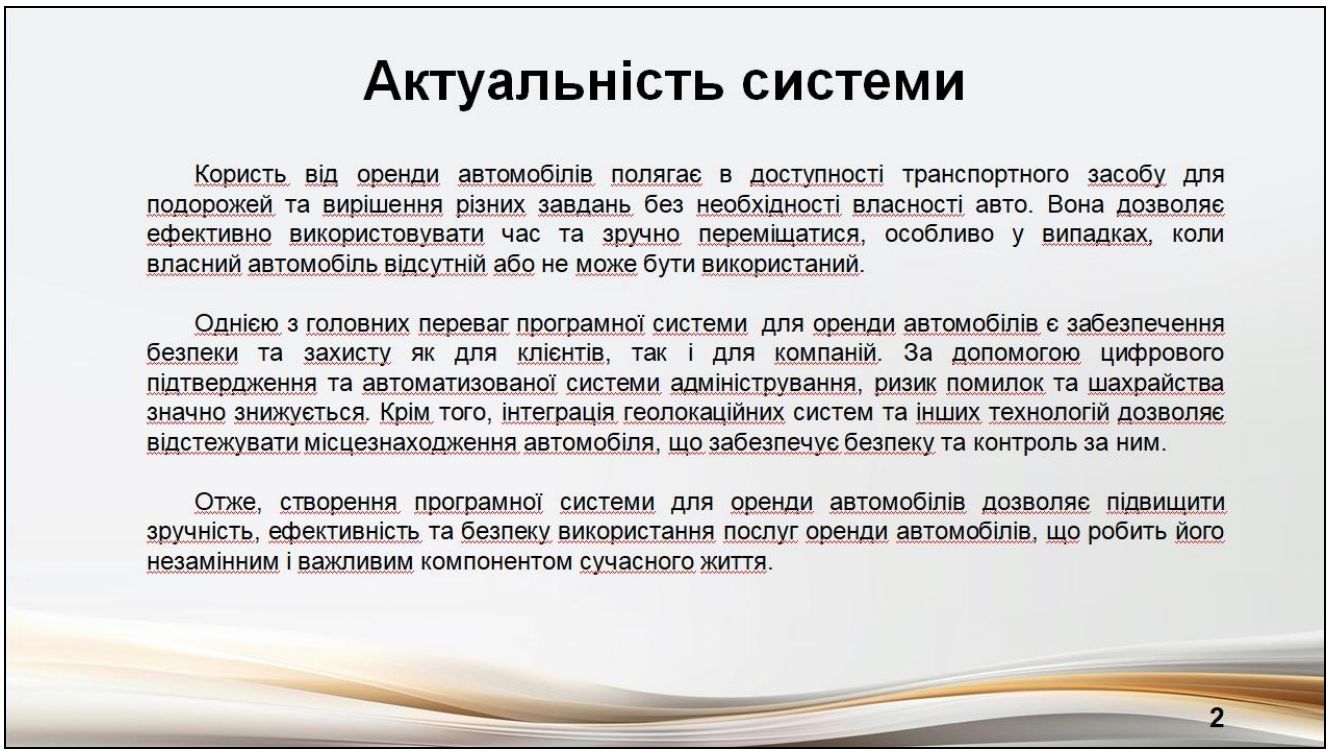


Рисунок Б.2 – Слайд «Актуальність системи»

Постановка задачі

- провести аналіз та моделювання предметної області програмної системи;
- спроектувати базу даних для збереження інформації з предметної області;
- розробити алгоритми підтримки оренди автомобіля на основних її етапах;
- спроектувати архітектуру програмної системи;
- виконати програмну реалізацію серверної та клієнтської частин системи;
- провести тестування створеного програмного продукту.

3

Рисунок Б.3 – Слайд «Постановка задачі»

Аналіз та моделювання предметної області

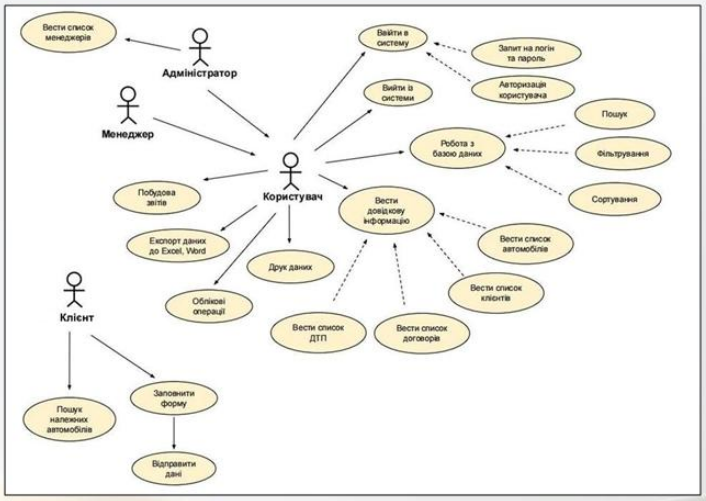


Рисунок 1 – USE-CASE діаграма

4

Рисунок Б.4 – Слайд «Аналіз моделювання предметної області»

Рисунок 2 – Загальна діаграма класів

Рисунок Б.5 – Слайд «Концептуальне моделювання»

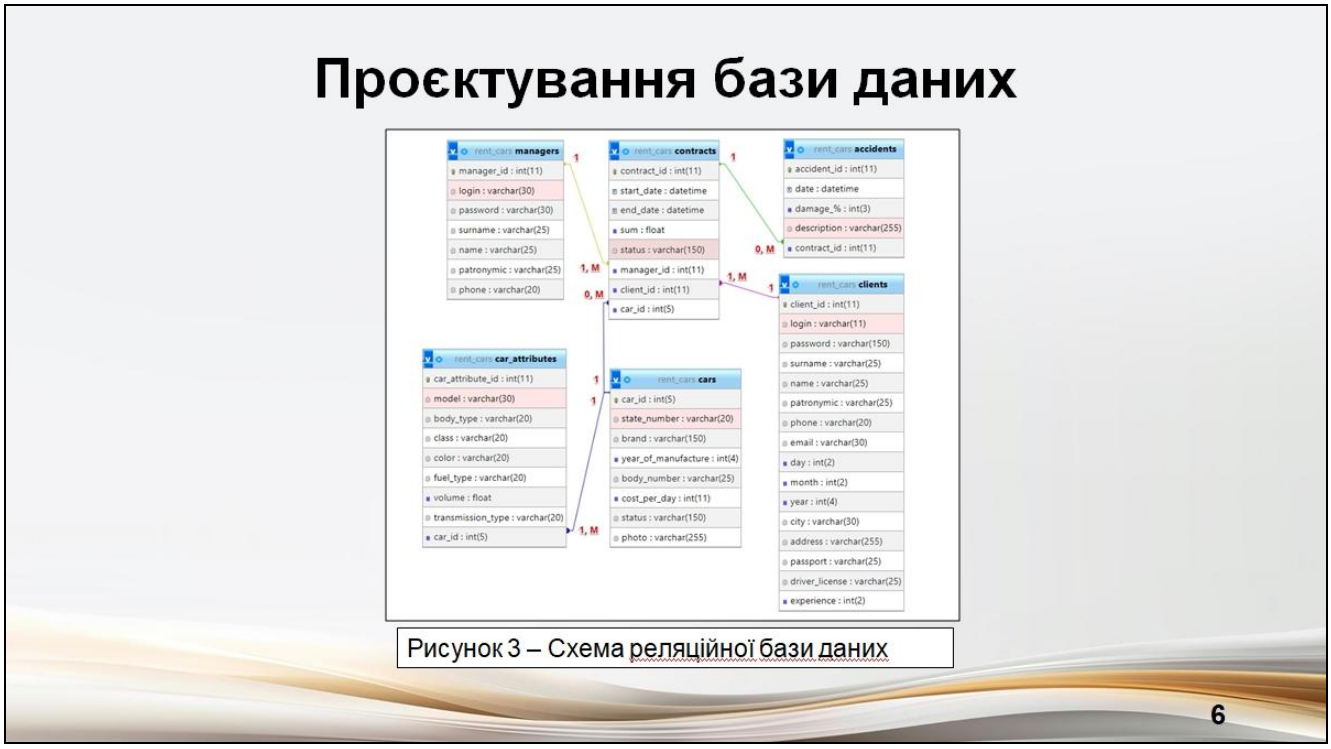


Рисунок 3 – Схема реляційної бази даних

Рисунок Б.6 – Слайд «Проєктування бази даних»

Рисунок Б.7 – Слайд «Алгоритм бронювання автомобіля»

Рисунок Б.8 – Слайд «Проєктування архітектури»

Рисунок Б.9 – Слайд «Проєктування UI/UX»

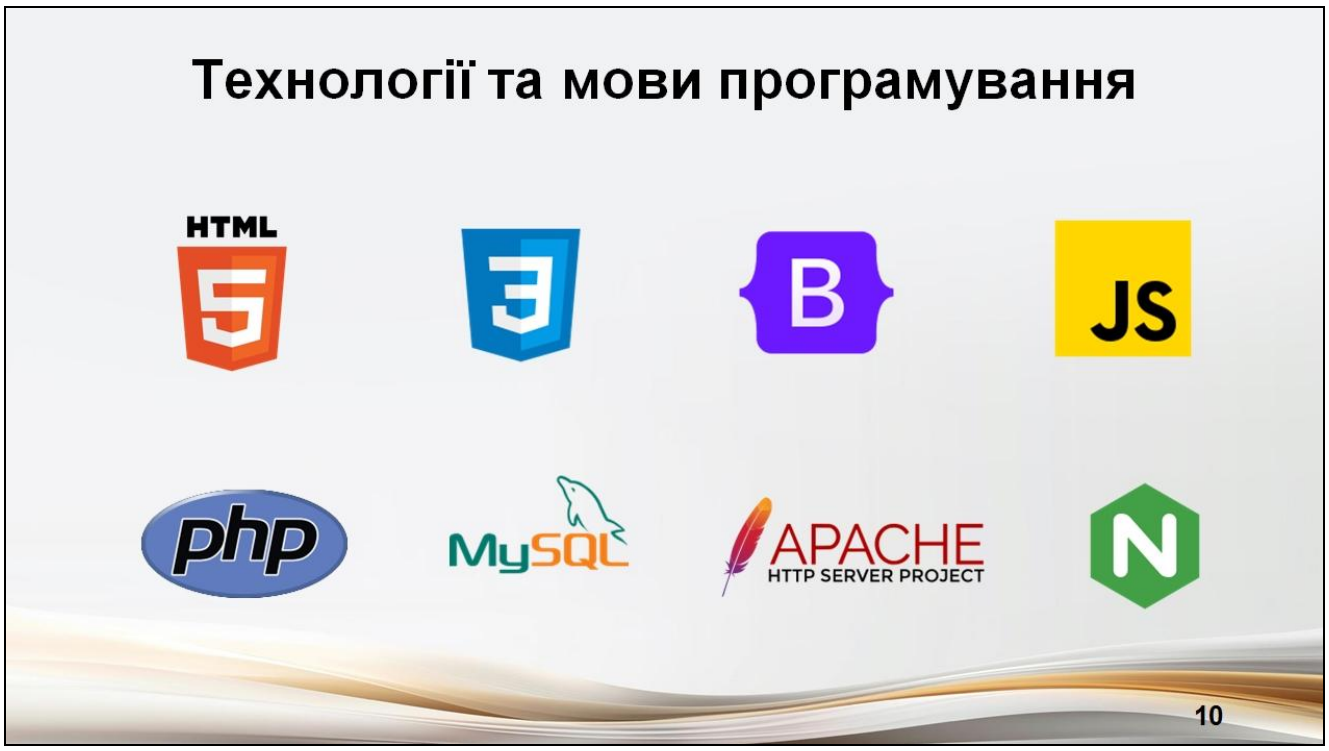


Рисунок Б.10 – Слайд «Технології та мови програмування»

Розробка серверної частини

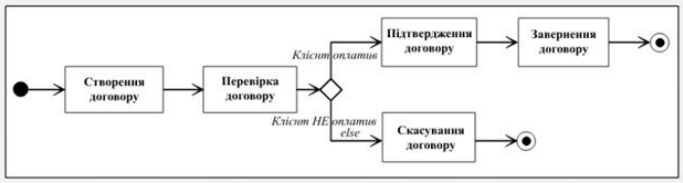


Рисунок 7 – Діаграма станів договору

Даний тригер активується перед оновленням запису у таблиці contracts. При цьому він перевіряє, чи новий статус договору відповідає правильній послідовності зміни статусів.

Якщо послідовність не є вірною, тригер спрацьовує та генерує помилку.

```
DELIMITER $$
DELIMITER $$
CREATE TRIGGER check_contract_status_sequence
    BEFORE UPDATE ON contracts
    FOR EACH ROW
BEGIN
    DECLARE old_contract_status VARCHAR(255);
    DECLARE new_contract_status VARCHAR(255);

    -- Отримуємо старий та новий статуси договору
    SET old_contract_status = OLD.status;
    SET new_contract_status = NEW.status;

    -- Перевіряємо послідовність зміни статусів
    IF (old_contract_status = 'created' AND new_contract_status != 'awaiting_payment') OR
       (old_contract_status = 'awaiting_payment' AND new_contract_status != 'active') OR
       (old_contract_status = 'active' AND new_contract_status NOT IN ('completed', 'canceled')) OR
       (old_contract_status = 'completed' AND new_contract_status != 'canceled') OR
       (old_contract_status = 'canceled' AND new_contract_status = 'active') OR
       (old_contract_status = 'completed' AND new_contract_status = 'active') THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Недопустима послідовність зміни статусів договору';
    END IF;
END$$
DELIMITER;
```

Рисунок 8 – Тригер для перевірки коректної зміни статусу при зміні статусу договору

11

Рисунок Б.11 – Слайд «Розробка серверної частини»

Інтерфейс системи

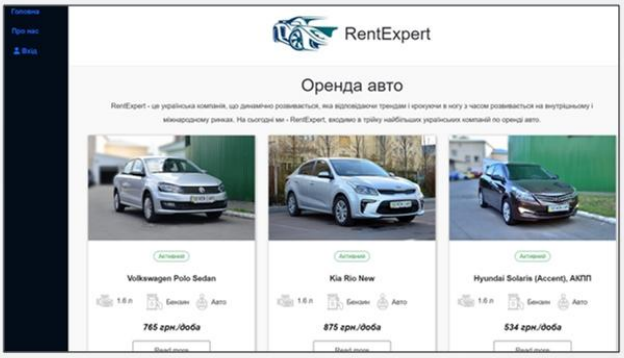


Рисунок 9 – Головна сторінка

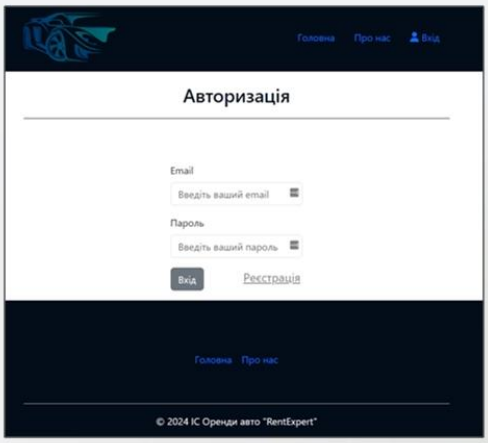


Рисунок 10 – Сторінка вторизації

12

Рисунок Б.12 – Слайд «Інтерфейс системи»

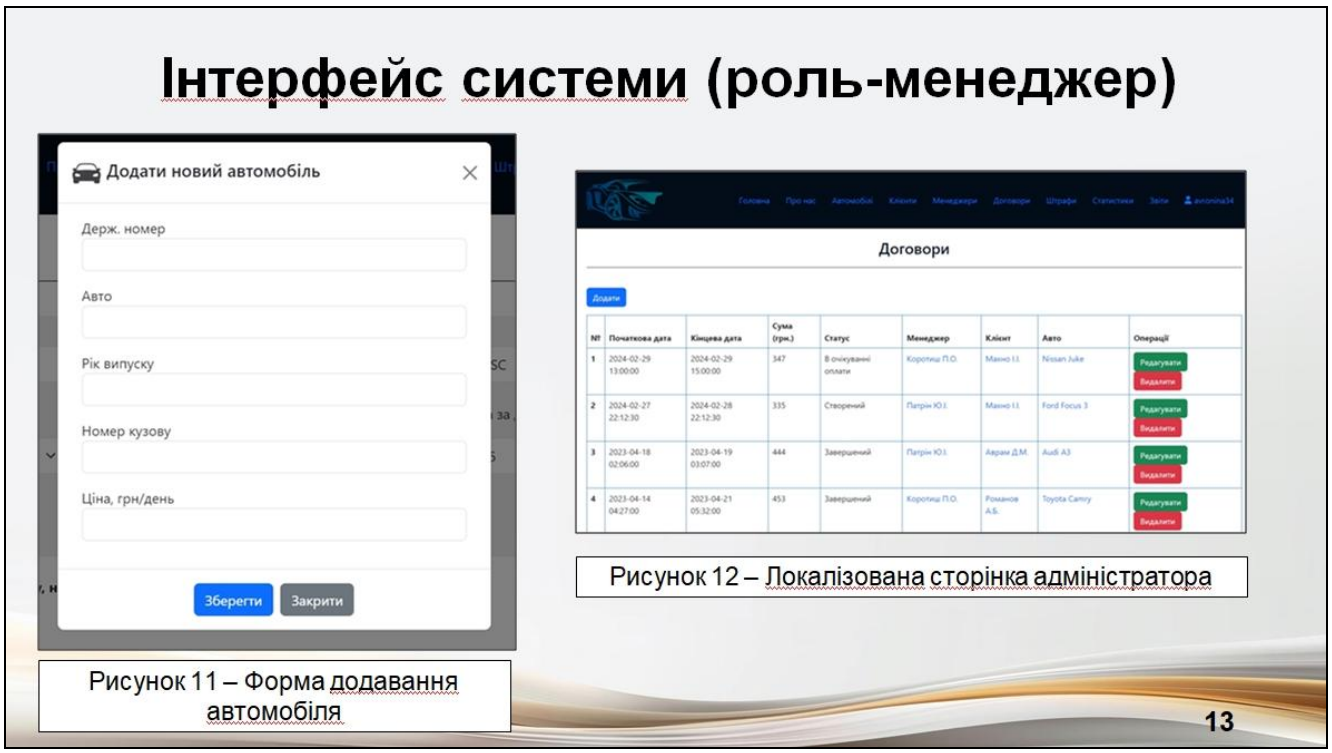


Рисунок 11 – Форма додавання автомобіля

Рисунок 12 – Локалізована сторінка адміністратора

Рисунок Б.13 – Слайд «Інтерфейс системи (роль-менеджер)»

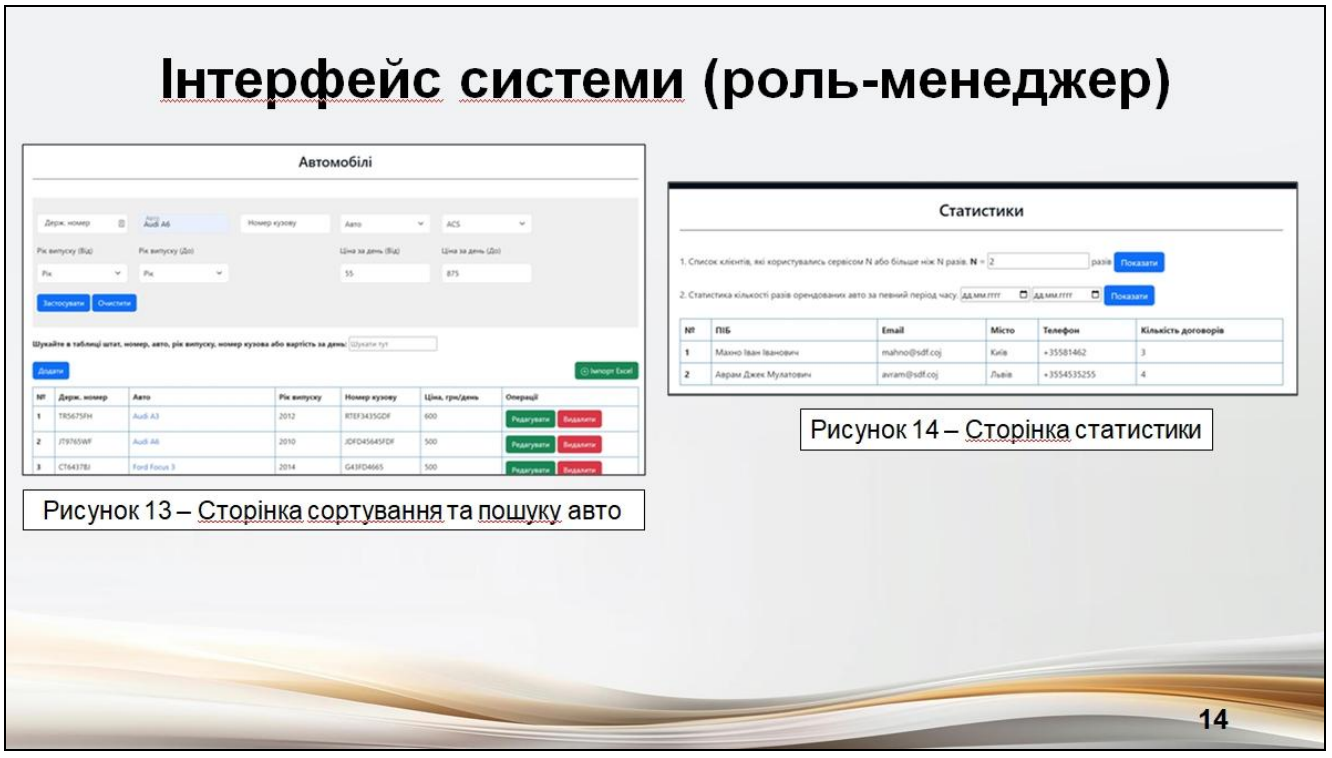


Рисунок 13 – Сторінка сортування та пошуку авто

Рисунок 14 – Сторінка статистики

Рисунок Б.14 – Слайд «Інтерфейс системи (роль-менеджер)»

Тестування системи

Таблиця 1. – Тест-кейс для авторизації користувача в ПС.

ID	Назва	Опис	Попередні умови	Кроки для виконання	Очікуваний результат
TC-001	Авторизація користувача з валідними даними	Перевірка успішної авторизації користувача в системі за допомогою валідних даних (email та пароль).	1. Користувач має обліковий запис на сайті. 2. Користувач знає свій email та пароль.	1. Відкрити веб-браузер і перейти на сторінку авторизації системи http://rent-cars.loc/login.php . 2. Ввести валідний email. 3. Ввести валідний пароль. 4. Натиснути кнопку "Вхід".	1. Користувач успішно авторизується на сайті. 2. Відображається головна сторінка користувача http://rent-cars.loc/ . 3. В лівому сайдбарі сторінки відображається ім'я користувача або його профільна інформація.

15

Рисунок Б.14 – Слайд «Тестування системи»

Висновки

При виконанні поставлених задач було:

- ✔ проведено аналіз та моделювання предметної області програмної системи;
- ✔ спроектовано базу даних для збереження інформації з предметної області;
- ✔ розроблено алгоритми підтримки оренди автомобіля на основних її етапах;
- ✔ спроектовано архітектуру програмної системи;
- ✔ виконано програмну реалізацію серверної та клієнтської частин системи
- ✔ проведено тестування створеного програмного продукту.

16

Рисунок Б.15 – Слайд «Висновеи»

ДОДАТОК В

Лістинг програмного коду

Приклад коду В.1 – Фрагмент реалізації створення бази даних.

```
--
-- База даних: `rent_cars`
--

-- --------------------------------------------------------

--
-- Структура таблиці `accidents`
--

CREATE TABLE `accidents` (
  `accident_id` int(11) NOT NULL,
  `date` datetime NOT NULL,
  `damage_%` int(3) NOT NULL,
  `description` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `contract_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- --------------------------------------------------------

--
-- Структура таблиці `cars`
--

CREATE TABLE `cars` (
  `car_id` int(5) NOT NULL,
  `state_number` varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL,
  `brand` varchar(150) COLLATE utf8mb4_unicode_ci NOT NULL,
  `year_of_manufacture` int(4) NOT NULL,
  `body_number` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `cost_per_day` int(11) NOT NULL,
  `status` varchar(150) COLLATE utf8mb4_unicode_ci NOT NULL,
  `photo` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

--
-- Структура таблиці `car_attributes`
--

CREATE TABLE `car_attributes` (
  `car_attribute_id` int(11) NOT NULL,
  `model` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `body_type` varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL,
  `class` varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL,
  `color` varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL,
  `fuel_type` varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL,
  `volume` float NOT NULL,
  `transmission_type`  varchar(20)  COLLATE  utf8mb4_unicode_ci  NOT
NULL,
  `car_id` int(5) NOT NULL
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- --------------------------------------------------------

--
-- Структура таблиці `clients`
--

CREATE TABLE `clients` (
  `client_id` int(11) NOT NULL,
  `surname` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `name` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `patronymic` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `phone` varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL,
  `email` varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL,
  `day` int(2) NOT NULL,
  `month` int(2) NOT NULL,
  `year` year(4) NOT NULL,
  `city` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `address` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `passport` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `driver_license` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `experience` int(2) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- --------------------------------------------------------

--
-- Структура таблиці `contracts`
--

CREATE TABLE `contracts` (
  `contract_id` int(11) NOT NULL,
  `start_date` datetime NOT NULL,
  `end_date` datetime NOT NULL,
  `sum` float NOT NULL,
  `status` varchar(150) COLLATE utf8mb4_unicode_ci NOT NULL,
  `manager_id` int(11) NOT NULL,
  `client_id` int(11) NOT NULL,
  `car_id` int(5) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- --------------------------------------------------------

--
-- Структура таблиці `managers`
--

CREATE TABLE `managers` (
  `manager_id` int(11) NOT NULL,
  `login` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `password` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `surname` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `name` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `patronymic` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,
  `phone` varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

```
--
-- Індекси збережених таблиць
--

--
-- Індекси таблиці `accidents`
--
ALTER TABLE `accidents`
  ADD PRIMARY KEY (`accident_id`),
  ADD KEY `contract_id` (`contract_id`);

--
-- Індекси таблиці `cars`
--
ALTER TABLE `cars`
  ADD PRIMARY KEY (`car_id`);

--
-- Індекси таблиці `car_attributes`
--
ALTER TABLE `car_attributes`
  ADD PRIMARY KEY (`car_attribute_id`),
  ADD KEY `car_id` (`car_id`);

--
-- Індекси таблиці `clients`
--
ALTER TABLE `clients`
  ADD PRIMARY KEY (`client_id`);

--
-- Індекси таблиці `contracts`
--
ALTER TABLE `contracts`
  ADD PRIMARY KEY (`contract_id`),
  ADD KEY `client_id` (`client_id`),
  ADD KEY `car_id` (`car_id`),
  ADD KEY `manager_id` (`manager_id`) USING BTREE;

--
-- Індекси таблиці `managers`
--
ALTER TABLE `managers`
  ADD PRIMARY KEY (`manager_id`);

--
-- AUTO_INCREMENT для збережених таблиць
--

--
-- AUTO_INCREMENT для таблиці `accidents`
--
ALTER TABLE `accidents`
  MODIFY `accident_id` int(11) NOT NULL AUTO_INCREMENT;

--
```

```
-- AUTO_INCREMENT для таблиці `cars`
--
ALTER TABLE `cars`
  MODIFY `car_id` int(5) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=17;

--
-- AUTO_INCREMENT для таблиці `car_attributes`
--
ALTER TABLE `car_attributes`
  MODIFY `car_attribute_id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT для таблиці `clients`
--
ALTER TABLE `clients`
  MODIFY    `client_id`    int(11)    NOT    NULL    AUTO_INCREMENT,
AUTO_INCREMENT=8;

--
-- AUTO_INCREMENT для таблиці `contracts`
--
ALTER TABLE `contracts`
  MODIFY    `contract_id`    int(11)    NOT    NULL    AUTO_INCREMENT,
AUTO_INCREMENT=12;

--
-- AUTO_INCREMENT для таблиці `managers`
--
ALTER TABLE `managers`
  MODIFY    `manager_id`    int(11)    NOT    NULL    AUTO_INCREMENT,
AUTO_INCREMENT=6;

--
-- Обмеження зовнішнього ключа збережених таблиць
--

--
-- Обмеження зовнішнього ключа таблиці `accidents`
--
ALTER TABLE `accidents`
  ADD  CONSTRAINT  `accidents_ibfk_1`  FOREIGN  KEY  (`contract_id`)
REFERENCES `contracts` (`contract_id`);

--
-- Обмеження зовнішнього ключа таблиці `car_attributes`
--
ALTER TABLE `car_attributes`
  ADD  CONSTRAINT  `car_attributes_ibfk_1`  FOREIGN  KEY  (`car_id`)
REFERENCES `cars` (`car_id`);

--
-- Обмеження зовнішнього ключа таблиці `contracts`
--
ALTER TABLE `contracts`
  ADD  CONSTRAINT  `contracts_ibfk_1`  FOREIGN  KEY  (`manager_id`)
REFERENCES `managers` (`manager_id`),
```

```
       ADD   CONSTRAINT   `contracts_ibfk_2`   FOREIGN   KEY   (`client_id`)
REFERENCES `clients` (`client_id`),
       ADD CONSTRAINT `contracts_ibfk_3` FOREIGN KEY (`car_id`) REFERENCES
`cars` (`car_id`);
COMMIT;
```

Приклад коду В.1 – Фрагмент реалізації програми в файлі **import_data.php**:

```
<?php
use PhpOffice\PhpSpreadsheet\Reader\Xlsx;
use RentCar\Car;

// Include PhpSpreadsheet library autoloader
require_once '../../vendor/autoload.php';

include '../config.php';
include '../classes/Car.php';

if ( isset( $_POST['import_submit'] ) ) {

  // Allowed mime types
  $excelMimes = array(
         'text/xls',
         'text/xlsx',
         'application/excel',
         'application/vnd.msexcel',
         'application/vnd.ms-excel',
         'application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet'
  );

  // Validate whether selected file is a Excel file
  if ( ! empty( $_FILES['file']['name'] ) && in_array(
$_FILES['file']['type'], $excelMimes ) ) {

         // If the file is uploaded
         if ( is_uploaded_file( $_FILES['file']['tmp_name'] ) ) {
                $reader        = new Xlsx();
                $spreadsheet   = $reader->load(
$_FILES['file']['tmp_name'] );
                $worksheet     = $spreadsheet->getActiveSheet();
                $worksheet_arr = $worksheet->toArray();

                // Remove header row
                unset( $worksheet_arr[0] );

                foreach ( $worksheet_arr as $row ) {
                       $state_number        = $row[0];
                       $brand               = $row[1];
                       $year_of_manufacture = $row[2];
                       $body_number         = $row[3];
                       $cost_per_day        = $row[4];
                       $photo               = $row[5];
```

```
                            if ( ! empty( $state_number )
                                 && ! empty( $brand )
                                 && ! empty( $year_of_manufacture )
                                 && ! empty( $body_number )
                                 && ! empty( $cost_per_day ) ) {
                                   // Check whether member already exists
in the database with the same email
                                   $prevQuery  = "SELECT car_id FROM cars
WHERE state_number = ?";
                                   $prevResult = $pdo->prepare( $prevQuery
);
                                   $prevResult->execute( [ $state_number ]
);
                                   $number_of_rows = $prevResult-
>fetchColumn();

                                   if ( (int) $number_of_rows > 0 ) {
                                          $car_data = [
                                                 'update_state_number'
=> $state_number,
                                                 'update_brand'
=> $brand,

       'update_year_of_manufacture' => $year_of_manufacture,
                                                 'update_body_number'
=> $body_number,
                                                 'update_cost_per_day'
=> $cost_per_day,
                                                 'update_photo'
=> $photo,
                                                 'import'
=> true
                                          ];
                                          // Update member data in the
database
                                          Car::get_instance()->update_car(
$pdo, $car_data );
                                   } else {
                                          $car_data = [
                                                 'state_number'        =>
$state_number,
                                                 'brand'               =>
$brand,
                                                 'year_of_manufacture' =>
$year_of_manufacture,
                                                 'body_number'         =>
$body_number,
                                                 'cost_per_day'        =>
$cost_per_day,
                                                 'photo'               =>
$photo
                                          ];
                                          // Insert member data in the
database
                                          Car::get_instance()->add_car(
$pdo, $car_data );
                                   }
```

```
                        }
                    }

                    $qstring = '?status=succ';
                } else {
                    $qstring = '?status=err';
                }
            } else {
                $qstring = '?status=invalid_file';
            }
        }

        // Redirect to the listing page
        header( "Location: /cars.php" . $qstring );
```

Приклад коду В.2 – Фрагмент реалізації тригерів атрибуту *status*:

```
        DELIMITER $$
        CREATE TRIGGER update_car_status
            AFTER UPDATE ON contracts
            FOR EACH ROW
        BEGIN
            DECLARE new_car_id INT;
            DECLARE new_contract_status VARCHAR(255);
            DECLARE new_car_status VARCHAR(255);
            -- Перевіряємо, чи статус договору змінився
            IF OLD.status != NEW.status THEN
                -- Отримуємо ID автомобіля зі зміненого договору
                SET new_car_id = NEW.car_id;
                -- Отримуємо новий статус договору
                SET new_contract_status = NEW.status;
                --  Отримуємо  новий  статус  автомобіля  згідно  зі  зміненим
статусом договору
                CASE new_contract_status
                    WHEN 'created' THEN SET new_car_status = 'available';
                    WHEN  'awaiting_payment'  THEN  SET  new_car_status  =
'booked';
                    WHEN 'active' THEN SET new_car_status = 'in_rent';
                    WHEN 'canceled' THEN SET new_car_status = 'available';
                    WHEN 'completed' THEN SET new_car_status = 'available';
                END CASE;
                -- Оновлюємо статус автомобіля
                UPDATE cars
                SET status = new_car_status
                WHERE new_car_id = car_id;
            END IF;
        END$$
        DELIMITER ;

        DELIMITER $$
        CREATE TRIGGER set_initial_contract_status
        BEFORE INSERT ON contracts
        FOR EACH ROW
        BEGIN
            SET NEW.status = 'created';
```

```
END;
$$
DELIMITER ;

DELIMITER $$
DELIMITER $$
CREATE TRIGGER check_contract_status_sequence
    BEFORE UPDATE ON contracts
    FOR EACH ROW
BEGIN
    DECLARE old_contract_status VARCHAR(255);
    DECLARE new_contract_status VARCHAR(255);

    -- Отримуємо старий та новий статуси договору
    SET old_contract_status = OLD.status;
    SET new_contract_status = NEW.status;

    -- Перевіряємо послідовність зміни статусів
    IF (old_contract_status = 'created' AND new_contract_status !=
'awaiting_payment') OR
        (old_contract_status = 'awaiting_payment' AND
new_contract_status != 'active') OR
        (old_contract_status = 'active' AND new_contract_status NOT IN
('completed', 'canceled')) OR
        (old_contract_status = 'completed' AND new_contract_status !=
'canceled') OR
        (old_contract_status = 'canceled' AND new_contract_status =
'active') OR
        (old_contract_status = 'completed' AND new_contract_status =
'active') THEN
         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Недопустима
послідовність зміни статусів договору';
END IF;
END$$
DELIMITER;
```

Приклад коду В.3 – Код реалізації методів на PHP:

```
<?php

public function get_min_cost( $pdo ) {
    if ( empty( $pdo ) ) {
       return;
    }
    try {
       $sql   = "SELECT MIN(" . self::TABLE_COLUMN_COST_PER_DAY . ")
as min_cost FROM " . $this->table_name;
       $stmt = $pdo->prepare( $sql );
       $stmt->execute();

       return $stmt->fetch( \PDO::FETCH_ASSOC );

    } catch ( Exception $e ) {
       $pdo->rollback();

       return "Error: $e";
```

```
        }
    }

    public function get_max_cost( $pdo ) {
        if ( empty( $pdo ) ) {
            return array();
        }
        try {
            $sql   = "SELECT MAX(" . self::TABLE_COLUMN_COST_PER_DAY . ")
as max_cost FROM " . $this->table_name;
            $stmt = $pdo->prepare( $sql );
            $stmt->execute();

            return $stmt->fetch( \PDO::FETCH_ASSOC );

        } catch ( Exception $e ) {
            $pdo->rollback();

            return "Error: $e";
        }
    }

    public function get_filter_cars( $pdo, $filter_data ) {
        if ( empty( $filter_data ) ) {
            return array();
        }
        try {
            $state_number  =  ! empty(  $filter_data['state_number']  )  ?
htmlspecialchars( trim( $filter_data['state_number'] ) ) : '';
            $brand                =  !  empty(  $filter_data['brand']  )  ?
htmlspecialchars( trim( $filter_data['brand'] ) ) : '';
            $body_number    =  !  empty(  $filter_data['body_number']  )  ?
htmlspecialchars( trim( $filter_data['body_number'] ) ) : '';
            $order_by      = ! empty( $filter_data['order_by'] ) && '-1'
!==      $filter_data['order_by']      ?      htmlspecialchars(      trim(
$filter_data['order_by'] ) ) : '';
            $sort                  =  !  empty(  $filter_data['sort']  )  ?
mb_strtoupper(  htmlspecialchars(  trim(  $filter_data['sort']  )  )  )  :
'DESC';

            $year_of_manufacture_from          =          !          empty(
$filter_data['year_of_manufacture_from']      )      &&      '-1'      !==
$filter_data['year_of_manufacture_from']    ?    htmlspecialchars(    trim(
$filter_data['year_of_manufacture_from'] ) ) : '';
            $year_of_manufacture_to                    =      !      empty(
$filter_data['year_of_manufacture_to']       )       &&       '-1'       !==
$filter_data['year_of_manufacture_to']    ?    htmlspecialchars(    trim(
$filter_data['year_of_manufacture_to'] ) ) : '';

            $cost_from  =  ! empty(  $filter_data['cost_from']  )  ?  (int)
$filter_data['cost_from'] : 0;
            $cost_to     =  !  empty(  $filter_data['cost_to']  )  ?  (int)
$filter_data['cost_to'] : 0;

            $sql_state_number = '';
            if ( ! empty( $state_number ) ) {
```

```
                $sql_state_number = " " . self::TABLE_COLUMN_STATE_NUMBER .
' LIKE "%' . mb_strtoupper( $state_number ) . '%" OR ';
            }

            $sql_brand = '';
            if ( ! empty( $brand ) ) {
                $sql_brand = " " . self::TABLE_COLUMN_BRAND . ' LIKE "%' .
$brand . '%" OR ';
            }

            $sql_body_number = '';
            if ( ! empty( $body_number ) ) {
                $sql_body_number = " " . self::TABLE_COLUMN_BODY_NUMBER . '
LIKE "%' . $body_number . '%" AND ';
            }

            $sql_year_of_manufacture = '';
            if ( $year_of_manufacture_from && $year_of_manufacture_to &&
(int) $year_of_manufacture_from < (int) $year_of_manufacture_to ) {
                $sql_year_of_manufacture = " " .
self::TABLE_COLUMN_YEAR_OF_MANUFACTURE . " BETWEEN " .
$year_of_manufacture_from . " AND " . $year_of_manufacture_to . " AND ";
            }

            $sql_cost = '';
            if ( $cost_from && $cost_to && $cost_from < $cost_to ) {
                $sql_cost = " " . self::TABLE_COLUMN_COST_PER_DAY . "
BETWEEN " . $cost_from . " AND " . $cost_to . " AND ";
            }

            if ( ! empty( $order_by ) ) {
                $sql_order_by = ' ORDER BY ' . $order_by . ' ' . $sort;
            } else {
                $sql_order_by = ' ORDER BY  cars.car_id ' . $sort;
            }

            $sql  = "SELECT * FROM " . $this->table_name . " WHERE" .
$sql_state_number . $sql_brand . $sql_body_number .
$sql_year_of_manufacture . $sql_cost . " " .
self::TABLE_COLUMN_BODY_NUMBER . " IS NOT NULL " . $sql_order_by;
            $stmt = $pdo->prepare( $sql );
            $stmt->execute();

            return $stmt->fetchAll( \PDO::FETCH_ASSOC );

        } catch ( Exception $e ) {
            $pdo->rollback();

            return "Error: $e";
        }
    }
```